



# System Management

Exploring Palm OS®

Written by Greg Wilson  
Edited by Jean Ostrem  
Technical assistance from Najeeb Abdulrahiman, Jie Su

Copyright © 1996–2004, PalmSource, Inc. and its affiliates. All rights reserved. This technical documentation contains confidential and proprietary information of PalmSource, Inc. (“PalmSource”), and is provided to the licensee (“you”) under the terms of a Nondisclosure Agreement, Product Development Kit license, Software Development Kit license or similar agreement between you and PalmSource. You must use commercially reasonable efforts to maintain the confidentiality of this technical documentation. You may print and copy this technical documentation solely for the permitted uses specified in your agreement with PalmSource. In addition, you may make up to two (2) copies of this technical documentation for archival and backup purposes. All copies of this technical documentation remain the property of PalmSource, and you agree to return or destroy them at PalmSource’s written request. Except for the foregoing or as authorized in your agreement with PalmSource, you may not copy or distribute any part of this technical documentation in any form or by any means without express written consent from PalmSource, Inc., and you may not modify this technical documentation or make any derivative work of it (such as a translation, localization, transformation or adaptation) without express written consent from PalmSource.

PalmSource, Inc. reserves the right to revise this technical documentation from time to time, and is not obligated to notify you of any revisions.

THIS TECHNICAL DOCUMENTATION IS PROVIDED ON AN “AS IS” BASIS. NEITHER PALMSOURCE NOR ITS SUPPLIERS MAKES, AND EACH OF THEM EXPRESSLY EXCLUDES AND DISCLAIMS TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, ANY REPRESENTATIONS OR WARRANTIES REGARDING THIS TECHNICAL DOCUMENTATION, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING WITHOUT LIMITATION ANY WARRANTIES IMPLIED BY ANY COURSE OF DEALING OR COURSE OF PERFORMANCE AND ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, ACCURACY, AND SATISFACTORY QUALITY. PALMSOURCE AND ITS SUPPLIERS MAKE NO REPRESENTATIONS OR WARRANTIES THAT THIS TECHNICAL DOCUMENTATION IS FREE OF ERRORS OR IS SUITABLE FOR YOUR USE. TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, PALMSOURCE, INC. ALSO EXCLUDES FOR ITSELF AND ITS SUPPLIERS ANY LIABILITY, WHETHER BASED IN CONTRACT OR TORT (INCLUDING NEGLIGENCE), FOR DIRECT, INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL, EXEMPLARY OR PUNITIVE DAMAGES OF ANY KIND ARISING OUT OF OR IN ANY WAY RELATED TO THIS TECHNICAL DOCUMENTATION, INCLUDING WITHOUT LIMITATION DAMAGES FOR LOST REVENUE OR PROFITS, LOST BUSINESS, LOST GOODWILL, LOST INFORMATION OR DATA, BUSINESS INTERRUPTION, SERVICES STOPPAGE, IMPAIRMENT OF OTHER GOODS, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR OTHER FINANCIAL LOSS, EVEN IF PALMSOURCE, INC. OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR IF SUCH DAMAGES COULD HAVE BEEN REASONABLY FORESEEN.

PalmSource, Palm OS, Palm Powered, Graffiti, HotSync, and certain other trademarks and logos are trademarks or registered trademarks of PalmSource, Inc. or its affiliates in the United States, France, Germany, Japan, the United Kingdom, and other countries. These marks may not be used in connection with any product or service that does not belong to PalmSource, Inc. (except as expressly permitted by a license with PalmSource, Inc.), in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits PalmSource, Inc., its licensor, its subsidiaries, or affiliates. All other product and brand names may be trademarks or registered trademarks of their respective owners.

IF THIS TECHNICAL DOCUMENTATION IS PROVIDED ON A COMPACT DISC, THE SOFTWARE AND OTHER DOCUMENTATION ON THE COMPACT DISC ARE SUBJECT TO THE LICENSE AGREEMENTS ACCOMPANYING THE SOFTWARE AND OTHER DOCUMENTATION.

Exploring Palm OS: System Management  
Document Number 3110-003  
November 9, 2004  
For the latest version of this document, visit  
<http://www.palmos.com/dev/support/docs/>.

PalmSource, Inc.  
1240 Crossman Avenue  
Sunnyvale, CA 94089  
USA  
[www.palmsource.com](http://www.palmsource.com)

# Table of Contents

---

<b>About This Document</b>	<b>xxv</b>
The <i>Exploring Palm OS</i> Series . . . . .	xxv
Additional Resources . . . . .	xxvi
Changes to This Document . . . . .	xxvii
3110-002. . . . .	xxvii
3110-001. . . . .	xxvii

## Part I: Concepts

<b>1 Attentions and Alarms</b>	<b>3</b>
Getting the User's Attention . . . . .	3
The Role of the Attention Manager . . . . .	3
Attention Manager Operation . . . . .	5
Getting the User's Attention . . . . .	11
Attentions and Alarms . . . . .	20
Detecting and Updating Pending Attentions . . . . .	20
Detecting Device Capabilities . . . . .	23
Controlling the Attention Indicator . . . . .	23
Alarms . . . . .	24
Setting an Alarm . . . . .	25
Alarm Scenario . . . . .	26
Summary of Attentions and Alarms . . . . .	28
<b>2 Features</b>	<b>31</b>
The Operating System Version Feature . . . . .	31
Application-Defined Features . . . . .	33
Using the Feature Manager . . . . .	34
Feature Memory . . . . .	34
Feature Memory Limitations. . . . .	35
<b>3 Preferences</b>	<b>37</b>
Accessing System Preferences . . . . .	37
Setting System Preferences. . . . .	38

---

Setting Application-Specific Preferences . . . . .	41
When to Use Application Preferences . . . . .	41
How to Store Preferences . . . . .	42
Which Preferences Database to Use . . . . .	43
Updating Preferences Upon a New Release . . . . .	45
<b>4 Sound</b>	<b>49</b>
Playing Simple Sounds . . . . .	50
Sound Preferences . . . . .	50
<b>5 Expansion</b>	<b>53</b>
Expansion Support . . . . .	53
Primary vs. Secondary Storage . . . . .	54
Expansion Slot . . . . .	54
Universal Connector . . . . .	55
Architectural Overview . . . . .	55
Block Device Drivers . . . . .	56
File Systems . . . . .	57
VFS Manager . . . . .	58
Expansion Manager . . . . .	59
Applications on Cards. . . . .	60
Card Insertion and Removal . . . . .	61
Start.prc. . . . .	66
Checking for Expansion Cards . . . . .	67
Verifying Handheld Compatibility . . . . .	67
Checking for Mounted Volumes . . . . .	67
Enumerating Slots . . . . .	68
Determining a Card's Capabilities . . . . .	69
Summary of Expansion Manager . . . . .	70
<b>6 Shared Libraries</b>	<b>71</b>
Exporting Globals . . . . .	73
Patching Shared Libraries . . . . .	74
Patch Configuration Database . . . . .	74
Constructing a Patch . . . . .	74
Registering the Patch . . . . .	77

---

The Patch Call Chain . . . . .	77
Adding and Removing Patches . . . . .	80
Patch Security . . . . .	81
The Program Loader . . . . .	81
Program Loader Library Functions . . . . .	81
Shared Libraries and the Program Loader . . . . .	81
<b>7 System Reset</b>	<b>83</b>
Soft Reset . . . . .	83
Safe-Mode Reset . . . . .	83
Hard Reset. . . . .	83
System Reset Calls . . . . .	84
<b>8 Threading</b>	<b>85</b>
Architecture Overview . . . . .	85
Threads and Scheduling. . . . .	85
Processes and Applications . . . . .	87
Thread Synchronization. . . . .	89
Inter-Process Communication . . . . .	90
Graphics Context. . . . .	92
Using the Threading APIs . . . . .	92
Application Launching . . . . .	92
Launching in the Background Process. . . . .	93
Manipulating Threads . . . . .	93
Inter-Process Communication (IPC). . . . .	95
Atomic Operations . . . . .	97
Synchronization . . . . .	97
Thread-Specific Data . . . . .	97
Accessing the User Interface from Outside the Main UI Thread	98
Summary of Threading . . . . .	99
<b>9 Power Management</b>	<b>101</b>
Palm OS Power Modes . . . . .	101
Guidelines for Application Developers . . . . .	103
Power Management Calls . . . . .	103

---

<b>10 The ROM Serial Number</b>	<b>105</b>
<b>11 Time</b>	<b>107</b>
Using Real-Time Clock Functions . . . . .	107
Using System Ticks Functions . . . . .	108
<b>12 Floating Point</b>	<b>109</b>
Summary of Float Manager . . . . .	110
<b>13 Debugging Strategies</b>	<b>113</b>
Displaying Development Errors . . . . .	113
Setting the Build Type. . . . .	114
Displaying Error Messages . . . . .	115
The Try-and-Catch Mechanism . . . . .	116
Using the Try-and-Catch Mechanism . . . . .	117
Summary of Debugging API . . . . .	118

## Part II: Reference

<b>14 Alarm Manager</b>	<b>121</b>
Alarm Manager Structures and Types . . . . .	122
AlmProcCmdEnum . . . . .	122
SysAlarmTriggeredParamType . . . . .	122
SysDisplayAlarmParamType . . . . .	123
Alarm Manager Constants . . . . .	124
Alarm Manager Error Codes . . . . .	124
AlmProcCmdEnumTag . . . . .	124
Alarm Manager Launch Codes . . . . .	125
sysAppLaunchCmdAlarmTriggered . . . . .	125
sysAppLaunchCmdDisplayAlarm . . . . .	125
Alarm Manager Functions and Macros . . . . .	126
AlmEnableNotification . . . . .	126
AlmGetAlarm . . . . .	126
AlmSetAlarm . . . . .	127

---

<b>15 Attention Manager</b>	<b>129</b>
Attention Manager Structures and Types. . . . .	129
AttnCommandArgsDrawDetailTag . . . . .	129
AttnCommandArgsDrawListTag . . . . .	130
AttnCommandArgsGotItTag . . . . .	132
AttnCommandArgsIterateTag . . . . .	132
AttnCommandArgsType . . . . .	133
AttnCommandType . . . . .	135
AttnFlagsType . . . . .	135
AttnLaunchCodeArgsType . . . . .	136
AttnLevelType . . . . .	137
AttnNotifyDetailsType . . . . .	138
Attention Manager Constants . . . . .	138
Attention Manager Error Codes . . . . .	138
Drawing Constants . . . . .	138
Feature Constants . . . . .	139
Feature Masks . . . . .	140
Feature Flags . . . . .	140
Attention Flags . . . . .	141
Attention Override Flags . . . . .	142
Commands . . . . .	143
Attention Levels . . . . .	144
Attention Manager Launch Codes . . . . .	145
sysAppLaunchCmdAttention . . . . .	145
Attention Manager Functions and Macros . . . . .	145
AttnDoSpecialEffects . . . . .	145
AttnForgetIt . . . . .	146
AttnForgetItV40 . . . . .	146
AttnGetAttention . . . . .	148
AttnGetAttentionV40 . . . . .	149
AttnGetCounts. . . . .	152
AttnGetCountsV40 . . . . .	152
AttnIndicatorEnable . . . . .	154
AttnIndicatorEnabled. . . . .	155
AttnIterate . . . . .	155

---

AttnIterateV40 . . . . .	156
AttnListOpen . . . . .	157
AttnUpdate . . . . .	157
AttnUpdateV40 . . . . .	159
<b>16 Category Manager Sync</b>	<b>161</b>
Category Manager Sync Functions and Macros . . . . .	161
CatMgrSyncGetModifiedCategories . . . . .	161
CatMgrSyncGetPurgeCounter . . . . .	163
CatMgrSyncReleaseStorage . . . . .	163
<b>17 Common Battery Types</b>	<b>165</b>
Common Battery Types Structures and Types. . . . .	165
SysBatteryKind . . . . .	165
SysBatteryState . . . . .	165
Common Battery Types Constants . . . . .	166
SysBatteryKindTag . . . . .	166
SysBatteryStateTag . . . . .	167
<b>18 Common Error Codes</b>	<b>169</b>
Common Error Codes Constants . . . . .	169
Error Code Classes . . . . .	169
. . . . .	175
. . . . .	175
Binder Errors . . . . .	176
DAL Errors . . . . .	177
Media Manager Errors . . . . .	177
System Errors . . . . .	179
. . . . .	182
<b>19 Cyclic Redundancy Check</b>	<b>183</b>
CRC Functions and Macros . . . . .	183
Crc16CalcBigBlock . . . . .	183
Crc16CalcBlock . . . . .	184
Crc32CalcBlock . . . . .	184

---

<b>20 DateTime</b>	<b>187</b>
DateTime Structures and Types . . . . .	188
DateFormatType . . . . .	188
DateTimeType . . . . .	188
DateType . . . . .	189
DaylightSavingsTypes . . . . .	189
DayOfMonthType . . . . .	189
DayOfWeekType . . . . .	190
TimeFormatType . . . . .	190
TimeType . . . . .	190
DateTime Constants. . . . .	191
String Lengths . . . . .	191
Months . . . . .	191
Days . . . . .	192
Conversions . . . . .	192
Template Formatting Characters . . . . .	193
Miscellaneous DateTime Constants . . . . .	194
DateFormatTag . . . . .	195
Template Value Types . . . . .	196
DaylightSavingsTag . . . . .	197
DayOfWeekTag . . . . .	198
TimeFormatTag . . . . .	200
DateTime Functions and Macros . . . . .	202
DateAdjust . . . . .	202
DateDaysToDate . . . . .	202
DateSecondsToDate . . . . .	203
DateTemplateToAscii . . . . .	203
DateToAscii . . . . .	206
DateToDays . . . . .	207
DateToDOWDMFormat. . . . .	208
DateToInt . . . . .	209
DayOfMonth . . . . .	209
DayOfWeek . . . . .	210
DaysInMonth . . . . .	210
TimAdjust. . . . .	211

---

TimDateTimeToSeconds . . . . .	211
TimeGetFormatSeparator . . . . .	212
TimeGetFormatSuffix . . . . .	212
TimeIs24HourFormat . . . . .	213
TimeToAscii . . . . .	213
TimeToInt . . . . .	214
TimeZoneToAscii. . . . .	214
TimeZoneToAsciiV50 . . . . .	215
TimSecondsToDateTime. . . . .	216
TimTimeZoneToUTC . . . . .	216
TimUTCToTimeZone . . . . .	217

**21 Debug Manager 219**

Debug Manager Functions and Macros . . . . .	219
DbgBreak . . . . .	219
DbgBreakMessage . . . . .	220
DbgBreakMessageIf . . . . .	220
DbgFatalErrorInContext. . . . .	221
DbgGetChar. . . . .	221
DbgIsPresent . . . . .	222
DbgLookupSymbol. . . . .	222
DbgMessage. . . . .	223
DbgOutputSync . . . . .	223
DbgPrintf . . . . .	223
DbgRestartMallocProfiling . . . . .	224
DbgSetMallocProfiling . . . . .	224
DbgUnmangleSymbol . . . . .	225
DbgVPrintf . . . . .	225
DbgWalkStack . . . . .	226

**22 Desktop Link Server 227**

Desktop Link Server Structures and Types . . . . .	227
DlkCallAppReplyParamType . . . . .	227
DlkCtlEnum . . . . .	228
Desktop Link Server Constants . . . . .	228
. . . . .	228

---

Miscellaneous Desktop Link Server Constants . . . . .	229
DlkCtlEnumTag . . . . .	229
DlkSyncStateType . . . . .	230
Desktop Link Server Functions and Macros . . . . .	231
DlkControl . . . . .	231
DlkGetSyncInfo . . . . .	234
DlkSetLogEntry . . . . .	236
<b>23 Error Manager</b>	<b>237</b>
ErrorManager Constants. . . . .	237
ErrDlgResultType . . . . .	237
Error Manager Functions and Macros . . . . .	238
DbgOnlyFatalError . . . . .	238
DbgOnlyFatalErrorIf . . . . .	238
ErrAlert. . . . .	239
ErrDisplay . . . . .	240
ErrDisplayFileLineMsg . . . . .	240
ErrFatalDisplay . . . . .	241
ErrFatalDisplayIf . . . . .	241
ErrFatalError . . . . .	242
ErrFatalErrorIf . . . . .	242
ErrFatalErrorInContext . . . . .	243
ErrGetErrorMsg . . . . .	243
ErrNonFatalDisplay . . . . .	244
ErrNonFatalDisplayIf . . . . .	245
<b>24 ErrTryCatch</b>	<b>247</b>
ErrTryCatch Structures and Types. . . . .	247
ErrExceptionType . . . . .	247
ErrJumpBuf . . . . .	248
ErrTryCatch Functions and Macros . . . . .	248
ErrCatch . . . . .	248
ErrCatchWithAddress . . . . .	249
ErrEndCatch . . . . .	250
ErrExceptionListAppend . . . . .	250
ErrExceptionListGetByThreadID . . . . .	251

---

ErrExceptionListRemove . . . . .	251
ErrLongJump . . . . .	252
ErrSetJump . . . . .	252
ErrThrow . . . . .	253
ErrThrowIf . . . . .	253
ErrThrowWithAddress . . . . .	253
ErrThrowWithHandler . . . . .	254
ErrTry . . . . .	255

**25 Expansion Manager 257**

Expansion Manager Structures and Types . . . . .	257
CardMetricsType . . . . .	257
ExpCardInfoType . . . . .	259
Expansion Manager Constants . . . . .	260
Expansion Manager Error Codes . . . . .	260
Defined Media Types . . . . .	262
Capability Flags . . . . .	263
Enumeration Constants . . . . .	263
Partition Type Flags . . . . .	263
Miscellaneous Expansion Manager Constants . . . . .	264
Expansion Manager Functions and Macros. . . . .	265
ExpCardInfo. . . . .	265
ExpCardIsFilesystemSupported . . . . .	266
ExpCardMediaType . . . . .	266
ExpCardMetrics . . . . .	267
ExpCardPresent . . . . .	268
ExpCardSectorRead . . . . .	269
ExpCardSectorWrite . . . . .	270
ExpSlotCustomControl . . . . .	271
ExpSlotEnumerate . . . . .	272
ExpSlotMediaType . . . . .	273
ExpSlotPowerCheck . . . . .	274

**26 Fatal Alert 277**

Fatal Alert Constants . . . . .	277
Fatal Alert Actions . . . . .	277

---

Fatal Alert Functions and Macros . . . . .	278
SysFatalAlert . . . . .	278
SysFatalAlertInit . . . . .	278
<b>27 Feature Manager</b>	<b>279</b>
Feature Manager Constants . . . . .	280
Feature Manager Error Codes . . . . .	280
Feature Manager Functions and Macros . . . . .	280
FtrGet . . . . .	280
FtrGetByIndex . . . . .	281
FtrPtrFree . . . . .	282
FtrPtrGet . . . . .	282
FtrPtrNew . . . . .	283
FtrPtrResize . . . . .	284
FtrSet . . . . .	285
FtrUnregister . . . . .	286
<b>28 Float Manager</b>	<b>287</b>
Float Manager Constants . . . . .	287
Float Manager Error Codes . . . . .	287
Miscellaneous Float Manager Constants . . . . .	287
Float Manager Functions and Macros . . . . .	288
FlpAddDouble . . . . .	288
FlpAddFloat . . . . .	288
FlpBase10Info . . . . .	289
FlpCompareDoubleEqual . . . . .	289
FlpCompareDoubleLessThan . . . . .	290
FlpCompareDoubleLessThanOrEqual . . . . .	291
FlpCompareFloatEqual . . . . .	291
FlpCompareFloatLessThan . . . . .	292
FlpCompareFloatLessThanOrEqual . . . . .	292
FlpCorrectedAdd . . . . .	293
FlpCorrectedSub . . . . .	294
FlpDivDouble . . . . .	295
FlpDivFloat . . . . .	295
FlpDoubleToFloat . . . . .	296

---

FlpDoubleToInt32 . . . . .	296
FlpDoubleToLongDouble . . . . .	297
FlpDoubleToLongLong . . . . .	297
FlpDoubleToUInt32. . . . .	297
FlpDoubleToULongLong . . . . .	298
FlpFloatToDouble . . . . .	298
FlpFloatToInt32 . . . . .	299
FlpFloatToLongDouble . . . . .	299
FlpFloatToLongLong . . . . .	299
FlpFloatToUInt32. . . . .	300
FlpFloatToULongLong . . . . .	300
FlpFToA . . . . .	301
FlpGetExponent . . . . .	301
FlpInt32ToDouble . . . . .	301
FlpInt32ToFloat . . . . .	302
FlpLongDoubleToDouble . . . . .	302
FlpLongDoubleToFloat . . . . .	302
FlpLongLongToDouble . . . . .	303
FlpLongLongToFloat . . . . .	303
FlpMulDouble . . . . .	304
FlpMulFloat . . . . .	304
FlpNegDouble . . . . .	305
FlpNegFloat . . . . .	305
FlpSubDouble . . . . .	305
FlpSubFloat . . . . .	306
FlpUInt32ToDouble. . . . .	306
FlpUInt32ToFloat. . . . .	307
FlpULongLongToDouble . . . . .	307
FlpULongLongToFloat . . . . .	307

**29 Host Control 309**

Host Control Structures and Types . . . . .	310
HostBool . . . . .	310
HostBoolType . . . . .	310
HostClockType . . . . .	310

---

HostControlSelectorType . . . . .	310
HostControlTrapNumber . . . . .	311
HostDirEntType . . . . .	311
HostDIRType . . . . .	311
HostErr . . . . .	312
HostErrType . . . . .	312
HostFILE . . . . .	312
HostFILEType . . . . .	312
HostID . . . . .	313
HostIDType . . . . .	313
HostPlatform . . . . .	313
HostPlatformType . . . . .	313
HostSignal . . . . .	314
HostSignalType . . . . .	314
HostSizeType . . . . .	314
HostStatType . . . . .	314
HostTimeType . . . . .	316
HostTmType . . . . .	317
HostUTimeType . . . . .	318
Host Control Constants . . . . .	318
Host Control Function Selectors . . . . .	318
Host Control Error Codes . . . . .	323
Host IDs . . . . .	325
Host Platforms . . . . .	326
Host Signals . . . . .	326
File Attributes . . . . .	327
Error Levels . . . . .	327
Miscellaneous Host Control Constants . . . . .	327
Host Control Functions and Macros . . . . .	328
HostAscTime . . . . .	328
HostClock. . . . .	328
HostCloseDir . . . . .	329
HostControl . . . . .	329
HostCTime . . . . .	329
HostEnteringApp . . . . .	330

---

HostErrNo . . . . .	330
HostExgLibAccept . . . . .	331
HostExgLibClose. . . . .	331
HostExgLibConnect . . . . .	331
HostExgLibControl. . . . .	331
HostExgLibDisconnect . . . . .	331
HostExgLibGet . . . . .	332
HostExgLibHandleEvent . . . . .	332
HostExgLibOpen. . . . .	332
HostExgLibPut. . . . .	332
HostExgLibReceive. . . . .	333
HostExgLibRequest . . . . .	333
HostExgLibSend . . . . .	333
HostExgLibSleep. . . . .	333
HostExgLibWake. . . . .	334
HostExitedApp . . . . .	334
HostExportFile. . . . .	334
HostFClose . . . . .	335
HostFEOF. . . . .	335
HostFError . . . . .	336
HostFFlush . . . . .	336
HostFGetC . . . . .	336
HostFGetPos . . . . .	337
HostFGetS. . . . .	337
HostFOpen . . . . .	338
HostFPrintF . . . . .	339
HostFPutC . . . . .	339
HostFPutS. . . . .	340
HostFRead . . . . .	340
HostFree . . . . .	341
HostFReopen . . . . .	341
HostFScanF . . . . .	342
HostFSeek. . . . .	342
HostFSetPos. . . . .	343
HostFTell . . . . .	343

---

HostFWrite . . . . .	344
HostGestalt . . . . .	344
HostGetChar . . . . .	345
HostGetDirectory . . . . .	345
HostGetEnv . . . . .	345
HostGetFile . . . . .	346
HostGetFileAttr . . . . .	346
HostGetFirstApp. . . . .	347
HostGetHostID . . . . .	347
HostGetHostPlatform. . . . .	347
HostGetHostVersion . . . . .	348
HostGetPreference . . . . .	348
HostGMTIME . . . . .	349
HostHostControl68K . . . . .	350
HostImportFile . . . . .	350
HostIsCallingTrap . . . . .	350
HostIsSelectorImplemented . . . . .	351
HostLocalTime. . . . .	351
HostLogEvent . . . . .	352
HostLogFile . . . . .	352
HostMalloc . . . . .	352
HostMkDir . . . . .	353
HostMkTime . . . . .	353
HostOpenDir . . . . .	353
HostPrintF . . . . .	354
HostProfileCleanup. . . . .	354
HostProfileDetailFn . . . . .	355
HostProfileDump . . . . .	355
HostProfileInit . . . . .	356
HostProfileStart . . . . .	357
HostProfileStop . . . . .	357
HostPutFile . . . . .	358
HostReadDir . . . . .	358
HostRealloc . . . . .	359
HostRemove. . . . .	359

---

HostRename. . . . .	360
HostRmdir . . . . .	360
HostSessionClose . . . . .	361
HostSessionCreate . . . . .	361
HostSessionOpen . . . . .	362
HostSessionQuit . . . . .	362
HostSetErrorLevel . . . . .	363
HostSetFileAttr . . . . .	363
HostSetLogFileSize . . . . .	364
HostSetPreference . . . . .	364
HostSignalResume . . . . .	365
HostSignalSend . . . . .	365
HostSignalWait . . . . .	366
HostSlotHasCard. . . . .	367
HostSlotMax. . . . .	367
HostSlotRoot . . . . .	368
HostStat . . . . .	368
HostStrFTime . . . . .	369
HostTime . . . . .	369
HostTmpFile . . . . .	370
HostTmpNam . . . . .	370
HostTraceClose . . . . .	370
HostTraceInit . . . . .	371
HostTraceOutputB . . . . .	371
HostTraceOutputT . . . . .	372
HostTraceOutputTL . . . . .	373
HostTraceOutputVT . . . . .	374
HostTraceOutputVTL. . . . .	375
HostTruncate . . . . .	375
HostUTime . . . . .	376
HostVFPrintF . . . . .	376
HostVFScanF . . . . .	377
HostVPrintF. . . . .	377

---

<b>30 Loader</b>	<b>379</b>
Loader Structures and Types . . . . .	380
SysModuleInfoType . . . . .	380
SysPatchInfoType . . . . .	381
Loader Constants . . . . .	382
Miscellaneous Loader Constants . . . . .	382
Loader Launch Codes . . . . .	383
sysPatchLaunchCmdClearInfo . . . . .	383
sysPatchLaunchCmdSetInfo . . . . .	383
Loader Functions and Macros . . . . .	384
SysGetEntryAddresses . . . . .	384
SysGetModuleDatabase . . . . .	385
SysGetModuleGlobals . . . . .	386
SysGetModuleInfo . . . . .	387
SysGetModuleInfoByDatabaseID . . . . .	388
SysGetRefNum . . . . .	389
SysLoadModule . . . . .	389
SysLoadModuleByDatabaseID . . . . .	392
SysRegisterPatch . . . . .	393
SysUnloadModule . . . . .	394
SysUnregisterPatch . . . . .	394
Application-Defined Functions . . . . .	395
SysMainEntryPtrType . . . . .	395
<b>31 Patch</b>	<b>397</b>
Patch Structures and Types. . . . .	397
SysPatchEntryNumType . . . . .	397
SysPatchTargetHeaderType . . . . .	398
Patch Constants . . . . .	399
Patch Flags . . . . .	399
Miscellaneous Patch Constants . . . . .	399
<b>32 PerfDriver</b>	<b>401</b>
PerfDriver Structures and Types . . . . .	401
PerfGenCPUClockInfoType . . . . .	401
PerfRefNumType . . . . .	402

---

PerfResultType . . . . .	402
PerfDriver Constants . . . . .	402
. . . . .	402
<b>33 Preferences</b>	<b>405</b>
Preferences Structures and Types . . . . .	405
PrefActivePanelParamsType . . . . .	405
Preferences Constants . . . . .	406
MeasurementSystemType . . . . .	406
SoundLevelTypeV20 . . . . .	406
SystemPreferencesChoice . . . . .	406
Preferences Launch Codes . . . . .	416
prefAppLaunchCmdSetActivePanel . . . . .	416
Preferences Functions and Macros . . . . .	416
PrefGetAppPreferences . . . . .	416
PrefGetPreference . . . . .	418
PrefSetAppPreferences . . . . .	418
PrefSetPreference. . . . .	420
<b>34 Sync Manager</b>	<b>421</b>
Sync Manager Constants. . . . .	421
Sync Manager Error Codes . . . . .	421
Sync Manager Security Policies . . . . .	422
Miscellaneous Sync Manager Constants . . . . .	422
Sync Manager Functions and Macros . . . . .	423
SyncAddSynchronizer . . . . .	423
SyncSessionGetAccess . . . . .	424
SyncSessionReleaseAccess. . . . .	424
<b>35 System Manager</b>	<b>427</b>
System Manager Constants . . . . .	427
Power Manager Error Codes . . . . .	427
System Features . . . . .	428
Processor Types . . . . .	432
Processor Masks . . . . .	433
Build Stages . . . . .	433

---

ROM Tokens . . . . .	434
Device Manufacturers . . . . .	434
Miscellaneous System Manager Constants . . . . .	435
System Manager Functions and Macros . . . . .	436
SysBatteryInfo . . . . .	436
sysFtrNumProcessorsIs68K . . . . .	437
sysFtrNumProcessorsIsARM . . . . .	438
SysGetROMToken . . . . .	438
sysGetROMVerBuild . . . . .	439
sysGetROMVerFix . . . . .	439
sysGetROMVerMajor . . . . .	440
sysGetROMVerMinor . . . . .	440
sysGetROMVerStage . . . . .	441
SysHandleEvent . . . . .	441
SysLCDBrightness . . . . .	442
SysLCDContrast . . . . .	442
sysMakeROMVersion . . . . .	443
SysRequestSleep . . . . .	443
SysSetAutoOffTime . . . . .	444
SysSleep . . . . .	444
SysTaskDelay . . . . .	444
SysTicksPerSecond . . . . .	445
SysTimeInCentiSecs . . . . .	445
SysTimeInMicroSecs . . . . .	446
SysTimeInMilliSecs . . . . .	446
SysTimeInMins . . . . .	446
SysTimeInSecs . . . . .	447
SysTimeToMicroSecs . . . . .	447
SysTimeToMilliSecs . . . . .	448
SysTimeToSecs . . . . .	448
SysTurnDeviceOn . . . . .	448
SysUIBusy . . . . .	449
<b>36 SysThread</b>	<b>451</b>
SysThread Structures and Types . . . . .	451

---

SysConditionVariableType . . . . .	451
SysCriticalSectionType . . . . .	452
SysThreadExitCallbackID . . . . .	452
SysThreadGroupHandle . . . . .	452
SysThreadGroupTag . . . . .	453
SysThreadGroupType . . . . .	453
SysTSDSlotID . . . . .	453
SysThread Constants . . . . .	454
Thread Priorities . . . . .	454
Miscellaneous System Thread Constants . . . . .	455
Predefined TSD Slot Names . . . . .	456
Predefined Semaphore Counts . . . . .	456
timeoutFlags_t . . . . .	457
SysThread Functions and Macros . . . . .	458
SysAtomicAdd32. . . . .	458
SysAtomicAnd32. . . . .	458
SysAtomicCompareAndSwap32 . . . . .	459
SysAtomicOr32 . . . . .	460
SysConditionVariableBroadcast . . . . .	460
SysConditionVariableClose . . . . .	461
SysConditionVariableOpen . . . . .	461
SysConditionVariableWait. . . . .	462
SysCriticalSectionEnter . . . . .	463
SysCriticalSectionExit. . . . .	463
SysCurrentThread . . . . .	464
SysGetRunTime . . . . .	464
SysSemaphoreCreate . . . . .	464
SysSemaphoreCreateEZ. . . . .	465
SysSemaphoreDestroy . . . . .	466
SysSemaphoreSignal . . . . .	466
SysSemaphoreSignalCount . . . . .	467
SysSemaphoreWait . . . . .	468
SysSemaphoreWaitCount . . . . .	469
SysThreadChangePriority . . . . .	470
SysThreadCreate . . . . .	471

---

SysThreadCreateEZ . . . . .	472
SysThreadDelay . . . . .	473
SysThreadExit . . . . .	474
SysThreadGroupCreate . . . . .	475
SysThreadGroupDestroy . . . . .	475
SysThreadGroupWait . . . . .	476
SysThreadInstallExitCallback . . . . .	476
SysThreadRemoveExitCallback . . . . .	477
SysThreadResume . . . . .	478
SysThreadStart. . . . .	479
SysThreadSuspend . . . . .	479
SysTSDAllocate . . . . .	481
SysTSDFree . . . . .	482
SysTSDGet . . . . .	482
SysTSDSet. . . . .	482
Application-Defined Functions . . . . .	483
SysThreadEnterFunc . . . . .	483
SysThreadExitCallbackFunc . . . . .	483
SysTSDDestructorFunc . . . . .	484

**37 System Utilities 485**

System Utilities Structures and Types . . . . .	486
CmpFuncPtr . . . . .	486
SearchFuncPtr . . . . .	486
SysBatteryKind . . . . .	486
SysDBListItemType . . . . .	486
System Utilities Constants . . . . .	487
Miscellaneous System Utilities Constants . . . . .	487
System Utilities Functions and Macros. . . . .	488
Abs . . . . .	488
SysAreWeRunningUI . . . . .	488
SysAreWeUIThread . . . . .	488
SysBatteryInfoV40 . . . . .	489
SysBinarySearch . . . . .	490
SysCopyStringResource . . . . .	491

---

SysCopyStringResourceV50 . . . . .	492
SysCreateDataBaseList . . . . .	492
SysCreateDataBaseListV50 . . . . .	494
SysCreatePanelList . . . . .	495
SysErrString . . . . .	496
SysErrStringTextOnly . . . . .	497
SysFormPointerArrayToStrings . . . . .	498
SysGetOSVersionString . . . . .	498
SysGetROMTokenV40 . . . . .	499
SysInsertionSort . . . . .	499
SysQSort . . . . .	501
SysRandom . . . . .	502
SysStringByIndex . . . . .	502
SysStringByIndexV50 . . . . .	503
<b>38 Time Manager</b>	<b>505</b>
Time Manager Constants . . . . .	505
Time Manager Error Codes . . . . .	505
Time Manager Functions and Macros . . . . .	506
TimGetSeconds . . . . .	506
TimGetTicks . . . . .	506
TimInit . . . . .	506
TimSetSeconds . . . . .	507
<b>39 Trace Manager</b>	<b>509</b>
Trace Manager Functions and Macros . . . . .	509
TM . . . . .	509
TmOutputB . . . . .	510
TmOutputT . . . . .	511
TmOutputTL . . . . .	512
TmOutputVT . . . . .	513
TmOutputVTL . . . . .	514
TraceDefine . . . . .	514
TraceOutput . . . . .	515
<b>Index</b>	<b>517</b>

# About This Document

---

This book documents many of the more advanced features of Palm OS. The operating system capabilities explored in this book are used by many, but not all, Palm OS applications. These capabilities include:

- Attention Manager
- Alarms
- System features and preferences
- Expansion
- Threading
- Shared Libraries
- Dates and Times
- Floating Point
- Debugging
- Power Management

## The *Exploring Palm OS* Series

This book is a part of the *Exploring Palm OS* series. Together, the books in this series document and explain how to use the APIs exposed to third-party developers by the fully ARM-native versions of Palm OS, beginning with Palm OS Cobalt. Each of the books in the *Exploring Palm OS* series explains one aspect of the Palm operating system, and contains both conceptual and reference documentation for the pertinent technology.

---

**IMPORTANT:** The *Exploring Palm OS* series is intended for developers creating native applications for Palm OS Cobalt. If you are interested in developing applications that work through PACE and that also run on earlier Palm OS releases, read the latest versions of the *Palm OS Programmer's API Reference* and *Palm OS Programmer's Companion* instead.

---

## About This Document

### *Additional Resources*

---

As of this writing, the complete *Exploring Palm OS* series consists of the following titles:

- *Exploring Palm OS: Programming Basics*
- *Exploring Palm OS: Memory, Databases, and Files*
- *Exploring Palm OS: User Interface*
- *Exploring Palm OS: User Interface Guidelines* (coming soon)
- *Exploring Palm OS: System Management*
- *Exploring Palm OS: Text and Localization*
- *Exploring Palm OS: Input Services*
- *Exploring Palm OS: High-Level Communications*
- *Exploring Palm OS: Low-Level Communications*
- *Exploring Palm OS: Telephony and SMS*
- *Exploring Palm OS: Multimedia*
- *Exploring Palm OS: Security and Cryptography*
- *Exploring Palm OS: Creating a FEP* (coming soon)
- *Exploring Palm OS: Porting Applications to Palm OS Cobalt*
- *Exploring Palm OS: Palm OS File Formats* (coming soon)

## Additional Resources

- Documentation  
PalmSource publishes its latest versions of this and other documents for Palm OS developers at  
<http://www.palmos.com/dev/support/docs/>
- Training  
PalmSource and its partners host training classes for Palm OS developers. For topics and schedules, check  
<http://www.palmos.com/dev/training>
- Knowledge Base  
The Knowledge Base is a fast, web-based database of technical information. Search for frequently asked questions

(FAQs), sample code, white papers, and the development documentation at

<http://www.palmos.com/dev/support/kb/>

## **Changes to This Document**

This section describes the changes made in each version of this document.

### **3110-002**

Minor editorial corrections.

### **3110-001**

The first release of this document for Palm OS Cobalt, version 6.0.

## **About This Document**

*Changes to This Document*

---



# Part I

# Concepts

This part contains conceptual and “how to” information on various aspects of the Palm OS operating system: the mechanisms by which you get the user’s attention; determining the user’s preferred means of operating; working with dates, times, and floating point numbers; detecting features of the operating system; patching a shared library; and debugging techniques.

The conceptual material in this part is organized into the following chapters:

<a href="#"><u>Attentions and Alarms</u></a> . . . . .	3
<a href="#"><u>Features</u></a> . . . . .	31
<a href="#"><u>Preferences</u></a> . . . . .	37
<a href="#"><u>Sound</u></a> . . . . .	49
<a href="#"><u>Expansion</u></a> . . . . .	53
<a href="#"><u>Shared Libraries</u></a> . . . . .	71
<a href="#"><u>System Reset</u></a> . . . . .	83
<a href="#"><u>Threading</u></a> . . . . .	85
<a href="#"><u>Power Management</u></a> . . . . .	101
<a href="#"><u>The ROM Serial Number</u></a> . . . . .	105
<a href="#"><u>Time</u></a> . . . . .	107
<a href="#"><u>Floating Point</u></a> . . . . .	109
<a href="#"><u>Debugging Strategies</u></a> . . . . .	113



# Attentions and Alarms

---

In this chapter you learn how to get the user's attention and how to set real-time alarms that can be used to either perform some periodic activity or display a reminder to the user.

This chapter is divided into the following broad topics:

- [Getting the User's Attention](#) begins with an introduction to the Attention Manager. This is followed by a detailed description of the Attention Manager from a user's perspective. Finally, it details what developers need to do in order to use the Attention Manager in their applications.
- [Alarms](#) covers the Alarm Manager, which can notify your programs when a specified point in time is reached.

## Getting the User's Attention

Palm OS<sup>®</sup> contains a standard mechanism that manages competing demands for the user's attention by both applications and drivers. This mechanism is known as the Attention Manager.

### The Role of the Attention Manager

This section provides a brief introduction to the Attention Manager. It covers the relationship between the Attention, Alarm and Notification Managers, and then discusses when it is appropriate to make use of the Attention Manager.

The Attention Manager provides a standard mechanism by which applications can tell the user that something of significance has occurred. It is designed to support communications devices which can receive data without explicit user interaction. The Attention Manager is responsible only for interacting with the user; it is not responsible for generating those events. In particular, the Alarm

## Attentions and Alarms

### *Getting the User's Attention*

---

Manager can be used in conjunction with the Attention Manager to inform the user that a particular point in time has been reached.

By maintaining a single list of all “alarm-like” things, the Attention Manager also improves the user’s experience when returning to the handheld after being gone for a while: he doesn’t have to click through a series of old alarm dialogs. Often the user doesn’t care about most of the missed appointments—although he might care about a few of them. Without the Attention Manager, the user cannot selectively dismiss or follow up on alarms.

Applications have complete control over the types of attention they can ask for. They can query the handheld for the set of special effects available—possibly including sound, vibration, and an LED—and then act on that set. The default option is to beep. All other options are either on or off; different vibrating patterns or multicolored LEDs are currently not supported. Note that the set of special effects is extensible; manufacturers may choose to add other means to get the user’s attention beyond the anticipated LED and vibration.

The Datebook, SMS, and Clock applications use the Attention Manager. Refer to the Datebook application’s source code for real-world examples of how you might use the Attention and Alarm Managers.

### **Attentions, Alarms and Notifications**

The Attention, Alarm, and Notification Managers are distinct subsystems that are often used in combination.

- The Attention Manager is designed solely to interact with the user when an event must be brought to the user’s attention.
- The Alarm Manager simply sends an event to an application when a particular point in time is reached. The application can then use the Attention Manager or some other mechanism to bring the alarm to the user’s attention, if appropriate.
- The Notification Manager informs those applications that have registered their interest whenever certain system-level or application-level events occur. If the user is to be informed of the event, the executable can use the Attention Manager.

### **When the Attention Manager Isn't Appropriate**

The Attention Manager is only designed for attempts to get attention that can be effectively suspended. It is not suitable for anything requiring an immediate response, such as a request to connect to another user or the “put away” dialog that is used during beaming. The Attention Manager also doesn't attempt to replace error messages. Applications must use modal dialogs and other existing OS facilities to handle these cases.

The Attention Manager is also not intended to replace the ToDo application, or to act as a universal inbox. Applications must make it clear that an item appearing in the Attention Manager is simply a reminder, and that dismissing it does not delete the item itself. That is, saying “OK” to an alarm does not delete the appointment, and dismissing an SMS reminder does not delete the SMS message from the SMS inbox.

### **Attention Manager Operation**

This section provides a detailed introduction to the Attention Manager from a user's point of view, introducing some of the terminology used throughout the rest of the chapter and pointing out operational subtleties that you should be aware of when developing applications that use the Attention Manager.

Attention-getting attempts can either be **insistent** or **subtle**. They differ only in the lengths to which each goes to get your attention. Insistent attempts get “in your face” by popping up a slip window (or “slip”) and triggering other visible and audible special effects in an effort to bring important events to your attention. A meeting reminder or incoming high-priority email message might warrant interrupting your work in this fashion. Subtle attentions substitute a small on-screen **attention indicator** for the slip, allowing you to be made aware of less-critical events without interrupting your current work flow. Although they can also trigger various special effects, subtle attentions don't typically do so. Examples of subtle events might include a reminder of an upcoming birthday or holiday, or an incoming SMS message.

## Attentions and Alarms

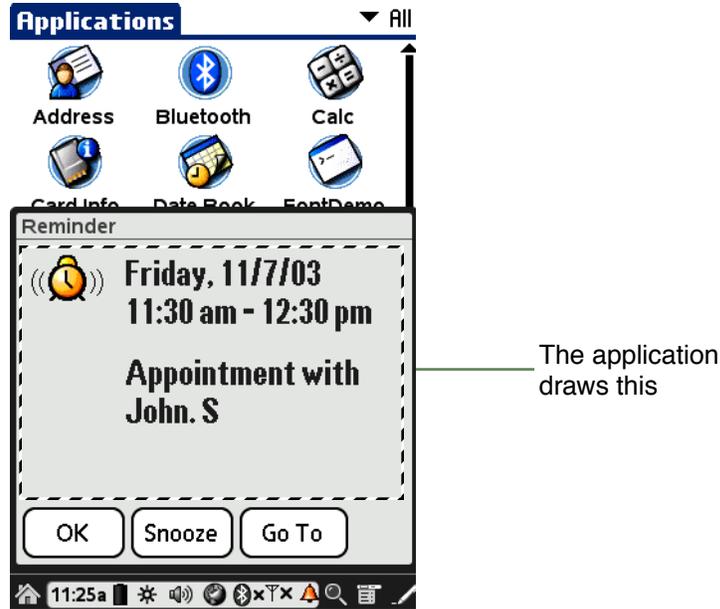
### Getting the User's Attention

---

#### Insistent Attentions

When an application makes an insistent attempt to get the user's attention, the **detail slip** opens:

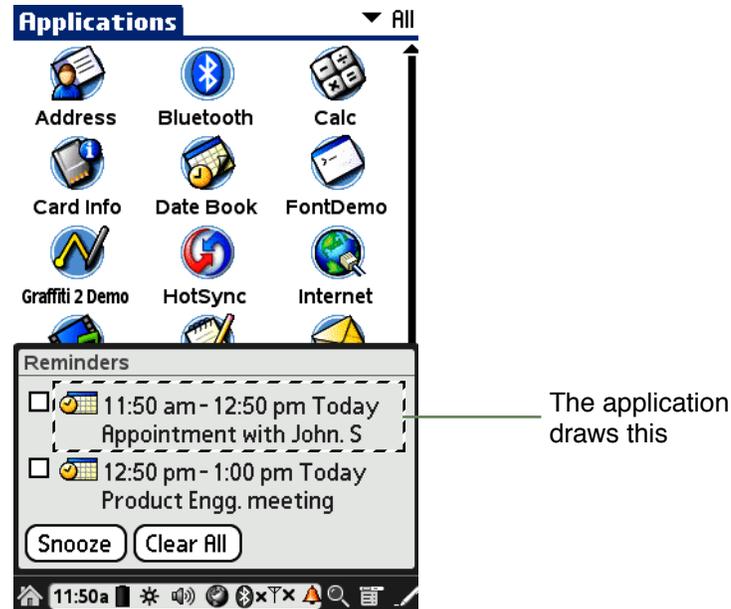
**Figure 1.1** Detail slip



The Attention Manager draws the title and the buttons. The application is responsible for drawing the rest. Most applications draw text and an appropriate icon, as shown in [Figure 1.1](#).

When a second application attempts to get attention, or when the first application makes a second attempt, and the first has not yet been dismissed or snoozed, the window changes to the **list slip**, presenting a list of things that require the user's attention:

**Figure 1.2 List slip**



In this slip, the Attention Manager draws the title and the buttons, and manages the list of items including the checkbox in the left-hand column. Items are listed in order of occurrence, with the newest at the top. The application is responsible for drawing some part of each line, giving it some flexibility over what to display. Applications have space to draw an icon and two lines of text in the standard font on the right-hand side of the list area.

In the detail slip the OK button dismisses the item. In the list slip, tapping the checkbox to the left of the item dismisses it. The Clear All button can be used to dismiss all items in the list view. Dismissing an item removes it from the list or closes the detail slip. Note that although it is gone from the Attention Manager, the item itself remains in the application.

In either slip, the "Snooze" button temporarily dismisses the Attention Manager slip. The attention indicator remains visible and the user can redisplay the slip at any time. After an interval of five minutes, if any attempts to get attention are still pending the Attention Manager redisplay the slip. Snooze does not remove attempts to get attention.

There is just one "Snooze" timer, and the snooze operation applies to the Attention Manager as a whole. This can lead to seemingly

## Attentions and Alarms

### *Getting the User's Attention*

---

odd behavior when new attention items are coming in while there is currently a snooze in progress. This situation should be rare, however.

To “go to” an individual item, tap the text or icon of the item in the list slip or tap the “Go To” button in the detail slip. This temporarily dismisses the Attention Manager and launches the appropriate application to display details about the item. For an alarm, this could take you to the Datebook view for the current day, with the current meeting scrolled into view. A successful “go to” also removes the attention item from the list.

### **Subtle Attentions**

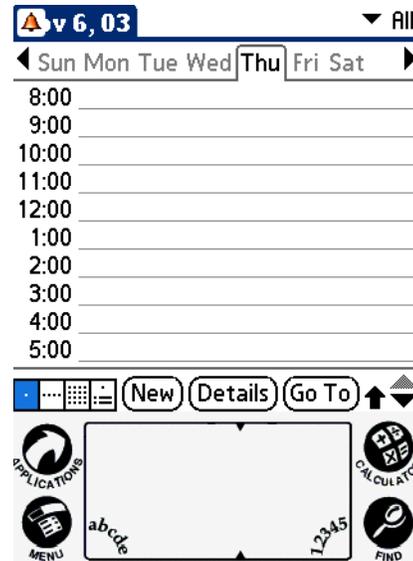
When an application makes a subtle attempt to get the users attention, no slip appears. Instead, on devices with no dedicated hardware, if the status bar is visible the attention indicator appears in the status bar as shown in [Figure 1.3](#).

**Figure 1.3 Attention indicator on the status bar**



If the status bar is not visible, the attention indicator appears, blinking, in the top left corner of the screen, as shown in [Figure 1.4](#).

**Figure 1.4 Attention indicator when status bar is not visible**



When the list contains one or more items, all of which have been seen by the user, the indicator (the bell icon in the status bar, or the blinking bell in the top left corner of the screen if the status bar is not visible) remains until the list is empty.

Tapping this indicator opens the Attention Manager in the list mode, even if there is only one item. Tapping to the right of the indicator, or tapping in the indicator's area when there are no pending attention attempts opens the menu bar as expected.

When the status bar is not visible, the attention indicator only functions with applications which use a standard form title object. In this case, however, the indicator doesn't appear when any of the following are true:

- the current application uses a custom title.
- the current application draws in the title area.
- the current form uses the Dialog title style.
- the current application's form title is too narrow to include the attention indicator.

Note that the attention indicator doesn't appear at all when there are no items in the Attention Manager's queue, whether or not the status bar is visible.

## Attentions and Alarms

### *Getting the User's Attention*

---

#### **Special Effects**

When a new attention item is added, the Attention Manager performs some combination of special effects, which can include playing sounds, flashing a LED, and triggering vibration. The exact combination of effects depends on user settings and on the application itself.

The Attention Manager attempts to open the slip before performing any special effects so you know immediately why it is trying to get your attention. However, it may not be possible to open the Attention Manager slip. If this is the case, the Attention Manager performs the special effects as soon as possible. It's better for the user to be made aware that something is happening, even if the handheld cannot say exactly what it is.

System-wide user preferences control the special effects: the volume at which to play alarms, whether or not to flash the LED (if any), whether or not to vibrate (if equipped). Applications can override these system-wide settings in either a positive or a negative way. For instance, an application could always blink the LED, even if the user said not to, or never blink the LED, even if the user desires it in general.

#### **Nagging**

If you don't explicitly dismiss or snooze an attention item it continues to "nag" you at predefined intervals, using the item's specified special effects. Applications control how frequently the user should be reminded, and how many times before the Attention Manager gives up.

When there are multiple attention items competing for nagging, the Attention Manager respects the nag settings for the most recent insistent item, or if there are none then for the most recent subtle item. Each special effect is handled separately; if one reminder wants sound but no vibration, and another wants vibration but no sound, the combination results in the sound from the first one and the vibration from the second one.

## **Attention Manager and Applications That Display Their Own Alarms**

The Attention Manager makes no attempt to override application behavior. If an application puts up its own dialog to get the user's attention, the Attention Manager doesn't get involved. Applications must be specifically written to use the Attention Manager in order to take advantage of its features and seamless integration with the Palm™ user experience.

## **Getting the User's Attention**

This section shows how your applications request the user's attention through the Attention Manager.

Getting the user's attention is simply a matter of calling [AttnGetAttention\(\)](#) with the appropriate parameters and then handling various launch codes sent by the Attention Manager. These launch codes allow your application to control what is displayed in the Attention Manager slips, to play sounds or perform other special effects, and to do any necessary processing when the user takes action on an existing attention item.

The `AttnGetAttention()` prototype looks like this:

---

```
status_t AttnGetAttention (DatabaseID dbID,  
uint32_t userData, AttnLevelType level, AttnFlagsType flags,  
uint16_t nagRateInSeconds, uint16_t nagRepeatLimit)
```

---

Specify the application requesting the user's attention in the `dbID` argument. You can use the [DmGetNextDatabaseByTypeCreator\(\)](#) function to obtain these values.

`userData` is used to distinguish a given attention attempt from others made by the same application; most applications pass the unique ID or other key for the record which caused the attention request. This value is passed to your code along with the launch code, and can be an integer, a pointer, or any other 32-bit value as needed by your application.

For the `level` argument, supply `kAttnLevelInsistent` or `kAttnLevelSubtle` depending on whether the given attention attempt is to be insistent or subtle.

## Attentions and Alarms

### *Getting the User's Attention*

---

Regardless of the level of the attention attempt, set the appropriate bits in the *flags* argument to cause sounds to play, LEDs to blink, or other physical effects to be performed. Depending on which flags you specify, the effect can always occur or can be suppressed by the user. For instance, to trigger a sound while honoring the user's preferences, you need only supply `kAttnFlagsSoundBit`. Or, to blink the LED but suppress any sounds, regardless of any preferences the user may have set, supply a value of `kAttnFlagsAlwaysLED | kAttnFlagsNoSound`. Finally, to choose only vibrate, do something like:

```
flags = kAttnFlagsNothing ^ kAttnFlagsNoVibrate
      | kAttnFlagsAlwaysVibrate;
```

While the above is somewhat complex, it does ensure that you override all defaults in the negative except vibration, which is overridden in the positive. See the definition of [AttnFlagsType](#) in the *Palm OS Programmer's API Reference* for a complete set of constants that can be used in combination for the *flags* argument.

---

**NOTE:** Applications may want to verify that the handheld is properly equipped to perform the desired effect. See "[Detecting Device Capabilities](#)" on page 23 for information on how to do this. If the handheld isn't properly equipped to handle a given special effect, the effect isn't performed. For example, if you set the `kAttnFlagsLEDBit` flag and the Palm Powered™ handheld doesn't have an LED, the attention attempt is processed as if the `kAttnFlagsLEDBit` had never been set.

---

Assuming that the handheld is capable of getting the user's attention using the requested special effect, the *nagRateInSeconds* and *nagRepeatLimit* arguments control how often and how many times the special effect is triggered in an attempt to get the user's attention. As the name implies, specify the amount of time, in seconds, the Attention Manager should wait between special effect triggers with *nagRateInSeconds*. Indicate the desired number of times the effect should be triggered using *nagRepeatLimit*. Applications typically supply a value of 300 for *nagRateInSeconds* and a value of 3 for *nagRepeatLimit*,

indicating that the special effect should be triggered three times after the initial attempt, at five minute intervals.

The following line of code shows how the SMS application calls the Attention Manager upon receiving a new SMS message. Note that a subtle attention is generated so that the user isn't interrupted every time an SMS message is received. Also note that the application doesn't override the user's settings when getting his attention. Finally, if the user doesn't respond to the first attention-getting attempt, the special effects are repeated three additional times in five-minute intervals.

---

```
err = AttnGetAttention(dbID, NULL, kAttnLevelSubtle,  
    kAttnFlagsUseUserSettings, 300, 3);
```

---

### **Attention Manager Commands**

In addition to calling [AttnGetAttention\(\)](#), your code must also respond to commands from the Attention Manager. The Attention Manager issues these commands by issuing a launch code. Among the possible commands are requests for your application to draw the contents of the detail and list slips, play application-specific sounds or perform other special effects, navigate to the item in your application that caused the attention item to be posted, and so forth.

#### ***Draw Detail and List Slips***

The Attention Manager's detail and list slips are drawn as a joint effort between the Attention Manager and applications requesting the user's attention. The shell of each slip, the title, and the buttons and checkboxes are drawn by the Attention Manager, while the remainder—text specific to each attention attempt, and frequently an accompanying icon—is drawn by the application itself. This gives each application full control over what sort of information to display.

## Attentions and Alarms

### Getting the User's Attention

Figure 1.5 Attention Manager slips



Although your application has full control over the display of its attention items, it may only draw in the designated area; active UI elements, such as scroll bars, custom buttons, or other widgets cannot be included. Users should be encouraged to launch the application with the “Go To” button and use the richer UI provided there. The clipping region is appropriately set so that your application cannot accidentally draw outside the designated area.

The `kAttnCommandDrawDetail` command indicates that your application is to draw the contents of the detail slip. Along with the command, the Attention Manager passes a structure of type [AttnCommandArgsDrawDetailTag](#). This structure contains the window-relative boundaries of the screen rectangle in which your application is to draw, a flag indicating whether this is the first time the user has seen this attention item, and a set of flags indicating which special effects will also be triggered.

The following code excerpt shows how a simple application might render the contents of the detail slip in response to [sysAppLaunchCmdAttention](#) launch code accompanied by a `kAttnCommandDrawDetail` command:

**Listing 1.1 Drawing the contents of the detail slip**

---

```
// Draw the icon
resH = DmGetResource(bitmapRsc, MyIconBitmap);
WinDrawBitmap(MemHandleLock(resH),
    paramsPtr->drawDetail.bounds.topLeft.x,
    paramsPtr->drawDetail.bounds.topLeft.y + 4);
MemHandleUnlock(resH);
DmReleaseResource(resH);

// Draw the text. The content of the string depends on the
// uniqueID that accompanies the kAttnCommandDrawDetail
// command
curFont = FntSetFont (largeBoldFont);
x = paramsPtr->drawDetail.bounds.topLeft.x + 37;
y = paramsPtr->drawDetail.bounds.topLeft.y + 4;
WinDrawChars(alertString, StrLen(alertString), x, y);
FntSetFont(curFont);
```

---

For a more complex, real-world example, see the `DrawDetailAlarm()` function in the Datebook application. In particular, note how that application adjusts the displayed text so that it all fits in the allocated space.

Note that because subtle attention items are only shown using the list view, your application doesn't need to respond to `kAttnCommandDrawDetail` if it doesn't produce insistent attention attempts.

The `kAttnCommandDrawList` command is similar; it indicates that your application is to draw a portion of the contents of the list slip. Along with this command, the Attention Manager passes a structure of type [AttnCommandArgsDrawListTag](#). Before sending your application the `kAttnCommandDrawList` command, the Attention Manager sets the background, foreground, and text colors as follows, depending on whether or not the item is selected:

---

<b>Affected Color</b>	<b>Not Selected</b>	<b>Selected</b>
Background Color	<code>UIFieldBackground</code>	<code>UIObjectSelectedFill</code>
Foreground Color	<code>UIObjectForeground</code>	<code>UIObjectSelectedForeground</code>
Text Color	<code>UIObjectForeground</code>	<code>UIObjectSelectedForeground</code>

---

## Attentions and Alarms

### *Getting the User's Attention*

---

The code to draw an attention item in the list slip is very similar to the code you use to draw the same item in the detail slip. Here is an excerpt:

#### **Listing 1.2 Drawing the contents of the list slip**

---

```
// Draw the icon.
resH = DmGetResource(bitmapRsc, MySmallIconBitmap);
iconP = (BitmapPtr)(MemHandleLock(resH));
// center it in the space allotted
iconOffset = (kAttnListMaxIconWidth - iconP->width)/2;
x = paramsPtr->drawList.bounds.topLeft.x;
y = paramsPtr->drawList.bounds.topLeft.y;
WinDrawBitmap(iconP, x + iconOffset, y);
MemHandleUnlock(resH);
DmReleaseResource(resH);

// Draw the text
curFont = FntSetFont(stdFont);
WinDrawChars(alertString, StrLen(alertString),
             x + kAttnListTextOffset, y);
FntSetFont(curFont);
```

---

The primary differences arise from the fact that the list slip provides a smaller, more structured area in which to draw. Also, the area in which you draw will always have been erased before you are asked to draw in it, so you don't ever have to erase it beforehand.

The icon should be no wider than `kAttnListMaxIconWidth` pixels, and should be centered within this width if it is smaller than `kAttnListMaxIconWidth`. The text should honor a left-hand margin of `kAttnListTextOffset` pixels. In the above example, the alert string is assumed to fit within the available space; see the `DrawListAlarm()` function in the Datebook application for an example of how that application adjusts the displayed text in the event that it doesn't all fit in the allocated space.

---

**NOTE:** Applications may, in certain rare circumstances, receive a `kAttnCommandDrawDetail` or `kAttnCommandDrawList` command for an item which is no longer valid. Respond by either calling [AttnForgetIt](#), by drawing nothing, or by drawing an error message.

---

### ***Play Sound or Perform a Custom Effect***

Most applications play a sound when attempting to get the user's attention. If the `kAttnFlagsAlwaysSound` is set when your application calls [AttnGetAttention](#), the Attention Manager sends a `kAttnCommandPlaySound` command to your application when attempting to get the user's attention. Your application should simply play the appropriate sound in response to this command. Both the Datebook and SMS applications play sounds based upon the user's preferences when getting the user's attention; see the Datebook application's source code for an example of how to respond to the `kAttnCommandPlaySound` command.

Because the Attention Manager can potentially play a sound, blink an LED, and vibrate in addition to displaying a slip or presenting the attention indicator, most applications don't request that some other application-specific custom effect be performed. If your application needs to do something different, you can specify `kAttnFlagsAlwaysCustomEffect` when calling `AttnGetAttention()`. This causes a `kAttnCommandCustomEffect` command to be sent to your application, at which time it should perform the desired effect. If your application is like most, however, it won't ever receive a `kAttnCommandCustomEffect` command, so you needn't worry about responding to it.

Neither `kAttnCommandPlaySound` nor `kAttnCommandCustomEffect` are accompanied by any kind of data structure indicating the sound or effect to be performed. If your application doesn't hard-wire this information, you may want to store it in the application's preferences database.

### ***Go There***

When the user taps on the "Go To" button in the detail view or on the item text or icon (not the checkbox) in the list view, your application receives a `kAttnCommandGoThere` command. It then needs to switch to your application and display the information relating to the chosen attention item. The `kAttnCommandGoThere` command is similar to the [sysAppLaunchCmdGoTo](#) launch code, but you don't have globals when your application receives the `kAttnCommandGoThere` command and your application is called using [SysAppLaunch\(\)](#), rather than [SysUIAppSwitch\(\)](#).

## Attentions and Alarms

### *Getting the User's Attention*

---

Because of this, most applications perform a `SysUIAppSwitch()` upon receiving `kAttnCommandGoThere`.

Note that your application should verify that the data that triggered the attention attempt is still relevant. In the Datebook, for instance, the user could:

1. Be alerted to an appointment by the Attention Manager.
2. Tap "Snooze."
3. Press the Datebook button and delete the appointment that was just brought to the user's attention.
4. Tap the attention indicator.
5. Tap the attention item corresponding to the now-deleted appointment.

In this scenario, the Datebook application could theoretically receive a `kAttnCommandGoThere` command along with a unique ID referencing a Datebook record that has been deleted. Whenever the Datebook application receives `kAttnCommandGoThere` along with a unique ID for a deleted Datebook database record, it calls [`AttnForgetIt`](#) for the now defunct attention item and then returns.

In reality, the Datebook calls `AttnForgetIt()` whenever the user deletes an alarm, and whenever an alarm is determined to be no longer valid. It can do this without even checking to see if the alarm is among those that the Attention Manager is currently tracking; if you pass a combination of database ID and unique ID to `AttnForgetIt()` that doesn't correspond to an item in the Attention Manager's queue, `AttnForgetIt()` does nothing and returns a value of `false`.

Most applications will choose to call `AttnForgetIt()` once the user has viewed the data corresponding to the attention item. This is how the SMS application operates: incoming messages are brought to the user's attention via the Attention Manager, and are removed from the Attention Manager's queue once they've been read.

### **Got It**

When the user dismisses a particular attention item, a `kAttnCommandGotIt` command is sent to the application. Along with this command is a boolean that indicates if the item was

explicitly dismissed by the user, since `kAttnCommandGotIt` is also issued as the result of a successful [AttnForgetIt](#) call. Upon receipt of this command, you may want to clean up memory, delete an alarm, or do other application-specific processing.

### ***Iterate***

When something happens that may potentially cause an application's attention items to become invalid, that application should call [AttnIterate\(\)](#). `AttnIterate()` generates a series of `kAttnCommandIterate` commands, one for each of the application's pending attention items. Thus, when your application receives a `kAttnCommandIterate` command it should validate the indicated attention item and take appropriate action if the item is no longer valid—up to and including calling [AttnForgetIt](#) for the item.

The SMS application does not respond to `kAttnCommandIterate`. The Datebook does; this command is generated whenever the user updates his preferences. Many applications call `AttnIterate()` after receiving a [sysAppLaunchCmdSyncNotify](#) launch code so that they can update (with [AttnUpdate](#)) or remove (with `AttnForgetIt()`) items which were affected by the HotSync® operation.

Note that you can safely call `AttnForgetIt()` from within the iteration since `AttnForgetIt()` only marks the record for deletion and thus doesn't confuse the iteration.

### ***Snooze***

Most applications—including the Datebook and SMS applications—ignore `kAttnCommandSnooze`, which indicates that the user has tapped the Snooze button. Beyond the unique ID that identifies the attention item, nothing accompanies the Snooze command. `kAttnCommandSnooze` is passed to each and every item currently pending, insistent or subtle. This means that applications with more than one attention item pending receive this command more than once.

### **Triggering Special Effects**

The Attention Manager activates any requested special effects for each attention item, but your application might want to activate

## Attentions and Alarms

### *Getting the User's Attention*

---

those special effects without also posting an attention item to the queue. You can do this through a call to [AttnDoSpecialEffects\(\)](#). Supply the appropriate combination of flags to trigger the desired effects. See [AttnFlagsType](#) for a complete list of flags.

## Attentions and Alarms

The Attention Manager is often used in conjunction with the Alarm Manager to get the user's attention at a particular time. The basic use of the Alarm Manager is covered in "[Alarms](#)" on page 24, but because the Attention Manager handles UI synchronization, do the following when using the Alarm Manager with the Attention Manager:

- Call [AttnGetAttention](#) when your application receives the [sysAppLaunchCmdAlarmTriggered](#) launch code.
- In your [sysAppLaunchCmdAlarmTriggered](#) handling code, set the `purgeAlarm` field in the launch code's parameter block to `true` before returning. This removes the alarm from the queue, so your application won't receive the [sysAppLaunchCmdDisplayAlarm](#) launch code.

Don't wait until the [sysAppLaunchCmdDisplayAlarm](#) launch code is received to call [AttnGetAttention\(\)](#).

## Detecting and Updating Pending Attentions

Once your application has requested that the Attention Manager get the user's attention, it may later need to update that attention request. For instance, if your application uses a single attention item to indicate that a number of unread messages have been received, it should update that item as additional items are received and read. The Attention Manager provides a handful of functions that allow applications to examine the Attention Manager's queue and update the items within that queue.

The [AttnGetCounts\(\)](#) function allows you to determine how many items are currently competing for the user's attention. It always returns the total number of items, but can return the number of subtle and/or insistent items as well. It can also return the counts

for all applications. For instance, the following would set `numItems` to the total number of pending attention items from all sources:

```
numItems = AttnGetCounts(0, 0, NULL, NULL);
```

Or, to get the number of subtle and insistent attention items in addition to the total requested by a single application, use something like:

---

```
numItems = AttnGetCounts(dbID, &insistentItems,  
    &subtleItems);
```

---

To verify each pending attention item for a given application, use the Attention Manager's [AttnIterate\(\)](#) function as described under "[Iterate](#)" on page 19.

After a `HotSync` is a popular time to invoke `AttnIterate()`; the `HotSync` operation may have altered the application's underlying data in such a way as to render pending attention items obsolete or invalid.

### Deleting Pending Attention Items

In many cases you'll need to delete an attention item. For instance, a `HotSync` may alter the underlying application data in such a way so as to invalidate the attention attempt. Or, the user might switch to your application and manually update the data in a similar way, for example by deleting an appointment for which there is a pending alarm. The [AttnForgetIt](#) function exists for this purpose. Simply invoke this function and supply the database ID and user data that uniquely identify the attention attempt. You don't even have to verify that the attention attempt is still in the Attention Manager's queue: `AttnForgetIt()` doesn't complain if the attention attempt doesn't exist, merely returning a value of `false` to indicate this condition.

### Updating Pending Attention Items

To update an existing attention item, use [AttnUpdate](#). This function is very similar to [AttnGetAttention](#), though instead of actual values you pass pointers to those values you want to update. Supply `NULL` pointers for `flagsP`, `nagRateInSecondsP`, and/or `nagRepeatLimitP` to leave them untouched. For instance, to

## Attentions and Alarms

### Getting the User's Attention

---

change the flags that control the special effects used to get the user's attention, do the following:

#### Listing 1.3 Updating an existing attention item

---

```
// This assumes that dbID and myUserData are
// declared and set elsewhere to values that identify the
// attention item we're trying to update
Boolean updated;
AttnFlagsType newFlags;

// set newFlags appropriately
updated = AttnUpdate(dbID, myUserData, NULL, &newFlags, NULL,
    NULL);
if (updated){
    // update succeeded
} else {
    // update failed - attention item may no longer exist
}
```

---

**NOTE:** Although `AttnUpdate()` may cause a given attention item to redraw, it does not rerun the special effects (if any) that occurred when that attention item was added. If you want to trigger Attention Manager effects for a particular item, call [AttnForgetIt](#) followed by [AttnGetAttention](#).

---

While updating the attention item, if the handheld is on and the Attention Manager slip is currently showing then `AttnUpdate()` forces the item to be redrawn. This in turn calls back to the client application so that it can update its portion of the slip.

`AttnUpdate()` causes the specified item to be redrawn if it is visible, regardless of the *flags*, *nagRateInSeconds*, and *nagRepeatLimit* parameters. Thus, `AttnUpdate()` isn't limited to updating one or more aspects of an attention item; it also allows an application to update the text of an attention attempt without having to destroy and then rebuild the Attention Manager slip.

Note that if the handheld is off, the update is delayed until the handheld is next turned on; `AttnUpdate()` doesn't itself turn the screen on.

## Detecting Device Capabilities

Although you can blindly request that a given special effect, such as vibration, be used to get the user's attention without checking to see if that special effect is supported on the Palm Powered handheld, you may want your application to behave differently in the absence of a particular device feature. The Attention Manager defines a feature that you use with the `FtrGet()` function to determine the handheld's physical capabilities. You can also use this feature to determine the user's attention-getting preferences. For example:

### Listing 1.4 Checking for vibrate capability

---

```
// See if the device supports vibration
FtrGet(kAttnFtrCreator, kAttnFtrCapabilities, &capabilities);
if (capabilities & kAttnFlagsHasVibrate){
    // Vibrate-specific processing goes here
    ...
}
```

---

See "[Attention Manager Constants](#)" on page 138 for descriptions of all of the relevant flags.

## Controlling the Attention Indicator

For the most part, applications can ignore the attention indicator, which consumes a small portion of a form's title bar if the status bar is not visible. If the status bar is visible, or if the currently-displayed form doesn't have a title bar, or if the form is modal, the attention indicator isn't drawn in the form's title bar. If an application takes over the entire screen or does something special with the form's title bar, it should explicitly disable the attention indicator while the form is displayed.

As an example, the Datebook application disables the attention indicator when:

- a note associated with a Datebook entry is displayed.
- an entry's description is being displayed in the week view.
- the time is being displayed in the title bar in place of the date.

## Attentions and Alarms

### Alarms

---

To disable the attention indicator that is displayed in the upper left corner of the screen when the status bar is not visible, simply call [`AttnIndicatorEnable\(\)`](#) and supply a value of `false` for the `enableIt` argument. To re-enable it, call [`AttnIndicatorEnable\(\)`](#) again, this time supplying an argument value of `true`.

If your application disables the attention indicator, it may want to provide some means for the user to open the Attention Manager's slip in list mode. The [`AttnListOpen\(\)`](#) function can be used to do this.

## Alarms

The Palm OS Alarm Manager provides support for setting real-time alarms, for performing some periodic activity, or for displaying a reminder. The Alarm Manager:

- Works closely with the Time Manager to handle real-time alarms.
- Sends launch codes to applications that set a specific alarm to inform the application the alarm is due.
- Allows only one alarm to be set per application.
- Handles alarms by application in a two cycle operation:
  - First, it notifies each application that the alarm has occurred. The application verifies that the alarm is still valid at this point. Applications that don't use the Attention Manager typically play a sound here.
  - Second, after all pending alarms have received their first notification, the Alarm Manager sends another notification to each application, allowing it to display some UI.

The Alarm Manager doesn't have any UI of its own; it doesn't provide reminder dialog boxes, and it doesn't play the alarm sound. Applications that need to bring an alarm to the user's attention must do this themselves. The Attention Manager is designed specifically to handle this interaction with the user; see "[Attentions and Alarms](#)" on page 20 for tips on doing this.

---

**IMPORTANT:** When the handheld is in sleep mode, alarms can occur almost a minute late.

---

## Setting an Alarm

The most common use of the Alarm Manager is to set a real-time alarm within an application. Often, you set this type of alarm because you want to inform the user of an event. For example, the Datebook application sets alarms to inform users of their appointments.

Implementing such an alarm is a two step process. First, use the function [AlmSetAlarm\(\)](#) to set the alarm. Specify when the alarm should trigger and which application should be informed at that time.

[Listing 1.5](#) shows how the Datebook application sets an alarm.

### Listing 1.5 Setting an alarm

```
static void SetTimeOfNextAlarm (uint32_t alarmTime,
uint32_t ref) {
    LocalID dbID;
    DmSearchStateType searchInfo;

    DmGetNextDatabaseByTypeCreator (true, &searchInfo,
        sysFileTApplication, sysFileCDatebook, true, &dbID);

    AlmSetAlarm (dbID, ref, alarmTime, true);
}
```

---

Second, have your [PilotMain\(\)](#) function respond to the [sysAppLaunchCmdAlarmTriggered](#) and [sysAppLaunchCmdDisplayAlarm](#) launch codes.

When an alarm is triggered, the Alarm Manager notifies each application that set an alarm for that time via the [sysAppLaunchCmdAlarmTriggered](#) launch code. After each application has processed this launch code, the Alarm Manager sends each application [sysAppLaunchCmdDisplayAlarm](#) so that the application can display the alarm. The section "[Alarm Scenario](#)" gives more information about when these launch codes are received

## Attentions and Alarms

### Alarms

---

and what actions your application might take. For a specific example of responding to these launch codes, see the Datebook sample code.

It's important to note the following:

- An application can have only one alarm pending at a time. If you call `AlmSetAlarm()` and then call it again before the first alarm has triggered, the Alarm Manager replaces the first alarm with the second alarm. You can use the [AlmGetAlarm\(\)](#) function to find out if the application has any alarms pending.
- `AlmSetAlarm()` takes a `uint32_t` parameter that you can use to pass a specific value to your alarm handling code when the alarm triggers. (This is the *ref* parameter shown in [Listing 1.5](#).) The parameter blocks for both launch codes provide access to this reference parameter.
- The database ID that you pass to `AlmSetAlarm()` is the local ID of the *application* (the `.prc` file), not of the record database that the application accesses. You use a record database's local ID more frequently than you do the application's local ID, so this is a common mistake to make.
- In `AlmSetAlarm()`, the alarm time is given as the number of seconds since 1/1/1904. If you need to convert a conventional date and time value to the number of seconds since 1/1/1904, use [TimDateTimeToSeconds\(\)](#).

To clear a pending alarm, call `AlmSetAlarm()` and pass 0 for the alarm seconds parameter.

### Alarm Scenario

Here's how an application and the Alarm Manager typically interact when processing an alarm:

1. The application sets an alarm using [AlmSetAlarm\(\)](#).

The Alarm Manager adds the new alarm to its alarm queue. The alarm queue contains all alarm requests. Triggered alarms are queued up until the Alarm Manager can send the launch code to the application that created the alarm. However, if the alarm queue becomes full, the oldest entry that has been both triggered and notified is deleted to make room for a new alarm.

2. When the alarm time is reached, the Alarm Manager searches the alarm queue for the first application that set an alarm for this alarm time.
3. The Alarm Manager sends this application the [sysAppLaunchCmdAlarmTriggered](#) launch code.
4. The application can now:
  - Set the next alarm.
  - Play a short sound.
  - Perform some quick maintenance activity.

The application should not perform any lengthy tasks in response to `sysAppLaunchCmdAlarmTriggered` because doing so delays other applications from receiving alarms that are set to trigger at the same time.

If the application is using the Attention Manager to bring this alarm to the user's attention, call [AttnGetAttention\(\)](#) here and set the `purgeAlarm` field in the launch code's parameter block to `true` before returning.

If this alarm requires no further processing, the application should set the `purgeAlarm` field in the launch code's parameter block to `true` before returning. Doing so removes the alarm from the queue, which means it won't receive the [sysAppLaunchCmdDisplayAlarm](#) launch code.

5. The Alarm Manager finds in the alarm queue the next application that set an alarm and repeats steps [3](#) and [4](#).  
This process is repeated until no more applications are found with this alarm time.
6. The Alarm Manager then once again finds the first application in the alarm queue who set an alarm for this alarm time and sends this application the launch code [sysAppLaunchCmdDisplayAlarm](#). Note that alarms that had their `purgeAlarm` field set to `true` during the processing of `sysAppLaunchCmdAlarmTriggered`—including all alarms that are being brought to the user's attention through the Attention Manager—are no longer in the queue at this point.
7. The application can now display a dialog box or some other type of reminder, if appropriate. At this point it isn't restricted to working quickly; it can take whatever time it

## Attentions and Alarms

### *Summary of Attentions and Alarms*

---

needs since at this point all applications have been notified that the alarm was triggered.

8. The Alarm Manager then finds the next entry in the alarm queue that has the same alarm time as the one being triggered, and repeats steps 6 and 7.

This process is repeated until no more applications are found with this alarm time.

If a new alarm time is triggered while an older alarm is still being displayed, all applications with alarms scheduled for this second alarm time are sent the `sysAppLaunchCmdAlarmTriggered` launch code, but the display cycle for the second set of alarms is postponed until all earlier alarms have finished displaying.

If a second alarm goes off before the first has been dismissed, the alarm manager sends the `sysAppLaunchCmdAlarmTriggered` launch code for the second alarm but waits to send the `sysAppLaunchCmdDisplayAlarm` launch code until after the first alarm's dialog has been dismissed. For applications that put up dialogs, this typically means that only one dialog at a time will appear on the screen. The Alarm Manager doesn't return to the event loop between the issuing of launch codes, so when the first alarm's dialog has been dismissed, the second alarm's dialog is immediately displayed. The net result for the user is that each alarm dialog in turn must be dismissed before the handheld can be used.

## Summary of Attentions and Alarms

---

### Attention Manager Functions

<a href="#"><code>AttnDoSpecialEffects()</code></a>	<a href="#"><code>AttnIndicatorEnabled()</code></a>
<a href="#"><code>AttnForgetIt()</code></a>	<a href="#"><code>AttnIterate()</code></a>
<a href="#"><code>AttnGetAttention()</code></a>	<a href="#"><code>AttnListOpen()</code></a>
<a href="#"><code>AttnGetCounts()</code></a>	<a href="#"><code>AttnUpdate()</code></a>
<a href="#"><code>AttnIndicatorEnable()</code></a>	

---

### Alarm Manager Functions

<a href="#"><code>AlmSetAlarm()</code></a>	<a href="#"><code>AlmGetAlarm()</code></a>
--	--

---

## **Attentions and Alarms**

*Summary of Attentions and Alarms*

---

## **Attentions and Alarms**

*Summary of Attentions and Alarms*

---

# Features

---

A **feature** is a 32-bit value that has special meaning to both the feature publisher and to users of that feature. Features can be published by the system or by applications.

Each feature is identified by a feature creator and a feature number:

- The feature creator is a unique creator registered with PalmSource, Inc. You usually use the creator type of the application that publishes the feature.
- The feature number is any 32-bit value used to distinguish between different features of a particular creator.

Once a feature is published, it remains present until it is explicitly unregistered or the handheld is reset. A feature published by an application sticks around even after the application quits.

This section introduces the Feature Manager by discussing these topics:

- [The Operating System Version Feature](#)
- [Application-Defined Features](#)
- [Using the Feature Manager](#)
- [Feature Memory](#)

## The Operating System Version Feature

As an example of what features are used for, the version of the operating system is stored in a feature. This feature is published by the system and contains a 32-bit representation of the operating system version. The operating system version has a feature creator of `sysFtrCreator` and a feature number of

## Features

### *The Operating System Version Feature*

---

`sysFtrNumROMVersion`. To obtain the operating system version you call [FtrGet\(\)](#), like this:

---

```
err = FtrGet(sysFtrCreator, sysFtrNumROMVersion,
            &romVersion);
```

---

On Palm OS Cobalt, the value written to `romVersion` by this call is `0x06003000`. This indicates that the operating system version is 6.0, and that it is a release ROM.

If your code is dependant on a particular Palm OS version, you'll want to obtain the value of the `sysFtrNumROMVersion` feature and compare it to a known base value. Rather than hard wiring an obscure constant like one of the above into your code, however, you can use the `sysMakeROMVersion` macro (defined in `SystemMgr.h`) to construct a version number for comparison purposes. It takes five parameters:

- Major version number
- Minor version number
- Fix level
- Build stage (either `sysROMStageDevelopment`, `sysROMStageAlpha`, `sysROMStageBeta`, or `sysROMStageRelease`)
- Build number

The fix level and build number parameters are normally set to zero, while build stage is usually set to `sysROMStageRelease`. Simply check to see whether `sysFtrNumROMVersion` is greater than or equal to the version number constructed with `sysMakeROMVersion`, as shown here:

---

```
// See if we're on ROM version 6.0 or later.
FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
if (romVersion >= sysMakeROMVersion(6, 0, 0,
    sysROMStageRelease, 0)) {
    ....
}
```

---

Other system features are defined in `SystemMgr.h`; see [“System Features”](#) on page 428 for a complete list. Checking for the presence of system features allows an application to be compatible with

multiple versions of the system by refining its behavior depending on which capabilities are actually present in the device. For instance, the blitter's capabilities can be determined by checking the version number of the Window Manager. If the Window Manager version is 4 or greater, "high-density" displays (greater than 160 by 160 pixels) are supported. If the Window Manager version is 5 or greater, quarter-VGA displays (320 by 240 pixels) are supported. (Note, however, that just because the blitter is capable of working with a quarter-VGA display, for example, that doesn't mean that the handheld actually *has* a quarter-VGA display. For that, you need to check the attributes of the screen with [WinScreenGetAttribute\(.\)](#))

In some cases the presence or absence of a feature is also important. For instance, the VFS Manager may or may not be present on a given device. If it is not present, the [FtrGet\(.\)](#) call returns an error. See "[Checking for the Presence of the VFS Manager](#)" on page 70 for more on checking for the VFS Manager feature.

---

**IMPORTANT:** Not all features are guaranteed to be present on any given device. Always check for specific features rather than relying on the system version number to determine if a specific API set is available.

---

## Application-Defined Features

Applications may find the Feature Manager useful for their own private use. For example, an application may want to publish a feature that contains a pointer to some private data it needs for processing launch codes. Or, an application might use features to hold small pieces of data that must persist across application launches.

The feature manager maintains one feature table in RAM, and one feature table in ROM. Application-defined features are stored in the RAM feature table. System features are stored in the ROM feature table. Note, however, that the contents of the ROM feature table are copied into the RAM feature table at system startup.

## Features

### *Using the Feature Manager*

---

## Using the Feature Manager

To check whether a particular feature is present, call `FtrGet()` and pass it the feature creator ID and feature number. If the feature exists, `FtrGet()` returns the 32-bit value of the feature. If the feature doesn't exist, an error code is returned.

To publish a new feature or change the value of an existing one, call `FtrSet()` and pass the feature creator ID, feature number, and the 32-bit value of the feature. A published feature remains available until it is explicitly removed by a call to `FtrUnregister()` or until the system resets; simply quitting an application doesn't remove a feature published by that application. Upon reset, all features and feature pointers are cleared out and must be re-initialized by the appropriate component.

You can get a complete list of all published features by calling `FtrGetByIndex()` repeatedly. The first time you call `FtrGetByIndex()`, pass an index value of 0. Increment the index value by 1 on subsequent calls until `FtrGetByIndex()` returns an error. Note that although `FtrGetByIndex()` accepts a parameter that specifies whether to search the ROM feature table or the RAM feature table, the contents of the ROM table are copied into the RAM table at system startup. Thus, the RAM feature table contains all feature values.

## Feature Memory

Feature memory provides quick, efficient access to data that persists between invocations of an application. The values stored in feature memory persist until the handheld is reset or until you explicitly free the memory. Feature memory is memory allocated from the storage heap. Thus, you write to feature memory using `DmWrite()`, which means that writing to feature memory is no faster than writing to a database. However, feature memory can provide more efficient access to that data in certain circumstances.

To allocate a chunk of feature memory, call `FtrPtrNew()`, specifying a feature creator, a feature number, the number of bytes

to allocate, and a location where the Feature Manager can write a pointer to the newly allocated memory chunk. For example:

---

```
FtrPtrNew(appCreator, myFtrMemFtr, 32, &ftrMem);
```

---

Elsewhere in your application, you can obtain the pointer to the feature memory chunk using [FtrPtrGet\(\)](#).

Feature memory is considered a performance optimization. The conditions under which you'd use it are not common, and you probably won't find them in a typical application.

One potential use of feature memory is to “publish” data from your application or library to other applications when that data doesn't fit in a normal 32-bit feature value. For example, suppose you are writing a communications library and you want to publish an icon that client applications can use to draw the current connection state. The library can use [FtrPtrNew\(\)](#) to allocate a feature memory chunk and store an icon representing the current state in that location. Applications can then use [FtrPtrGet\(\)](#) to access the icon and display the connection state on the screen.

## Feature Memory Limitations

Feature pointer memory is allocated from the storage heap as part of a “features” temporary database, so applications must use [DmWrite\(\)](#) to write to feature pointers. (While a feature pointer's contents are modified via [DmWrite\(\)](#), you modify the pointer itself using [FtrPtrResize\(\)](#) or [FtrPtrFree\(\)](#).)

If one thread or process modifies the feature pointer while another one is using it, the result is undefined. This is generally not a problem because feature pointers are usually private and used by only one application at a time, so synchronization is never required. However, it is important to note that feature pointers are not a good foundation for sharing memory across threads or processes in Palm OS Cobalt, and you should not use them for that purpose.

## **Features**

*Feature Memory*

---

# Preferences

---

The Preferences Manager handles both system-wide preferences and application-specific preferences. The Preferences Manager maintains preferences in two separate databases:

- The “saved” preferences database contains preferences that are backed up during a HotSync operation. There is one “saved” preferences database that all applications use. This database contains all system-wide preferences as well as application-specific preferences.
- The “unsaved” preferences database contains application-specific preferences that are not to be backed up during a HotSync operation. There is one “unsaved” preferences database that all application use.

This section describes how to obtain and set values for each of these preferences databases. It covers:

- [Accessing System Preferences](#)
- [Setting System Preferences](#)
- [Setting Application-Specific Preferences](#)

## Accessing System Preferences

The system preferences specify how users want their Palm Powered™ handhelds to behave. For example, system preferences specify how dates and times are displayed and whether the system plays a sound when an alarm fires. These values are typically set using the built-in Preferences or Security application. Applications should, as a rule, respect the values stored in the system preferences.

To obtain the value of a system preference, use the [PrefGetPreference\(\)](#) function and pass one of the [SystemPreferencesChoice](#) enum constants. For example, if an application’s user interface displays the current date and time, it

## Preferences

### Setting System Preferences

---

could do the following to find out how the user wants the date and time displayed:

```
TimeFormatType timeFormat = (TimeFormatType)
    PrefGetPreference(prefTimeFormat);
DateFormatType dateFormat = (DateFormatType)
    PrefGetPreference(prefDateFormat);
```

Note that the `PrefGetPreference` function by default returns a `uint32_t` value. This return value must be cast to the appropriate type for the preference being returned.

Also note that the system preferences structure has been updated many times and maintains its own version information. Each Palm OS release that modifies the system preferences structure adds its new values to the end and increments the structure's version number. Palm OS Cobalt supports version 11 and earlier preferences; see the documentation on [SystemPreferencesChoice](#) for more information.

## Setting System Preferences

Occasionally, an application may need to set the value of a system-wide preference. It is strongly recommended that you not override the system preferences without user input.

For example, suppose you are writing a replacement for the built-in Address Book application. The Preferences application contains a panel where the user can remap the Address Book hard key to open any application they choose. However, you want to make it more convenient for your users to remap the Address Book button, so you might display an alert that asks first-time users if they want the button remapped. If they tap Yes, then you should call [PrefSetPreference\(\)](#) with the new value. The code might look like the following:

### Listing 3.1 Setting a system preference

---

```
if (PrefGetPreference(prefHard2CharAppCreator !=
    myAppCreatorId)) {
    if (FrmAlert(MakeMeTheDefaultAlert) == 0) {
        /* user pressed Yes */
```

```

        PrefSetPreference(prefHard2CharAppCreator,
            myAppCreatorId);
    }
}

```

---

### Preferences in the User Interface

[Table 3.1](#) shows the `SystemPreferencesChoice` constants and how they correspond to the values that users can set in the Preferences and Security applications. For further information about each preference, see the [SystemPreferencesChoice](#) enum documentation.

**Table 3.1 Preferences set in Preferences and Security apps**

Application/Panel	Field	SystemPreferencesChoice Constant
Preferences application General panel	Auto-off After	<code>prefAutoOffDuration,</code> <code>prefAutoOffDurationSecs</code>
	Stay on in Cradle	<code>prefStayOnWhenPluggedIn</code>
	System Sound	<code>prefSysSoundLevelV20,</code> <code>prefSysSoundVolume</code>
	Alarm Sound	<code>prefAlarmSoundLevelV20,</code> <code>prefAlarmSoundVolume</code>
	Alarm Vibrate <sup>1</sup>	<code>prefAttentionFlags</code>
	Alarm LED <sup>1</sup>	<code>prefAttentionFlags</code>
	Game Sound	<code>prefGameSoundLevelV20,</code> <code>prefGameSoundVolume</code>
	Beam Receive field	<code>prefBeamReceive</code>
Preferences application Date & Time panel	Set Time Zone field	<code>prefTimeZone</code>
	Daylight Saving	<code>prefDaylightSaving</code> <code>Adjustment</code>

## Preferences

### Setting System Preferences

---

**Table 3.1 Preferences set in Preferences and Security apps**

<b>Application/Panel</b>	<b>Field</b>	<b>SystemPreferencesChoice Constant</b>
Preferences application Formats panel	Preset to	prefCountry68K
	Time	prefTimeFormat
	Date	prefDateFormat, prefLongDateFormat
	Week starts	prefWeekStartDay
	Numbers	prefNumberFormat
Preferences application Buttons panel	Buttons on main panel	prefHard1CharAppCreator, prefHard2CharAppCreator, prefHard3CharAppCreator, prefHard4CharAppCreator, prefCalcCharAppCreator
	Pen button	prefRonamaticChar
	HotSync button	prefHardCradleCharApp Creator prefHardCradle2CharApp Creator
Security application	Auto Lock Handheld	prefAutoLockType, prefAutoLockTime, prefAutoLockTimeFlag
	Current Security	prefHidePrivateRecordsV33, prefShowPrivateRecords
	Lock & Turn Off... button	prefDeviceLocked

---

1. The Alarm Vibrate and Alarm LED preferences only appear on handhelds that have the appropriate hardware capabilities.

## Setting Application-Specific Preferences

You can use the Preferences Manager to set and retrieve preferences specific to your application. You do this by storing the preferences in one of two databases: the “saved” preferences database or the “unsaved” preferences database.

To write application preferences, you use [PrefSetAppPreferences\(\)](#). To read them back in, you use [PrefGetAppPreferences\(\)](#). Typically, you write the preferences in response to the `appStopEvent` when control is about to pass to another application. You read the preferences in response to a normal launch.

`PrefSetAppPreferences()` and `PrefGetAppPreferences()` take roughly the same parameters: the application creator ID, a preference ID that uniquely identifies this preference resource, a pointer to a structure that holds the preference values, the size of the preferences structure, and a Boolean that indicates whether the “saved” or the “unsaved” preferences database is to be used. `PrefSetAppPreferences()` also takes a version number for the preference structure. This value is the return value for `PrefGetAppPreferences()`.

The following sections discuss the issues involved in using application-specific preferences:

- [When to Use Application Preferences](#)
- [How to Store Preferences](#)
- [Which Preferences Database to Use](#)
- [Updating Preferences Upon a New Release](#)

### When to Use Application Preferences

You use application preferences to store state specific to your application that should persist across invocations of your application. For example, the built-in applications store information about the last form and the last record or records displayed before control is switched to another application. This way, the user can be returned to the same view when he or she goes back to that application.

## Preferences

### *Setting Application-Specific Preferences*

---

You can also use preferences for other values. You might allow the user to customize the way the application behaves and store such information in the preferences database. You might also use the preferences database as a way to share information with other applications.

Make sure that the preference values you choose are as concise as possible. In games, for example, it is often tempting to store a bitmap for the current state of the screen. Such a bitmap is over 25KB on a color handheld, and it is therefore best avoided. Instead, it is better to store items that let you recreate the current state, such as the player's position in pixels and the current level.

There are other ways to store values pertinent to your application. For example, you can store a single value as a feature using the [Feature Manager](#). Note, however, that preferences (including those stored in the "unsaved" database) survive a soft reset because they reside in the storage heap. Features are deleted upon a soft reset. For this reason, preferences are more appropriate for storing application state than feature are.

Instead of storing application state values as preferences or features, you could also use a database that your application creates and maintains itself. If you choose this method of storing application preference values, you must write your own functions to read the preferences from the database and write the preferences to the database. If you want the preferences backed up, you need to set the backup bit. However, there may be cases where using your own database has advantages. See "[Which Preferences Database to Use](#)" on page 43.

## How to Store Preferences

Most applications store a single preference structure under a single preference resource ID. When the application receives an `appStopEvent`, it writes the entire structure to the resource using [PrefSetAppPreferences\(\)](#). When it receives a `sysAppLaunchCmdNormalLaunch`, it reads the structure back in using [PrefGetAppPreferences\(\)](#).

Storing a single preference structure in the database is a convention that most applications follow because it is convenient to access the all preferences at once. Your application can store more than one

preference resource, if you prefer. This requires more calls to `PrefSetAppPreferences()` and `PrefGetAppPreferences()`, but you may find it more convenient to use several preference resources if you have several variable-length preferences.

## Which Preferences Database to Use

Both `PrefGetAppPreferences()` and `PrefSetAppPreferences()` take a boolean value that indicates whether the value is to be read from and written to the “saved” or the “unsaved” preferences database. To write the preference to the “saved” preferences database, set this boolean value to `true`. To write to the “unsaved” preferences database, set it to `false`.

The only difference between the two databases is that the “saved” preferences database is backed up when a user performs a `HotSync` operation, while the “unsaved” preferences database is not backed up by default. (The user can use a third-party tool to set the backup bit in the “unsaved” preferences database, which would cause it to be backed up.) Both the “saved” and the “unsaved” preferences reside in the storage heap and thus persist across soft resets. The only way that preferences are lost is if a hard reset is performed.

Use the “saved” preferences only for items that must be restored after a hard reset, and use the “unsaved” preferences for the current state of the application. For example, if your application has a registration code, you might write that to the “saved” preferences database so that the user does not have to look up the registration code and re-enter it after a hard reset. However, the loss of such items as the current form being displayed and the current database record being displayed isn’t devastating, so they are written to the “unsaved” preferences database. For games, you might write the high score to the “saved” preferences database and any information about the current game to the “unsaved” preferences database.

It is important to use the “saved” preferences database sparingly. Any time that any application stores or changes a preference in the “saved” preferences database, the **entire** database is backed up during the next `HotSync` operation. For users with a large number of applications, this practice can potentially impact the amount of time that it takes to perform a `HotSync` operation.

## Preferences

### *Setting Application-Specific Preferences*

---

[Listing 3.2](#) shows the preferences structures and the `StopApplication` function from the `HardBall` application. The `HardBallPreferenceType`, which is written to the “saved” preferences database, only stores the high score information and accumulated time. All other preferences are stored in `GameStatusType`, which is written to the “unsaved” preferences database.

#### **Listing 3.2 Saving application-specific preferences**

---

```
typedef struct {
    SavedScore highScore[highScoreMax];
    uint8_t lastHighScore;
    uint8_t startLevel;
    uint32_t accumulatedTime;
} HardBallPreferenceType;

typedef struct {
    enum gameProgress status;
    uint8_t periodLength;
    uint32_t nextPeriodTime;
    uint32_t periodsToWait;
    Boolean paused;
    uint32_t pausedTime;
    BrickType brick[rowsOfBricks][columnsOfBricks];
    uint8_t bricksRemaining;
    uint8_t level;
    WorldState last;
    WorldState next;
    RemovedBrick brokenBricks[brokenBricksMax];
    int16_t brokenBricksCount;
    uint8_t ballsRemaining;
    Boolean movePaddleLeft;
    Boolean movePaddleRight;
    SoundType soundToMake;
    int8_t soundPeriodsRemaining;
    int32_t scoreToAwardBonusBall;
    Boolean lowestHighScorePassed;
    Boolean highestHighScorePassed;
    Boolean gameSpedUp;
    Boolean cheatMode;
    uint32_t startTime;
} GameStatusType;

HardBallPreferenceType Prefs;
static GameStatusType GameStatus;
```

```
static void StopApplication (void)
{
    ...
    // Update the time accounting.
    Prefs.accumulatedTime += (TimGetTicks() -
        GameState.startTime);

    // If we are saving a game resuming (it hasn't started
    // playing yet) then preserve the game status.
    if (GameState.status == gameResuming) {
        GameState.status = SavedGameStatus;
    }

    // Save state/prefs.
    PrefSetAppPreferences (appFileCreator, appPrefID,
        appPrefVersion, &Prefs, sizeof (Prefs), true);

    PrefSetAppPreferences (appFileCreator, appSavedGameID,
        appSavedGameVersion, &GameState, sizeof (GameState),
        false);

    // Close all the open forms.
    FrmCloseAllForms ();
}
```

---

If you have a large amount of preference data that must be backed up during a HotSync operation and is frequently changed, you could, as a performance optimization, store the preferences in a database that your own application creates and maintains rather than in the “saved” preferences database. This saves the user from having to have the entire “saved” preferences database backed up on every HotSync operation. The disadvantage of this technique is that you must write all code to maintain the database and to retrieve information from it.

## Updating Preferences Upon a New Release

When you update your application, you may have new items that you want to store in the preferences database. You may choose to write a separate preference record to the database. However, it is better to update the current preference structure, size permitting.

## Preferences

### *Setting Application-Specific Preferences*

---

The `PrefSetAppPreferences()` and `PrefGetAppPreferences()` functions use a versioning system that allows you to update an existing preference structure. To use it, keep track of the version number that you pass to `PrefSetAppPreferences()`. Add any new preferences to the end of the preferences structure, and then increment the version number. You might use a macro for this purpose:

```
#define CurrentPrefsVersion 2
```

When a user launches the new version of the application, `PrefGetAppPreferences()` is called before `PrefSetAppPreferences()`. The `PrefGetAppPreferences()` function returns the version number of the preference structure that it retrieved from the database. For example, if the new version is version 2, `PrefGetAppPreferences()` returns 1 the first time that version 2 of the application is run. If the returned version does not match the current version, you know that the user does not have values for the new preferences introduced in version 2. You can then decide to provide default values for those new preferences.

The first time any version of your application is run, `PrefGetAppPreferences()` returns `noPreferenceFound`. This indicates that the user does not have any preferences for the current application and the application must supply default values for the entire preferences structure. [Listing 3.3](#) shows how the Datebook handles retrieving the version number from `PrefGetAppPreferences()`.

#### **Listing 3.3 Checking the preference version number**

---

```
#define datebookPrefsVersionNum 4

int16_t DatebookLoadPrefs (DatebookPreferenceType* prefsP)
{
    uint16_t prefsSize;
    int16_t prefsVersion = noPreferenceFound;
    Boolean haveDefaultFont = false;
    uint32_t defaultFont;

    ErrNonFatalDisplayIf(!prefsP, "null prefP arg");

    // Read the preferences / saved-state information. Fix-up if no prefs or
```

```
// older/newer version
prefsSize = sizeof (DatebookPreferenceType);
prefsVersion = PrefGetAppPreferences (sysFileCDatebook, datebookPrefID,
    prefsP, &prefsSize, true);

// If the preferences version is from a future release (as can happen when
// going back and syncing to an older version of the device), treat it the
// same as "not found" because it could be significantly different
if ( prefsVersion > datebookPrefsVersionNum )
    prefsVersion = noPreferenceFound;

if ( prefsVersion == noPreferenceFound ) {
    // Version 1 and 2 preferences
    prefsP->dayStartHour = defaultDayStartHour;
    prefsP->dayEndHour = defaultDayEndHour;
    prefsP->alarmPreset.advance = defaultAlarmPresetAdvance;
    prefsP->alarmPreset.advanceUnit = defaultAlarmPresetUnit;
    prefsP->saveBackup = defaultSaveBackup;
    prefsP->showTimeBars = defaultShowTimeBars;
    prefsP->compressDayView = defaultCompressDayView;
    prefsP->showTimedAppts = defaultShowTimedAppts;
    prefsP->showUntimedAppts = defaultShowUntimedAppts;
    prefsP->showDailyRepeatingAppts =
        defaultShowDailyRepeatingAppts;

    // We need to set up the note font with a default value for the system.
    FtrGet(sysFtrCreator, sysFtrDefaultFont, &defaultFont);
    haveDefaultFont = true;

    prefsP->v20NoteFont = (FontID)defaultFont;
}

if ((prefsVersion == noPreferenceFound) || (prefsVersion <
    datebookPrefsVersionNum)) {
    // Version 3 preferences
    prefsP->alarmSoundRepeatCount = defaultAlarmSoundRepeatCount;
    prefsP->alarmSoundRepeatInterval = defaultAlarmSoundRepeatInterval;
    prefsP->alarmSoundUniqueRecID = defaultAlarmSoundUniqueRecID;
    prefsP->noteFont = prefsP->v20NoteFont;

    // Fix up the note font if we copied from older preferences.
    if ((prefsVersion != noPreferenceFound) && (prefsP->noteFont ==
        largeFont))
        prefsP->noteFont = largeBoldFont;

    if (!haveDefaultFont)
        FtrGet(sysFtrCreator, sysFtrDefaultFont, &defaultFont);
}
```

## Preferences

### *Setting Application-Specific Preferences*

---

```
    prefsP->apptDescFont = (FontID)defaultFont;
}

if ((prefsVersion == noPreferenceFound) || (prefsVersion <
    datebookPrefsVersionNum)) {
    // Version 4 preferences
    prefsP->alarmSnooze = defaultAlarmSnooze;
}

return prefsVersion;
}
```

---

# Sound

---

The Palm OS Sound Manager controls two independent sound facilities:

- **Simple sound:** Single voice, monophonic, square-wave sound synthesis, useful for system beeps.
- **Sampled sound:** Stereo, multi-format, sampled data recording and playback. Sampled sounds can be generated programmatically or read from a soundfile.

These facilities are independent of each other. Although you can play a simple sound and a sampled sound at the same, their respective APIs have no effect on each other. For example, you can't use the sampled sound volume-setting function (`SndStreamSetVolume()`) to change the volume of a simple sound.

Comparing the two facilities, simple sound is easy to understand and requires very little programming: In most cases, you load up a structure, call a function, and out pops a beep. Correspondingly, the sound itself is primitive. (An example of simple sound programming is given in "[Sound Preferences](#)," below.)

Sampled sound, on the other hand, is (or can be) much richer, but requires more planning than simple sound. How much more depends on what you're doing. Playing samples from a soundfile isn't much more difficult than playing a simple sound, but you have to supply a soundfile. Generating samples programmatically—and recording sound—requires more work: You have to implement a callback function that knows something about sound data.

---

**IMPORTANT:** One significant difference between simple sounds and sampled sounds is that they use different volume scales: Simple sound volumes are in the range [0, 64]; sampled sound volumes are [0, 1024].

---

## Sound

### *Playing Simple Sounds*

---

The remainder of this chapter is limited to a discussion of simple sounds. More complex sounds fall under the heading of “Multimedia”; see *Exploring Palm OS: Multimedia* for complete conceptual and reference information.

## Playing Simple Sounds

There are three ways to play a simple sound:

- You can play a single tone of a given pitch, amplitude, and duration by calling `SndDoCmd ( )`.
- You can play a pre-defined system sound (“Information,” “Warning,” “Error,” and so on) through `SndPlaySystemSound ( )`.
- You can play a tune by passing in a Level 0 Standard MIDI File (SMF) through the `SndPlaySmf ( )` function. For example, the alarm sounds used in the built-in Date Book application are MIDI records stored in the System MIDI database. For information on MIDI and the SMF format, go to the official MIDI website, <http://www.midi.org>.

## Sound Preferences

If you’re adding short, “informative” sounds to your application, such as system beeps, alarms, and the like, you should first consider using the (simple) system sounds that are defined by the Palm OS, as listed in the reference documentation for the `SndPlaySystemSound ( )` function.

If you want to create your own system-like sounds, you should at least respect the user’s preferences settings with regard to sound volume. There are a number of sound preference constants:

- `prefSysSoundVolume` is the default system volume.
- `prefGameSoundVolume` is used for game sounds.
- `prefAlarmSoundVolume` is used for alarms.

To apply a sound preference setting to a simple sound volume, you have to retrieve the setting and apply it yourself. For example, here

we retrieve the alarm sound volume and use it to set the volume of a simple sound:

---

```
/* Create a 'sound command' structure. This will encode the parameters of the
tone we want to generate.
*/
SndCommandType sndCommand;

/* Ask for the 'play a tone' command. */
sndCommand.cmd = sndCmdFreqDurationAmp;

/* Set the frequency and duration. */
sndCommand.param1 = 1760;
sndCommand.param2 = 500;

/* Now get the alarm volume and set it in the struct. */
sndCommand.param3 = PrefGetPreference (prefAlarmSoundVolume);

/* Play the tone. */
SndDoCmd( 0, &sndCommand, true);
```

---

For greatest compatibility with multiple versions of the sound preferences mechanism, your application should check the version of Palm OS on which it is running. See [“The Operating System Version Feature”](#) on page 31 for more information. For more information on preferences in general, see [Chapter 33, “Preferences,”](#) on page 405.

For reference information on the Sound Manager APIs, see *Exploring Palm OS: Multimedia*.

## **Sound**

*Sound Preferences*

---

# Expansion

---

This chapter describes how to work with expansion cards and add-on devices using the Palm OS® Expansion and Virtual File System (VFS) Managers.

- [Expansion Support](#) introduces basic terminology and discusses the hardware and file systems supported by the Expansion and VFS Managers.
- [Architectural Overview](#) illustrates the Palm OS expansion architecture and discusses the differences between primary and secondary storage.
- [Applications on Cards](#) covers the various implications of running Palm OS applications from an expansion card.
- [Card Insertion and Removal](#) covers, in detail, the sequence of events that occur when an expansion card is inserted into or removed from an expansion slot.
- [Checking for Expansion Cards](#) shows you how to verify that the handheld supports expansion, how to check each of the handheld's slots for expansion cards, and how to determine the capabilities of a card in a given slot.

## Expansion Support

The Palm OS Expansion and VFS Managers are optional system extensions that provide a standard mechanism by which Palm OS applications can take advantage of the expansion capabilities of various Palm Powered™ handhelds. This capability not only augments the memory and I/O of the handheld, but facilitates data interchange with other Palm Powered handhelds and with devices that aren't running the Palm OS. These other devices include digital cameras, digital audio players, desktop or laptop computers, and the like.

### Primary vs. Secondary Storage

All Palm Powered handhelds contain **primary storage**—directly addressable memory that is used for both long-term and temporary storage. This includes storage RAM, used to hold nonvolatile user data and applications; and dynamic RAM, which is used as working space for temporary allocations.

On most handhelds, primary storage is contained entirely within the device itself. The Palm OS memory architecture doesn't limit devices to this, however; devices can be designed to accept additional storage RAM. The products developed by Handspring™ work this way; memory modules plugged into the Springboard slot are fully-addressable and appear to a Palm OS application as additional storage RAM.

**Secondary storage**, by contrast, is designed primarily to be add-on nonvolatile storage. Although not limited to any particular implementation, most secondary storage media:

- can be inserted and removed from the expansion slot at will
- are based upon a third-party standard, such as Secure Digital (SD) memory cards, MultiMedia (MMC) cards, CompactFlash, Sony's Memory Stick™, and others
- present a serial interface, accessing data one bit, byte, or block at a time

Applications access primary storage either directly, in the case of most dynamic RAM, or through the Database and Resource Managers. To access secondary storage, however, applications use the Expansion and VFS Managers. These have been designed to support as broad a range of serial expansion architectures as possible.

### Expansion Slot

The expansion slots found on many Palm Powered handhelds vary depending on the manufacturer. While some may accept SD and MMC cards, others may accept Memory Stick or CompactFlash. Note that there is no restriction on the number of expansion slots that a given handheld can have.

Depending on the expansion technology used, there can be a wide variety of expansion cards usable with a given handheld:

- Storage cards provide secondary storage and can either be used to hold additional applications and data, or can be used for a specific purpose, for instance as a backup mechanism.
- ROM cards hold dedicated applications and data.
- I/O cards extend the handheld's I/O capabilities. A modem, for instance, could provide wired access, while a Bluetooth™ transceiver could add wireless capability.
- "Combo" cards provide both additional storage or ROM along with some I/O capability.

## Universal Connector

Certain Palm Powered handhelds may be equipped with a universal connector that connects the handheld to a HotSync® cradle. This connector can be used to connect the handheld to snap-on I/O devices as well. A special device driver dedicated to this connector allows handheld-to-accessory communication using the serial portion of the connector. This "plug and play" driver presents the peripheral as a card in a slot, even to the extent of providing the card insertion notification when the peripheral is attached.

Because the universal connector's driver makes a snap-on peripheral appear to be a card in a slot, such peripherals can be treated as expansion cards, at least from an application developer's perspective. For the remainder of this chapter, wherever an I/O card could be used, the phrase "expansion card" can be taken to mean both "expansion card" and "plug and play peripheral."

## Architectural Overview

[Figure 5.1](#) illustrates the Palm OS expansion architecture. It is designed to be flexible enough to support multiple file systems and diverse physical expansion mechanisms while still presenting a consistent set of APIs to applications and to other parts of the Palm OS. The following sections describe the major components of the Palm OS expansion architecture. Working from the bottom up, those

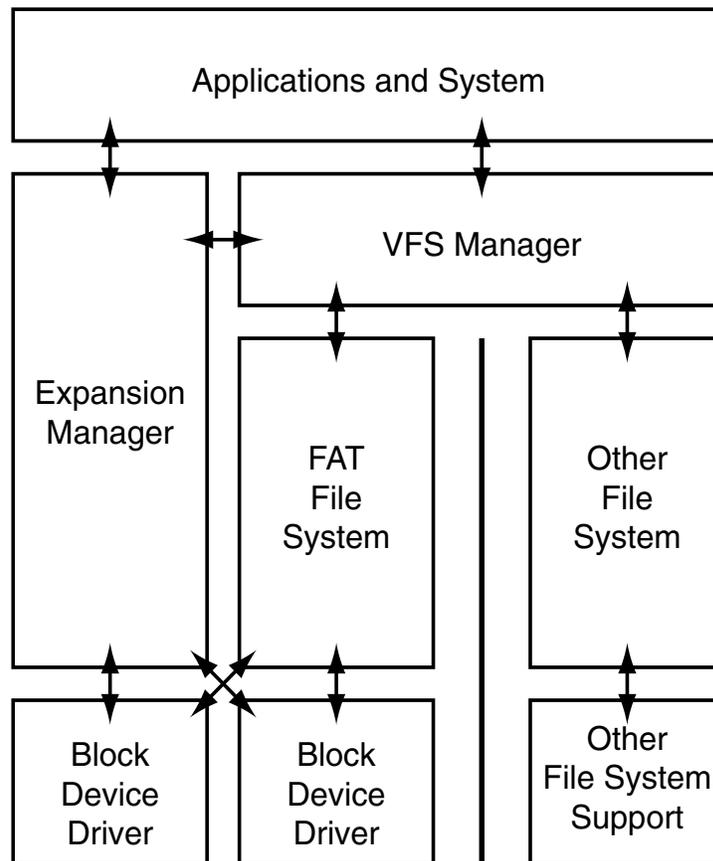
## Expansion

### Architectural Overview

---

components are: block device drivers, file systems, the VFS Manager, and the Expansion Manager.

**Figure 5.1 Palm OS expansion architecture**



### Block Device Drivers

A block device driver is a standard Palm OS shared library that encapsulates direct access to the hardware and provides a standard set of services to the Expansion Manager (and optionally to file system libraries). Adding support for a new type of hardware expansion is usually simply a matter of writing a block device driver for it. As illustrated in [Figure 5.1](#), applications don't normally interact directly with device drivers.

Each expansion slot has a block device driver associated with it. Slots are identified by a unique **slot reference number**, which is

assigned by the Expansion Manager. Expansion cards themselves are not numbered individually; applications typically reference the slot into which a card is inserted. Note, however, that a slot may or may not have a card in it at any given time, and that a card can be inserted and removed while an application is running.

The current implementation only supports one volume per slot.

## **File Systems**

The Palm OS expansion architecture defines a common interface for all file system implementations on the Palm OS. This interface consists of a complete set of APIs for interacting with the file system, including the ability to open, close, read, write, and delete both files and directories on named volumes.

File system implementations are packaged as shared libraries of type `sysFileTFileSystem('libf')`. They are modular plug-ins that add support for a particular type of file system, such as VFAT, HFS, or NFS. The Palm OS expansion architecture allows multiple file system libraries to be installed at any given time. Typically, an implementation of the VFAT file system is present.

VFAT is the industry standard for flash memory cards of all types. It enables easy transfer of data and or applications to desktops and other devices. The VFAT file system library natively supports VFAT file systems on secondary storage media. It is able to recognize and mount FAT and VFAT file systems, and offers to reformat unrecognizable or corrupted media.

Because the VFAT file system requires long filenames to be stored in Unicode/UCS2 format, the standard VFAT file system library supports conversion between UCS2 and Shift-JIS (the standard Palm OS multi-byte character encoding), and the Palm/Latin encoding.

The FAT filesystem component in Palm OS Cobalt supports FAT12/16 and VFAT as well as FAT32. Volumes greater than 512 MB are implicitly formatted as FAT32. However, the system recognizes volumes of any size that are formatted as FAT12 or FAT16.

## VFS Manager

The VFS (Virtual File System) Manager provides a unified API that gives applications access to many different file systems on many different media types. It abstracts the underlying file systems so that applications can be written without regard to the actual file system in use. The VFS Manager includes APIs for manipulating files, directories, and volumes.

---

**NOTE:** Although the great majority of the functions in the VFS Manager can be used by any application, some are intended only for use by drivers and file systems. Others are not intended for use by third-party applications but are designed primarily for system use.

---

### The VFS Manager, the Data Manager, and File Streaming APIs

With the addition of the VFS Manager to the Palm OS, there are now three distinct ways applications can store and retrieve Palm OS user data:

- The Data Manager manages user data in the storage heap. It was specifically designed to make the most of the limited dynamic RAM and the nonvolatile RAM used instead of disk storage on most handhelds. Use the Data Manager to store and retrieve Palm OS user data when storage on the handheld is all that is needed, or when efficient access to data is paramount.
- The File Streaming API is a layer on top of the Data Manager that provides file functionality with all data being read from or written to a database in the storage heap. Most applications have no need for the File Streaming APIs; they are primarily used by applications that need to work with large blocks of data.
- The VFS and Expansion Managers were designed specifically to support many types of expansion memory as secondary storage. The VFS Manager APIs present a consistent interface to many different types of file systems on many types of external media. Applications that use the VFS APIs can support the widest variety of file systems. Use the VFS Manager when your application needs to read and write data stored on external media.

Palm OS applications should use the appropriate APIs for each given situation. The Data Manager, being an efficient manager of storage in the storage heap, should be used whenever access to external media is not absolutely needed. Use the VFS API when interoperability and file system access is needed. Note, however, that the VFS Manager adds the extra overhead of buffering all reads and writes in memory when accessing data, so only applications that specifically need this functionality should use the VFS Manager.

For more information on the Data and Resource Managers, as well as on the File Streaming APIs and the VFS Manager, see *Exploring Palm OS: Memory, Databases, and Files*.

## Expansion Manager

The Expansion Manager is a software layer that manages block device drivers on Palm OS handhelds. Supported expansion card types include, but are not limited to, Memory Stick and SD cards. The Expansion Manager does not support these expansion cards directly; rather, it provides an architecture and higher level set of APIs that, with the help of low level block device drivers and file system libraries, support these types of media.

The Expansion Manager:

- broadcasts notification of card insertion and removal
- plays sounds to signify card insertion and removal
- mounts and unmounts card-resident volumes

---

**NOTE:** Some of the other functions provided by the Expansion Manager are for use by drivers and file systems and are not generally used by their-party applications.

---

For details of the APIs presented by the VFS Manager, see *Exploring Palm OS: Memory, Databases, and Files*.

# Applications on Cards

Palm OS applications located in the `/PALM/Launcher` directory of an expansion card volume appear in a separate Launcher category when the card is inserted into the handheld's expansion slot. If you tap the icon for one of these applications, it is copied to main memory and then launched.

Applications launched from a card ("card-launched" applications) are first sent a `sysAppLaunchCmdCardLaunch` launch code, along with a parameter block that includes the reference number of the volume on which the application resides and the complete path to the application. When processing this launch code, the application shouldn't interact with the user or access globals. Unless the application sets the `sysAppLaunchStartFlagNoUISwitch` bit in the `start` flags (which are part of the parameter block), the application is then sent a `sysAppLaunchCmdNormalLaunch` launch code. This is when the application should, if it needs to, interact with user. Applications may want to save some state when `sysAppLaunchCmdCardLaunch` is received, then act upon that state information when `sysAppLaunchCmdNormalLaunch` is received.

When the user switches to a new application, the card-launched application is removed from main memory. Note, however, that any databases created by the card-launched application remain.

There are certain implications to this "copy and run" process.

- There must be sufficient memory for the application. If the handheld doesn't have enough memory to receive the application, it isn't copied from the expansion card and it isn't launched.
- The copying process takes time. For large applications, this can cause a noticeable delay before the application is actually launched.
- If some version of the application on the card is already present in main memory, the Launcher puts up a dialog that requires the user to choose whether or not to overwrite the in-memory version.
- Card-launched applications have a limited lifetime: applications reside in main memory only while they are

running. When the user switches to a different application, the card-launched application that was just running is removed from main memory. If the card-launched application is then re-launched, it is once again copied into the handheld's memory.

- “Legacy” applications—those that are unaware that they are being launched from a card—only work with databases in main memory. Associated databases aren't copied to main memory along with the application unless the database is bundled with the application. Databases created by card-launched applications are not removed along with the application, however, so this data is available to the application when it is subsequently run. Applications that are written to take advantage of the VFS Manager can read and write data on the expansion card, so this limitation generally only applies to legacy applications.

Bundled databases, although copied to main memory along with their associated application, are meant for static data that doesn't change, such as a game level database. Bundled databases are not copied back to the card; they are simply deleted from memory when the user chooses another application. To bundle a database with an application, give it the same creator ID as the owning application, set the `dmHdrAttrBundle` bit, and place it in the `/PALM/Launcher` directory along with the application.

- Unless a card-launched application is running, it doesn't receive notifications or launch codes since it isn't present on the handheld. In particular, these applications don't receive notifications and aren't informed when an alarm is triggered.

## Card Insertion and Removal

The Expansion Manager supports the insertion and removal of expansion media at any time. The handheld continues to run as before, though an application switch may occur upon card insertion. The handheld need not be reset or otherwise explicitly informed that a card has been inserted or removed.

## Expansion

### Card Insertion and Removal

---

---

**WARNING!** Due to the way certain expansion cards are constructed, if the user removes an expansion card while it is being written to, in certain rare circumstances it is possible for the card to become damaged to the point where either it can no longer be used or it must be reformatted. To the greatest extent possible, applications should only write to the card at well-defined points, and the application should warn the user—perhaps with a “Please Wait” or progress dialog—at that time not to remove the expansion card. The card can be removed while an application is reading from it without fear of damage.

---

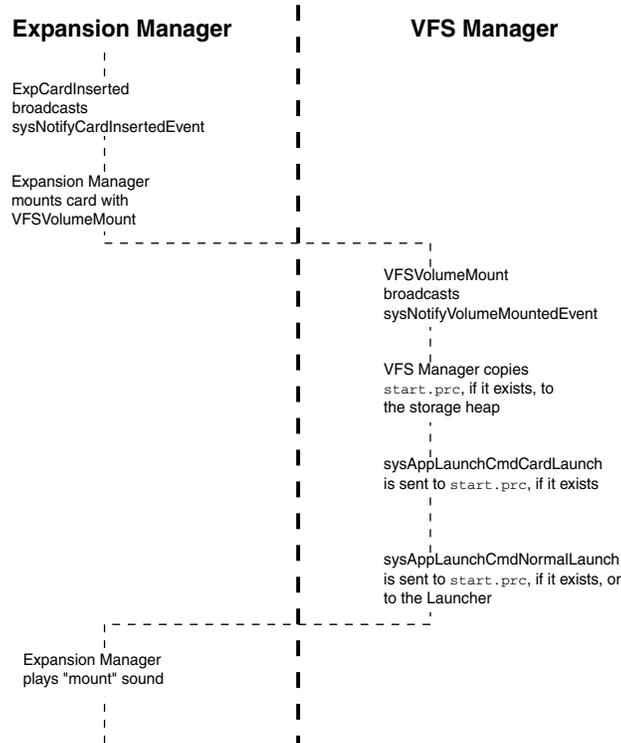
The Palm OS uses a series of notifications to indicate that a card has been inserted or removed, or that a volume has been mounted or unmounted. The following table lists these notifications, and the priority for which they have been registered by the Expansion and VFS Managers. Note that the priorities may change in a future release, so applications shouldn’t depend on these precise values. Applications that register for these using normal priority get the correct behavior.

**Table 5.1 Expansion card notifications**

Notification	Registered by	Priority
<a href="#"><u>sysNotifyCardInsertedEvent</u></a>	Exp. Manager	20
<a href="#"><u>sysNotifyCardRemovedEvent</u></a>	Exp. Manager	-20
<a href="#"><u>sysNotifyVolumeMountedEvent</u></a>	Exp. Manager	-20
<a href="#"><u>sysNotifyVolumeMountedEvent</u></a>	VFS Manager	10
<a href="#"><u>sysNotifyVolumeUnmountedEvent</u></a>	Exp. Manager	-20

The following diagram shows the sequence of events that occur when an expansion card is inserted into a Palm Powered handheld’s expansion slot. For clarity, it assumes that no errors occur. If the card doesn’t contain a mountable volume, and if the card cannot be formatted and then mounted, this sequence is aborted and the card remains unmounted, although the card insertion notification is still broadcast.

Figure 5.2 Sequence of events upon card insertion



The Expansion Manager registers for `sysNotifyCardInsertedEvent` with a priority of 20, ensuring that it is notified after other handlers that may have registered with normal priority. To override the Expansion Manager's default handler, register your handler to receive `sysNotifyCardInsertedEvent` with normal priority, and have it set the appropriate bits in the `handled` member of the `SysNotifyParamType` structure:

- `expHandledVolume` indicates that any volumes associated with the card have been dealt with, and prevents the Expansion Manager from mounting or unmounting the card's volumes.
- `expHandledSound` indicates that your application has handled the playing of an appropriate sound, and prevents

## Expansion

### *Card Insertion and Removal*

---

the Expansion Manager from playing a sound when the card is inserted or removed.

Note that the number of the slot into which the card was inserted is passed to your handler using the `notifyDetailsP` member—which is a `UInt16`, cast to a `void *`—of the `SysNotifyParamType` structure.

Although most applications only register for volume mount and unmount notifications, if you need to receive notifications when the user removes a card from a slot managed by the Expansion Manager, have your application register to receive `sysNotifyCardRemovedEvent`. Unlike with `sysNotifyCardInsertedEvent`, the Expansion Manager registers for `sysNotifyCardRemovedEvent` with a priority of -20, ensuring that it receives the notification before other handlers that are registered for it with normal priority. This notification, too, passes the number of the slot from which the card was removed to your handler using the `notifyDetailsP` member—which is a `UInt16`, cast to a `void *`—of the `SysNotifyParamType` structure.

The VFS Manager registers for `sysNotifyVolumeMountedEvent` with a priority of 10. To override the VFS Manager's default handler, register your handler to receive `sysNotifyVolumeMountedEvent` with normal priority, and have it set the appropriate bits in the `handled` member of the `SysNotifyParamType` structure:

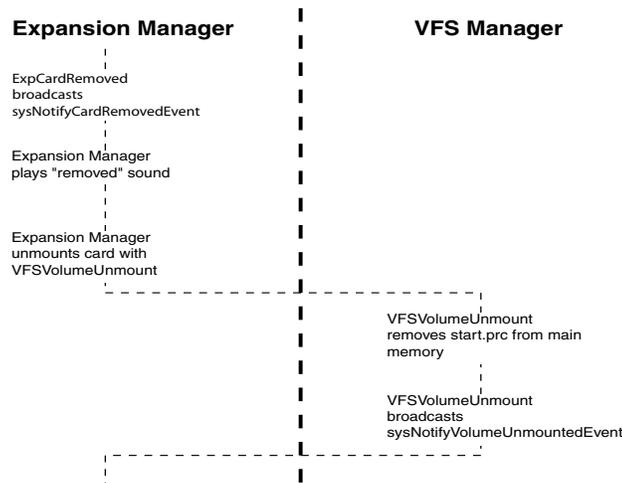
- `vfsHandledUIAppSwitch` indicates that your application has handled `SysUIAppSwitch` to `start.prc`. This bit prevents the VFS Manager from performing its own `SysUIAppSwitch` to `start.prc` (although `start.prc` is still loaded and a `SysAppLaunch` is performed), and also prevents the launcher from switching to itself.
- `vfsHandledStartPrc` indicates that your handler has dealt with the automatic running of `start.prc`. The VFS Manager won't load it and won't call either `SysAppLaunch` or `SysUIAppSwitch`.

Note that if your application handles the running of `start.prc`, you need to keep security in mind. If the handheld is locked when an expansion card is inserted, the VFS Manager's own handler

defers the execution of `start.prc` until the user unlocks the handheld.

Card removal follows a similar sequence, although there is no equivalent to `start.prc` that is automatically run. This sequence is illustrated in the following diagram.

**Figure 5.3 Sequence of events upon card removal**



Upon card removal, the Expansion Manager broadcasts a notification to all applications that have registered to receive card removal notifications and unmounts any mounted volumes on the card. This causes the VFS Manager to issue a card unmounted notification. Each application must register for the card unmounted notification and provide the necessary error handling code if card removal at any time will cause a problem for the application.

Note that the card insertion and removal notifications are intended primarily for system use, although they can be registered for by applications that need them. Applications that deal only with file systems and the VFS Manager should confine themselves to the volume mounted and unmounted notifications.

## Expansion

### *Card Insertion and Removal*

---

## Start.prc

Upon receipt of a `sysNotifyVolumeMountedEvent` that hasn't already been handled (as indicated by the state of the `vfsHandledStartPrc` bit, as described in the previous section), the VFS Manager copies `/Palm/start.prc`—and its overlay, if there is one—to the storage heap and launches it. This process enables “application cards”—single-function cards that execute automatically upon card insertion. It also allows for combo cards that automatically load any necessary drivers and applications to support card I/O.

To launch `start.prc`, the VFS Manager first sends it a special launch code, `sysAppLaunchCmdCardLaunch`. If the application only needs to do a bit of work and return, it should do it here and then set the `sysAppLaunchStartFlagNoUISwitch` bit in the start flags, which are part of the `sysAppLaunchCmdCardLaunch` parameter block. Note that the application doesn't have access to globals and it shouldn't interact with the user here. If the `sysAppLaunchStartFlagNoUISwitch` bit is not set, as it isn't if the application ignores the `sysAppLaunchCmdCardLaunch` launch code, the VFS Manager then sends it a `sysAppLaunchCmdNormalLaunch` launch code to run the application normally. This ensures backwards compatibility with applications that do not understand the `sysAppLaunchCmdCardLaunch` launch code. This is where the application can interact with the user; an application may want to save state when it receives `sysAppLaunchCmdCardLaunch`, and then act upon that state when it receives `sysAppLaunchCmdNormalLaunch`.

To avoid running out of stack space, the VFS Manager sets the “new stack” bit when launching `start.prc`. The `start.prc` application remains in system memory until the volume from which it was copied is removed. `start.prc` is deleted before `VFSVolumeUnmount` broadcasts `sysNotifyVolumeUnmountedEvent` but after the Expansion Manager broadcasts `sysNotifyCardRemovedEvent`. By registering for `sysNotifyCardRemovedEvent`, `start.prc` can react to the volume being removed before it is deleted.

---

**NOTE:** If an expansion card is inserted while the handheld is locked, `start.prc` is not executed until the user unlocks the handheld.

---

## Checking for Expansion Cards

Before looking for an expansion card, your program should first make sure that the handheld supports expansion by verifying the presence of the Expansion and VFS Managers. It can then query for mounted volumes. Finally, your program may want to ascertain the capabilities of the card; whether it has memory, whether it does I/O, and so on. The following sections describe each of these steps.

### Verifying Handheld Compatibility

There are many different Palm OS handhelds, and in the future there will be many more. Some will have expansion slots to support secondary storage, and some will not. Hardware to support secondary storage is optional, and may or may not be present on a given handheld. Since the Expansion and VFS Managers are of no use on a handheld that has no physical expansion capability, they are optional system extensions that are not present on every Palm Powered handheld.

Due to the great variability both in handheld configuration and in the modules which can be plugged into or snapped onto the handheld, applications shouldn't attempt to detect the manufacturer or model of a specific handheld when determining if it supports secondary storage. Instead, check for the presence and capabilities of the underlying operating system.

### Checking for Mounted Volumes

Many applications rely on the handheld's expansion capabilities for additional storage. Applications that don't care about the physical characteristics of the secondary storage module, and that don't need to know the slot into which the module is inserted, can rely on the fact that the Palm OS automatically mounts any recognized volumes inserted into or snapped onto the handheld. Thus, many

## Expansion

### Checking for Expansion Cards

---

applications can simply enumerate the mounted volumes and select one as appropriate. The following code illustrates how to do this:

#### Listing 5.1 Enumerating mounted volumes

---

```
UInt16 volRefNum;
UInt32 volIterator = vfsIteratorStart;

while (volIterator != vfsIteratorStop) {
    err = VFSVolumeEnumerate(&volRefNum, &volIterator);
    if (err == errNone) {
        // Do something with the volRefNum
    } else {
        // handle error... possibly by
        // breaking out of the loop
    }
}
```

---

The volume reference number obtained from [VFSVolumeEnumerate\(\)](#) can then be used with many of the volume, directory, and file operations that are described later in this chapter.

Occasionally an application needs to know more than that there is secondary storage available for use. Those applications likely need to take a few extra steps, beginning with checking each of the handheld's slots.

## Enumerating Slots

Before you can determine which expansion modules are attached to a Palm OS handheld, you must first determine how those modules could be attached. Expansion cards and some I/O devices could be plugged into physical slots, and snap-on modules could be connected through the handheld's universal connector. Irrespective of how they're physically connected, the Expansion Manager presents these to the developer as slots. Enumerating these slots is made simple due to the presence of the [ExpSlotEnumerate\(\)](#) function. The use of this function is illustrated here:

### **Listing 5.2 Iterating through a handheld's expansion slots**

---

```
UInt16 slotRefNum;
UInt32 slotIterator = expIteratorStart;

while (slotIterator != expIteratorStop) {
    // Get the slotRefNum for the next slot
    err = ExpSlotEnumerate(&slotRefNum, &slotIterator);
    if(err == errNone) {
        // perform slot-specific processing here
    } else {
        // handle error... possibly by
        // breaking out of the loop
    }
}
```

---

The slot reference number returned by `ExpSlotEnumerate` uniquely identifies a given slot. This can be supplied to various Expansion Manager functions to obtain information about the slot, such as whether there is a card or other expansion module present in the slot.

#### **Checking a Slot for the Presence of a Card**

Use the [ExpCardPresent](#) function to determine if a card is present in a given slot. Given the slot reference number, this function returns `errNone` if there is a card in the slot, or an error if either there is no card in the slot or there is a problem with the specified slot.

#### **Determining a Card's Capabilities**

Just knowing that an expansion card is inserted into a slot or connected to the handheld isn't enough; your application needs to know something about the card to ensure that the operations it needs to perform are compatible with the card. For instance, if your application needs to write data to the card, it's important to know if writing is permitted.

The capabilities available to your application depend not only on the card but on the block device driver as well. Handheld manufacturers will provide one or more block device drivers that define standard interfaces to certain classes of expansion hardware. Card and device manufacturers may also choose to provide card-

## Expansion

### *Summary of Expansion Manager*

---

specific block device drivers, or they may require that applications use the slot custom control function and a registered creator code to access and control certain cards.

The block device driver is responsible for querying expansion cards for a standard set of capabilities. When a block device driver is present for a given expansion card, you can use the [ExpCardInfo](#) function to determine the following:

- the name of the expansion card's manufacturer
- the name of the expansion card
- the "device class," or type of expansion card. Values returned here might include "Ethernet" or "Backup"
- a unique identifier for the device, such as a serial number
- whether the card supports both reading and writing, or whether it is read-only
- whether the card supports a simple serial interface

Note that the existence of the `ExpCardInfo` function does not imply that all expansion cards support these capabilities. It only means that the block device driver is able to assess a card and report its findings up to the Expansion Manager.

## Summary of Expansion Manager

---

### Expansion Manager Functions

---

<a href="#">ExpCardInfo()</a>	<a href="#">ExpSlotEnumerate()</a>
<a href="#">ExpCardIsFilesystemSupported()</a>	<a href="#">ExpSlotMediaType()</a>
<a href="#">ExpCardMediaType()</a>	<a href="#">ExpSlotPowerCheck()</a>
<a href="#">ExpCardMetrics()</a>	
<a href="#">ExpCardPresent()</a>	
<a href="#">ExpCardSectorRead()</a>	
<a href="#">ExpCardSectorWrite()</a>	
<a href="#">ExpSlotCustomControl()</a>	

---

# Shared Libraries

---

A **shared library** is an executable module that is compiled and linked separately. Like all executable modules, a shared library is contained in a PRC file that is installed into either the system storage heap or some type of external storage media. After being installed to the storage heap, a PRC becomes a resource database that contains several resources including:

- The executable module's code segment.
- A copy of the module's initialized static data.
- Relocation information.

Executable modules residing in the system storage heap are executed in place, while those in external storage media must be copied into the storage heap before being executed. In either case, the runtime environment must be set up before a code segment can be executed correctly. The runtime services are responsible for preparing the necessary runtime environment for an executable module. They are also responsible for cleaning up the runtime environment after an executable module has exited.

The Palm OS Cobalt runtime services consist of the following components:

**Program Loader:** Sets up the necessary runtime environment for executable modules. It's also involved in the process of applying patches when loading patchable shared libraries.

**Dynamic Linker:** Resolves function calls across executable module boundaries (for instance, from an application to a shared library).

**Boot/Process Startup Code:** Sets up the initial runtime environment at boot time and each time a new process is started.

Applications can make calls to **shared libraries**, which are separately compiled and linked executable modules. Shared

## Shared Libraries

---

libraries loaded by an application are executed as subroutines of that application. Consequently, shared libraries are confined by the boundary of the calling application's process.

The operating system is presented to an application as a set of shared libraries. Some operating system shared library functions invoke software interrupts (SWIs) internally, which effectively transfer control to the kernel. For the most part, applications do not invoke SWIs directly. As far as an application is concerned, operating system calls are simply subroutine calls to shared libraries.

Static data of applications and shared libraries have per-process instances. This is also known as **process-own data**. Process-own data enables the same program to execute in multiple processes simultaneously, each with independent state.

The term “executable module” encompasses applications, shared libraries, and **plug-ins**—manually-loaded shared libraries. From the runtime services' point of view, applications, shared libraries, and plug-ins are no different. An executable module:

- must have executable instructions in its code segment.
- must have an identifiable main entry point in its code segment. This main entry point must have the following C prototype:

```
uint32_t PilotMain(uint16_t cmd, void *cmdPBP,
uint16_t launchFlags)
```

- may have zero or more additional exported entry points in its code segment. Exported entry points can have arbitrary C prototypes.
- may have a data segment to hold its state (static data) at runtime.

Each executable module can be given a type, such as “application” or “shared library,” when it's placed into a container (PRC file). However, these types are of interest only to higher-level operating system services. Executable modules of all types are treated by the runtime services without regard to type.

The following features also apply to executable modules in Palm OS Cobalt:

- All executable modules can always have static data. C static and global variables can be used freely in programs. The runtime services allocate, initialize and de-allocate memory for such variables automatically.
- Because shared libraries have main entry points, they are “launchable” and so able to register for and receive notifications as applications do.
- Every executable module receives a [sysLaunchCmdInitialize](#) launch code right after it is loaded and a [sysLaunchCmdFinalize](#) launch code right before it is unloaded by the program loader. This gives the program a chance to perform customized initialization and de-initialization.

## Exporting Globals

Palm OS Cobalt executable modules cannot export global data directly. However, the runtime services do provide means for exporting data indirectly, at a procedure call level.

An executable module that wants to make all or part of its globals accessible by other modules can do this by putting those globals in a single C structure and defining this structure as part of its external API. The executable module should handle the [sysLaunchCmdGetGlobals](#) launch code in its [PilotMain\(\)](#) function by returning the address of this global structure in the memory location pointed to by the *cmdPBP* parameter of [PilotMain\(\)](#). When, in order to get an executable’s globals, an application calls [SysGetModuleGlobals\(\)](#) and passes the reference number of a module that implements [sysLaunchCmdGetGlobals](#), the function returns either the address of the global structure or the address of the module’s data segment, depending on the value of the *wantStructure* parameter.

Whether [SysGetModuleGlobals\(\)](#) is able to return the address of the globals structure depends on whether the module identified by *refNum* defines such a structure and returns its address in response to the [sysLaunchCmdGetGlobals](#) launch code. This function returns `sysErrNotSupported` if *wantStructure* is `true` and the module doesn’t allow the globals structure address to

## Shared Libraries

### *Patching Shared Libraries*

---

be retrieved. (To prevent globals from being retrieved, simply return NULL in response to a `sysLaunchCmdGetGlobals` launch code.)

If `wantStructure` is false when `SysGetModuleGlobals()` is called, the base address of the module's data segment is returned. For executable modules that don't support the `sysLaunchCmdGetGlobals` launch code, and for those that return NULL in response to this launch code, this is the only way to gain access to the shared library's data. However, in this case the caller will have to possess sufficient knowledge on the memory map of the shared library's data segment in order to access data items located at certain offsets.

---

**NOTE:** If when `wantStructure` is false the returned base address of the module's data segment is NULL, the module has no static data.

---

## Patching Shared Libraries

### Patch Configuration Database

When multiple patches exist on the same shared library entry point, calls to that entry point will go through a chain of functions. Each of the functions in the chain can do something interesting and optionally, at some point, invoke the next function in the chain. The patch's configuration information determines the order in which it should be called. The program loader manages this information in a patch configuration database, and provides a set of APIs for manipulating it.

### Constructing a Patch

A patch is an executable module with its own code and data segments. Besides the common 'acod' (code) and 'adat' (data) resources, each patch has an additional 'amdp' (ARM Module Patch) resource.

Within the 'acod' resource of a patch, like that of a shared library, there is an embedded vector table, each entry of which points to a patching function within the same 'acod' resource. However, the

vector table of a patch is different from that of a shared library in that its entries are not arranged in the order of increasing index numbers. The ' amdp ' resource supplies information for the program loader to know which entry in the patch's vector table applies to which entry of which shared library.

An ' amdp ' resource is simply an array of `SysPatchTargetType` structures; these structures look like this:

---

```
typedef SysPatchTargetType {
    SysPatchTargetHeaderType header;
    SysPatchEntryNumType entryNums[];
} SysPatchTargetType;
```

---

Each `SysPatchTargetType` structure identifies a set of entry points of a shared library that this patch wants to patch. This implies that a single patch can target multiple shared libraries simultaneously. Each element of the `entryNums` array corresponds to a patching function whose address is listed in this patch's own vector table. The value of the element is the entry number of the shared library that the corresponding patching function is targeting.

The `SysPatchTargetType` structure contains a header field that identifies the shared library being patched. This header is defined as follows:

---

```
typedef SysPatchTargetHeaderType {
    uint32_t type;
    uint32_t creator;
    uint16_t rsrcID;
    uint16_t flags;
    uint32_t numEntries;
} SysPatchTargetHeaderType;
```

---

The `type`, `creator`, and `rsrcID` fields identify the type, creator ID, and resource ID of the shared library being patched. The `numEntries` field identifies the number of entries in the `SysPatchTargetType` structure's `entryNums` array. The remaining field, `flags`, has two bits defined. These bits indicate whether the patch must be the head or the tail in the call chain. Normally, a patch should work properly regardless of its position within the call chain, so these flags are very rarely used. Use these flags only when absolutely needed, and use them with caution; just

## Shared Libraries

### Patching Shared Libraries

---

because you request that a patch be placed at a particular position doesn't guarantee that it will be placed there.

You build a patch just as you would a normal shared library. Just make sure that you construct the 'amdp' resource properly. For instance, the following steps show how you might patch `SystemMain()`, `SysGetEvent()`, and `SysHandleEvent()` of `SystemLib`.

1. Construct the patch's shared library definition file, as shown [Listing 6.1](#).

#### Listing 6.1 A sample shared library definition file

---

```
EXPORTS
    MySystemMain; name of my patching function
    MySysGetEvent; name of my patching function
    MySysHandleEvent; name of my patching function
```

---

2. Build an 'amdp' resource. [Listing 6.2](#) shows how to do this.

#### Listing 6.2 Defining an 'amdp' resource

---

```
typedef MyPatchRsrcType {
    SysPatchTargetHeaderType system;
    SysPatchEntryNumType systemEntries[3];
} MyPatchRsrcType;

const MyPatchRsrcType myPatchRsrc = {
    {
        'prdm', // type of SystemLib
        'psys', // creator ID of SystemLib
        0, // resource ID of SystemLib
        3, // number of entries to patch
        0 // flags - no requirement to be head or tail
    },
    {
        kEntryNumMain, // entry number of SystemMain
        3, // entry number of SysGetEvent
        7 // entry number of SysHandleEvent
    }
};
```

---

When constructing your 'amdp' resource, note that the entry numbers of the patched shared libraries must appear in

*exactly* the same order as the corresponding patching functions in the shared library definition file. Also, the entry numbers of any patched shared library must appear in the ' amdp ' resource in the order of increasing entry numbers. If the main entry point is patched, it must precede any other entries.

3. Package the ' acod ' (code), ' adat ' (data), and ' amdp ' resources, all with the same resource ID, in a single PRC of type ' apch ' using your creator ID.

If this sample patch is installed onto a device, it will be loaded whenever the target shared library (SystemLib, in this example) is loaded into a process in which this patch is allowed to run.

## Registering the Patch

You don't have to register a patch if it is packaged in a PRC of type ' apch '. The program loader automatically recognizes patches of this type.

If a patch is packaged in a PRC whose type is not ' apch ', you must call [SysRegisterPatch\(\)](#) to register it with the program loader. Such patches aren't loaded if they are not registered. Note that you don't have to un-register a patch when it is deleted from the storage heap since the program loader automatically un-registers a registered patch if it can't find it in the storage heap.

## The Patch Call Chain

Using the ' amdp ' resources, the program loader builds a chain of functions for each patched shared library entry. The program loader maintains such call-chain information for every patched shared library entry point. It also provides a means for each patch to retrieve the address of the next function in the chain.

Each patch has its own main entry point, `PilotMain()`. Like the main entry point of any executable module, the `PilotMain()` of a patch receives [sysLaunchCmdInitialize](#) and [sysLaunchCmdFinalize](#) launch codes when the patch is loaded and unloaded. As well, a patch receives a third launch code—[sysPatchLaunchCmdSetInfo](#)—once for each of the shared libraries it patches. The launch code's `cmdPBP` parameter is a

## Shared Libraries

### *Patching Shared Libraries*

---

pointer to a [SysPatchInfoType](#) structure, which provides information about the patched shared library that is being loaded.

The `sysPatchLaunchCmdSetInfo` launch code tells the patch that one of the shared libraries it wants to patch is being loaded, providing the patch with a good opportunity to retrieve and save addresses of functions in the next patch in the call chain. Taking the patch used to illustrate how you construct a patch in “[Constructing a Patch](#)” on page 74 as an example, its `PilotMain()` function might handle the `sysPatchLaunchCmdSetInfo` launch code like this (note that this patch is patching two shared libraries: `DALLib` and `SystemLib`):

#### **Listing 6.3 Getting and saving function addresses from the next patch in the call chain**

---

```
// These static variables save pointers to globals of the patched libraries
static void *gOriginalDALGlobalsP;
static void *gOriginalSystemGlobalsP;

// These static variables save function pointers of next patches
static KALThreadCreateProcType *gNextKALThreadCreateP;
static KALThreadDestroyProcType *gNextKALThreadDestroyP;
static KALThreadStartProcType *gNextKALThreadStartP;
static SysMainEntryType *gNextSystemMainP;
static SysGetEventProcType *gNextSysGetEventP;
static SysHandleEventProcType *gNextSysHandleEventP;

UInt32 PilotMain(UInt32 cmd, MemPtr cmdPBP, UInt16 launchFlags) {
    switch(cmd) {
        case sysPatchLaunchCmdSetInfo:
            {
                UInt32 refNum = (SysPatchInfoType*)cmdPBP->refNum;
                UInt32 myIndex = (SysPatchInfoType*)cmdPBP->index;
                UInt32 type = (SysPatchInfoType*)cmdPBP->type;
                UInt32 creator = (SysPatchInfoType*)cmdPBP->creator;
                UInt32 rsrcID = (SysPatchInfoType*)cmdPBP->rsrcID;
                Err (*getNextPatchFuncP)(UInt32, UInt32, UInt32, void **) =
                    (SysPatchInfoType*)cmdPBP->sysGetNextPatchP;

                if(type == 'boot' && creator == 'psys' && rsrcID == 0) {
                    // DALLib is being loaded
                    // Save pointer to original globals so that we can
                    // access them later
                    SysGetModuleGlobals(refNum, &gOriginalDALGlobalsP);
                }
            }
    }
}
```

```
        // Get and save pointers to next procedures in the call chain
        (*getNextPatchFuncP)(refNum, kEntryNumKALThreadCreate,
            myIndex+1, &gNextKALThreadCreateP);
        (*getNextPatchFuncP)(refNum, kEntryNumKALThreadDestroy,
            myIndex+1, &gNextKALThreadDestroyP);
        (*getNextPatchFuncP)(refNum, kEntryNumKALThreadStart,
            myIndex+1, &gNextKALThreadStartP);

    } else if(type == 'boot' && creator == 'psys' && rsrcID == 1) {

        // SystemLib is being loaded
        // Save pointer to original globals so we can access them later
        SysGetModuleGlobals(refNum, &gOriginalSystemGlobalsP);

        // Get and save pointers to next procedures in the call chain
        (*getNextPatchFuncP)(refNum, kEntryNumMain, myIndex+1,
            &gNextSystemMainP);
        (*getNextPatchFuncP)(refNum, kEntryNumSysGetEvent, myIndex+1,
            &gNextSysGetEventP);
        (*getNextPatchFuncP)(refNum, kEntryNumSysHandleEvent, myIndex+1,
            &gNextSysHandleEventP);
    }

    break;
}
}
}
```

---

For more information on the functions used in the above example, see [Chapter 30, “Loader,”](#) on page 379.

Once the call chain function pointers have been saved as illustrated above, when a patching function wants to invoke the next function in the call chain it can simply use the appropriate saved function pointer.

Patches are always loaded and unloaded along with the original shared library. During loading, the original shared library is loaded first, and then the patches are loaded in the reverse of the order that they occur in the call chain (that is, the first patch in the call chain is the last loaded). During unloading, the patches are unloaded in the order in which they occur in the call chain (the first patch in the call chain is the first to be unloaded), after which the original shared library is unloaded. Note that both the original shared library and the patches receive [sysLaunchCmdInitialize](#) and

## Shared Libraries

### *Patching Shared Libraries*

---

[sysLaunchCmdFinalize](#) launch codes. If the `PilotMain()` of a shared library is patched, then the patching function is invoked before that of the shared library, just like any other patched entry.

## Adding and Removing Patches

Patches can be added to or removed from the patch configuration database at any time with [SysRegisterPatch\(\)](#) and [SysUnregisterPatch\(\)](#). The program loader builds the call chains for a shared library only while loading that shared library for the first time in a process. It never tries to modify a shared library's call chains while that shared library is running. Thus, changes made to the patch call chain for a running shared library won't take effect until the shared library is unloaded and then reloaded.

When adding or removing a patch from the patch configuration database, you must take the following into account in order for the new configuration to take effect:

- If the shared library being patched is not yet loaded in any process, nothing further has to be done. The new configuration will take effect automatically when that shared library is loaded.
- If the shared library being patched is already loaded in any process, the new configuration will not take effect in that process until the shared library is unloaded and then reloaded. If it is important that the new configuration take effect immediately, the best thing to do is to restart that process.
- If the shared library being patched is one of the modules involved in the boot sequence (such as `SystemLib`), the whole device must be reset in order for the new configuration to take effect.

After a patch is installed onto the system, unless it can be determined that the target shared libraries are not currently running the patch usually needs to request a soft reset of the device in order for it to be applied.

---

**NOTE:** In order to run in a secure process, a patch must be digitally signed by a party that is trusted by that process.

---

## Patch Security

Earlier releases of Palm OS allowed any program to patch operating system entry points without restriction. Palm OS Cobalt restricts the patching of operating system entry points in that:

- Only patches signed by trusted parties are applied in secure processes.
- Not all of the operating system entry points are patchable. Non-patchable entry points include operating system functions that execute in supervisor mode (kernel functions, interrupt service routines, and the like).

## The Program Loader

The program loader is responsible for loading and unloading executable modules in the calling process. It also provides a means for retrieving information about executable modules and a means for patching shared library entries.

### Program Loader Library Functions

The program loader is implemented as a set of library functions exported by one of the core operating system shared libraries. The library functions perform most of their operations within the context of the calling process, but they also interact with the Application Manager thread (in the System process) to perform certain security-related operations.

See [Chapter 30, “Loader,”](#) on page 379 for a complete description of each of the loader APIs.

### Shared Libraries and the Program Loader

Programmers can use the program loader in either of the following ways when working with shared libraries:

**The implicit approach:** The client program of a shared library can be statically linked with call stubs of that shared library. The client program can then make shared library calls as if those shared library functions are defined locally. In this scenario the client program does not need to make an operating

## Shared Libraries

### *The Program Loader*

---

system call to load that shared library before calling its functions. Whenever the client program makes a call to a shared library function, the stub of that function is entered first. The first time the client program makes a call to a shared library the stub automatically invokes the program loader and dynamic linker to load that shared library and link it with the client, after which it redirects the call to its target location. Once a client program has loaded a shared library, subsequent calls made by that client program are routed to the shared library directly without the involvement of the program loader and dynamic linker. A shared library loaded this way is automatically unloaded when the operating system detects that all client programs that loaded that shared library have exited.

**The explicit approach:** The client program of a shared library can explicitly call operating system functions ([SysLoadModule\(\)](#) or [SysLoadModuleByDatabaseID\(\)](#)) to load that shared library and retrieve addresses of exported functions ([SysGetEntryAddresses\(\)](#)). Then, it can make calls to those functions indirectly using the acquired function addresses. After using the shared library, the client program must manually unload it with a call to [SysUnloadModule\(\)](#). If a shared library is used in this way, the client program does not have to be linked with the call stubs of that shared library. Shared libraries used in this way are also known as **plug-ins**.

# System Reset

---

A reset is normally performed by sticking your stylus, a bent-open paper clip, or the like into a small hole in the back of the device. Depending on additional keys held down, the reset behavior varies, as described in this chapter.

## Soft Reset

A soft reset clears the dynamic heaps of all applications. The storage heap remains untouched. The operating system restarts from scratch with a new stack, new global variables, restarted drivers, and a reset communication port. Applications on the device with the appropriate preferences bit set receive a [`sysAppLaunchCmdSystemReset`](#) launch code.

## Safe-Mode Reset

Holding down the scroll up button while pressing the reset switch causes the same soft reset logic with the following two exceptions:

- The `sysAppLaunchCmdSystemReset` launch code is not sent to applications. This is useful if there is an application on the device that crashes upon receiving this launch code (not uncommon) and therefore prevents the system from booting.
- The OS won't load any program that's installed in RAM during startup. This is useful if you have to delete or replace an executable module in RAM. If a module is loaded—and therefore its database is open—it cannot be replaced or deleted from the system.

## Hard Reset

A hard reset is performed by pressing the reset switch while holding down the power key. This has all the effects of the soft reset. In

## System Reset

### *System Reset Calls*

---

addition, the storage heap is erased. As a result, all programs, data, patches, user information, and so on in RAM are lost. A confirmation message is displayed asking the user to confirm the deletion of all data.

The `sysAppLaunchCmdSystemReset` launch code is sent to the applications at this time. If the user selected the “Delete all data” option, the digitizer calibration screen comes up first. The default databases for the four main applications are copied out of the ROM.

## System Reset Calls

The System Manager provides support for rebooting the Palm Powered device. It calls [SysReset\(\)](#) to reset the device. This call does a soft reset and has the same effect as pressing the reset switch on the unit. *Normally applications should not use this call.*

`SysReset()` is used, for example, by the Sync application. When the user copies a patch that has the “reset after installation” bit set onto the Palm Powered device, the Sync application automatically resets the device after the sync is completed to allow the patch to be applied.

# Threading

---

Two of the primary differences between Palm OS Cobalt and earlier versions of Palm OS are the introduction of protected memory and multi-threading. This chapter discusses these features and how you can make use of them in your Palm OS applications.

This chapter is divided into the following broad topics:

<a href="#">Architecture Overview</a> . . . . .	85
<a href="#">Using the Threading APIs</a> . . . . .	92
<a href="#">Summary of Threading</a> . . . . .	99

## Architecture Overview

To understand the Palm OS Cobalt architecture, you must understand some of the terminology used in Palm OS Cobalt and how those terms compare with their use in other operating systems. Note that not all of the features described here are available in the SDK.

## Threads and Scheduling

A **thread** is autonomous unit of execution with its own set of registers, stack, program counter, and other state needed to execute code. Palm OS Cobalt allows for multiple threads executing simultaneously. A thread that would like to execute code is in a READY state. When two or more threads are ready to run, the system schedules them based on their priority, using round-robin scheduling for multiple threads with the same priority.

[Table 8.1](#) lists the possible states for a Palm OS Cobalt thread:

**Table 8.1 Thread states**

State	Description
RUNNING	The thread is running. That is, it is currently executing code.
READY	The thread is ready to run, but because it is not the highest priority thread it is queued on the ready list.
WAITING	The thread is blocked, waiting for some condition to clear or resource to become available. A thread can wait on one or more other threads by calling <a href="#">SysThreadGroupWait()</a> .
SUSPENDED	The thread is suspended. A suspended thread does not resume execution until it has been resumed. You suspend a thread by calling <a href="#">SysThreadSuspend()</a> and resume a suspended thread by calling <a href="#">SysThreadResume()</a> .
WAIT-SUSPEND	The thread is both waiting and suspended.
DORMANT	The thread has been created but not yet started.
FAULTED	During execution of the thread's code, a fault occurred.
FAULT-SUSPEND	The thread is both faulted and suspended.

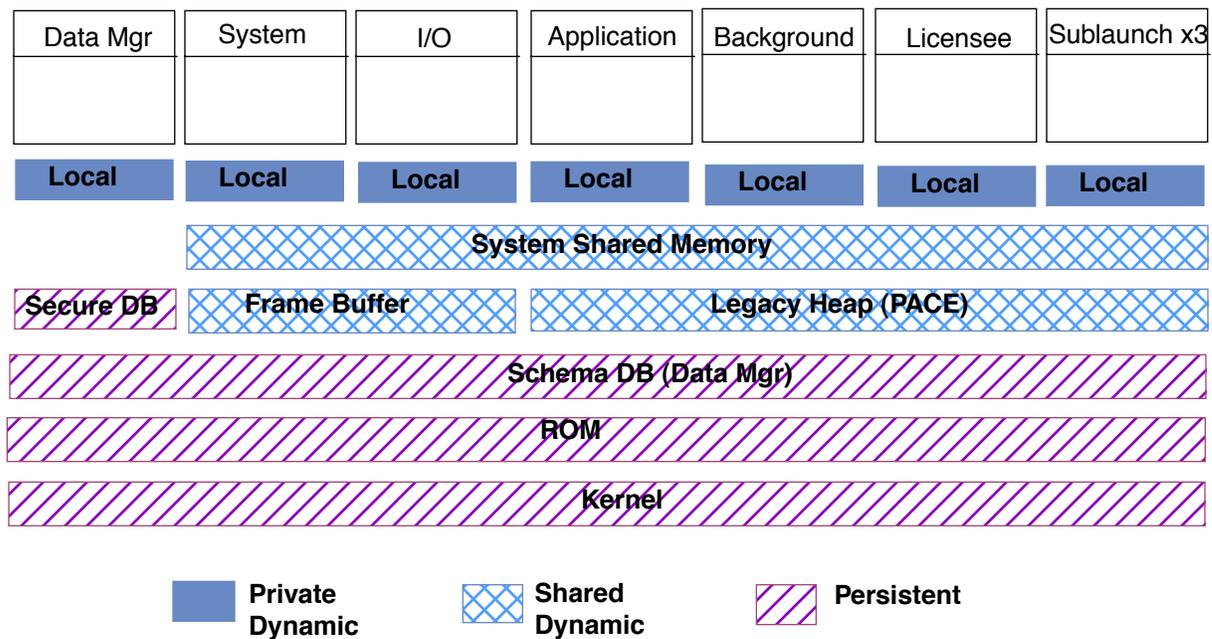
Thread **priority** levels range from 1 (best) to 255 (worst). The thread with the best priority that is ready to run is always scheduled at the next context switch, and all threads of worse priority are stopped until the best-priority thread has blocked. Although thread priorities range from 1 to 255, in reality user threads are typically restricted to a priority no better than 30 (`sysThreadPriorityBestUser`). See "[Thread Priorities](#)" on page 454 for a list of commonly-used thread priorities.

Palm OS Cobalt allows for thread-specific data. This takes the form of slots containing integer values, with each thread having its own value in a particular slot.

## Processes and Applications

A **process** is a protected environment in which one or more threads reside. A thread must reside in one and only one process. The process provides that thread's heap and other global state, executable code, and communication channels with other parts of the system. The system enforces restrictions on access to memory and other resources between processes. It is not, by default, possible for a thread in one process to access the resources in another process, although such access may be granted if requested. Palm OS Cobalt provides for shared memory areas that are accessible to multiple processes. All threads running in a particular process have unrestricted access to all resources in that process. [Figure 8.1](#), below, illustrates the Palm OS Cobalt processes and the access they have to the various local and shared memory areas.

**Figure 8.1 Process-to-Memory Mapping**



## Threading

### Architecture Overview

---

Note that third-party developers can write code that runs in either the Application, Background, or Sublaunch processes (and note that there are up to three Sublaunch processes in Palm OS Cobalt).

Dynamic creation of processes is not supported in Palm OS Cobalt. The system creates a fixed set of processes while booting up and may create and destroy some well-defined processes on behalf of applications as needed. Due to restrictions on the total number of available processes, you cannot create new processes in an ad-hoc manner.

### Applications and Sublaunching

An application runs in its own process, called the **Application process**. Only one application runs at a time. When an application switch occurs, the currently-running application exits, and its process is torn down. A new Application process is then created, and the new application is started within it.

When an application asks another application to perform some service on its behalf, this is known in Palm OS as a **sublaunch**. When a sublaunch occurs, the sublaunched code is loaded into the Application process and executed in-place. *This is intrinsically unsafe, however, because the sublaunched code has complete access to the hosting application.* Because of this, Palm OS Cobalt provides for **remote sublaunching**. During a remote sublaunch, a new temporary process is created in which the target code is executed. This process is granted temporary status as the Application process until the sublaunch completes. It can do anything that a normal application can. While the remote sublaunched code is executing, the main application is completely blocked, so there is effectively still only one flow of execution.

All occurrences in the system of sublaunches that are outside of the application's direct control (such as notifications) are performed as a remote sublaunch.

### Background Threads

Palm OS Cobalt also creates a process called the **Background Process**. Applications can use this process to execute code that needs to persist across application switches. (Any threads created in the Application process are torn down with the rest of the process as

part of an application switch.) The system provides APIs for applications to spawn threads in the Background Process, which are then free to run independently from the main UI.

Note that code from multiple independent applications may be running in the Background Process, and any application is free to load code into the process as desired. *Because of this, the Background Process is not a secure address space.* Secure operations must be executed in the Application process, where the application has full control over what is loaded. In addition, crashing code will bring down all other threads running in the Background Process. There are facilities for applications to be notified of a thread crashing so they may restart any desired threads.

## Thread Synchronization

Palm OS Cobalt provides two major categories of synchronization primitives. Traditional primitives are implemented in the kernel and can be used to synchronize threads across processes. Lightweight primitives are implemented in user space and can only be used to synchronize threads in the same process. The main goal of the lightweight primitives is to reduce resource usage, as they require no kernel state and as little as four bytes of storage. However, they also tend to have better performance—any operation that does not require blocking or interacting with another blocking thread can be done entirely in user space without a kernel call.

### Traditional Synchronization Primitives

Among the traditional synchronization primitives that Palm OS Cobalt supports are mutexes and counted semaphores.

A **mutex** is a simple locking primitive. Only one thread can hold a mutex at a time; all other threads trying to acquire the mutex are blocked until the first thread releases it, at which point the next thread gains access to the mutex and continues its execution.

A **counted semaphore** provides a very general synchronization primitive, which can be used to construct many other types of synchronization semantics (mutex, reader/writer, producer/consumer, and so on). A variation on this, the fast semaphore, is a more efficient implementation for threads in the same process—but most code should use a lightweight critical section instead.

### Lightweight Synchronization Primitives

In addition to mutexes and counted semaphores, Palm OS Cobalt also supports critical sections and condition variables, both of which fall under the classification of “lightweight primitives.”

A **critical section** is the lightweight version of a mutex. A critical section is simply a set of statements that can only be executed by a single process at any given time.

A Palm OS Cobalt **condition variable** provides a subset of traditional condition variable semantics. The operations supported in Palm OS Cobalt are:

- Close
- Open
- Wait: causes the thread to block until the condition variable is open
- Broadcast: causes all waiting threads to continue

Condition variables can be combined with critical sections similar to POSIX semantics—the critical section is atomically released and acquired while blocked on the condition variable.

### Inter-Process Communication

There are a variety of methods in Palm OS Cobalt for getting data from one process to another. The following sections briefly summarize them, from the lowest level up.

#### Shared Memory

There are a variety of services in the system that can be placed under the category of shared memory. While many of these don’t actually use shared memory for their implementation, relying instead on IPC or other communication mechanisms, they all share the key characteristic of placing data in a common location that threads can access in an ad-hoc manner. These provide little to no support for synchronization and no way to target data to specific threads.

### **Feature Manager**

The Feature Manager is a simple name-space mapping of tuples (creator ID, feature number) to integer values. In Palm OS Cobalt the Feature Manager is a system service, so the data it contains is global to the system. A thread in one process can place a value in the Feature Manager, which can then be read from any other process. In Palm OS Cobalt there is no concept of an owner for a feature, so the system cannot remove features created by an application after it has gone away. Accordingly, *all data in the Feature Manager is lost upon a system reset.*

### **Data Manager**

The Data Manager maintains a system-global pool of persistent data. While there is some access control available in the Data Manager APIs, this can be generally considered a shared memory area. Because data here is persistent, it generally isn't useful for passing transient data across processes. Like the Feature Manager, there is no strong concept of ownership, so the system cannot easily clean up data that is no longer needed.

### **Memory Dealer**

While the Palm OS Cobalt kernel fully supports memory segments and access control to them across processes, limitations on the total number of available segments means that applications can't directly make use of this facility. This is the underlying reason for the limitation on number of processes.

### **Events**

Traditional Palm OS events can be delivered to the currently running application from any other process. There are also APIs for sending events to specific windows and event queues—but note that these only work within the same process.

### **Named Pipes**

Threads can use IOS to create a named pipe, through which other threads can rendezvous for communication. These can be created and accessed by any thread in any process. The traditional IOS APIs are used to send and receive data, and both data and IOS file descriptors can be sent through a named pipe.

## Threading

### *Using the Threading APIs*

---

## Graphics Context

All of the traditional UI APIs in Palm OS Cobalt are associated with a “graphics context,” which is bound by thread. There are APIs available for creating and destroying the graphics context associated with a thread; these APIs can be used from any process. For example, a thread running in the Background Process can create a graphics context, bring up a dialog, and then close the dialog, destroy the context, and continue. The application’s thread—known as the **Main UI Thread**—is given a special graphics context with additional features not available to other threads; these additional features are needed for backward compatibility. Windows in the Main UI Thread can be back-buffered, and the thread also serves as the target for all application-related events.

There is no way to directly communicate between different threads doing UI—even in the same process, the threads have completely separate graphics contexts. As mentioned above, there are some APIs that allow events to be delivered to the event queue of a graphics context for communication with its thread.

## Using the Threading APIs

The APIs for the features previously described come from a variety of places in the system. Many are slight abstractions on top of the low-level kernel API in the System Manager. Others are part of the UI and application model; these are declared in `Event.h` and `Window.h`.

The main APIs in the SDK are essentially the traditional Palm OS APIs working on an underlying protected memory system.

## Application Launching

Most of the traditional Palm OS application APIs exist in Palm OS Cobalt and work as in previous versions of the OS. In particular, [`SysAppLaunch\(\)`](#) performs a sublaunch in the same address space as the current application, and [`SysUIAppSwitch\(\)`](#) performs a switch to a new application. Unlike previous versions of the OS, an application switch results in the current application’s process being

torn down and restarted, so no local state, such as memory chunks, can persist across switches.

[`SysAppLaunchRemote\(\)`](#) is like `SysAppLaunch()`, but performs the sublaunch in a separate, freshly-created process. This allows applications to perform sublaunches of code that isn't trusted without compromising their own security.

[`SysNotifyBroadcastDeferred\(\)`](#) allows notifications to be sent from outside of the Main UI Thread. Like `SysAppLaunchDeferred()`, this function is asynchronous, so no result is returned directly from the function.

## Launching in the Background Process

Use [`EvtCreateBackgroundThread\(\)`](#) to create a new thread in the Background Process and launch the specified PRC in that thread.

These threads in the Background Process run independently of the main application—they can continue running through any application switches. In the case of the Background Process crashing and restarting, applications can register to receive a notification of this event in order to restart any needed background threads.

There is a limited set of things these background threads can do. Most Palm OS APIs work from within the Background Process—including IOS, Data Manager, and Feature Manager functions—but there are a number that are not allowed. Some of the more important ones include:

- [`SysAppLaunch\(\)`](#) and [`SysAppLaunchRemote\(\)`](#) can only be called from the main UI thread.
- [`SysNotifyBroadcast\(\)`](#) can only be called from the main UI thread. Use [`SysNotifyBroadcastDeferred\(\)`](#) from other threads.

## Manipulating Threads

The APIs in `SysThread.h` provide a simple mechanism for creating threads. They allow developers to implement multi-threading in their own application, and not just in the special environment of the Background Process. This is especially useful for

## Threading

### *Using the Threading APIs*

---

I/O, so that network and other I/O operations don't stop the application's UI or force you into a convoluted programming model.

Palm OS Cobalt provides a number of functions for creating and managing threads. These are defined in `SysThread.h`.

[`SysThreadCreate\(\)`](#) and [`SysThreadStart\(\)`](#) are used together to create and then start a thread. They allow control over the new thread's priority, stack size, and entry function. The resulting thread can exit either by returning from its initial function or with an explicit call to [`SysThreadExit\(\)`](#). Use [`SysThreadDelay\(\)`](#) to cause a thread to sleep for a specified amount of time.

---

**NOTE:** The application's primary thread is always created with the default stack size (the default stack size is set by the licensee for each device, and thus is not a constant value across the range of Palm Powered devices). Applications should avoid doing operations that need excessive stack in the primary thread. Instead, spawn new threads to perform such operations.

---

[`SysThreadInstallExitCallback\(\)`](#) allows the current thread to install a function that is executed when the thread exits.

[`SysThreadRemoveExitCallback\(\)`](#) gets rid of an exit callback that was previously installed for the thread.

[`SysThreadChangePriority\(\)`](#) changes the scheduling priority of a thread. You must supply the ID of the thread being changed. Note that the ID of the current thread can be determined by calling [`SysCurrentThread\(\)`](#).

### Thread Groups

There are three functions for creating and managing **thread groups**. Thread groups are a convenience provided by the operating system that allows you to wait for one or more threads to exit. Thread groups are useful for unloading libraries that have spawned their own threads. Note that destroying a thread group implicitly waits for all threads in that group to exit.

[`SysThreadGroupCreate\(\)`](#) creates a new thread group. When you create the thread group, you must specify all of the threads that are to be a member of the group. Another thread can then call

[SysThreadGroupWait\(\)](#) to block on the threads in the thread group; once all of the threads in the group have exited, the thread waiting on the thread group will resume.

You can destroy a thread group with [SysThreadGroupDestroy\(\)](#). Note that this function waits until all of the thread group's threads have exited.

## Inter-Process Communication (IPC)

There are a number of ways in which process can communicate with each other in Palm OS Cobalt.

### IPC with Shared Memory

The traditional Palm OS Feature Manager and Data Manager APIs are available to all threads in the system. Because these use a common memory area, they can be used for simple IPC operations. These are a very limited solution, however, and won't meet the needs of many developers, particularly developers of enterprise applications.

### IPC with Named Pipes

The IOS subsystem in Palm OS Cobalt includes a facility called **named pipes** that can be used to transfer data across processes. This is accomplished by calling [IOSPipe\(\)](#) from within one process to create the pipe and then publishing it under a specific name with [IOSFattach\(\)](#). At this point, a thread in any other process can open the pipe as a normal device using the [IOSOpen\(\)](#) call and start performing I/O on it using [IOSRead\(\)](#), [IOSWrite\(\)](#), and other IOS functions.

Applications can send IOS file descriptors through these pipes. You cannot, however, transfer other system objects such as semaphores, memory segments, and the like.

### Communication in Same Process

Event queues are a simple mechanism that allow you to easily communicate between threads in the same process. Each thread has an event queue; a thread can obtain a handle to its event queue by calling [EvtGetThreadEventQueue\(\)](#). Be sure to release this

## Threading

### Using the Threading APIs

---

handle once you no longer need it by calling [EvtReleaseEventQueue\(\)](#).

[EvtAddEventToQueue\(\)](#) sends an event to the currently running application. Because “current application” is always a moving target, the receiver of this API is not always clear, so this function is not generally useful. It does, however, provide at least one channel of communication from the Background Process to the Application process. Applications will likely want to use [EvtAddEventToEventQueue\(\)](#) instead. This function sends an event to a specific event queue, identified by the event queue’s handle. It also allows you to specify a “reply queue” to which replies can be posted. Often you’ll want the calling thread’s queue to be the reply queue—the thread handle returned from [EvtGetThreadEventQueue\(\)](#) will serve this purpose.

### Communicating with the Background Process

Event queues can also be used when communicating across process boundaries. [EvtCreateBackgroundThread\(\)](#) creates a thread in the Background Process and returns a communication path back to the caller through which events can be sent. This allows the original application to control the background thread by sending it event codes. An optional parameter to this function, *callerQueue*, allows the caller to pass to the background thread an event queue that the background thread can use to pass events back to the caller. To have the background thread post replies back to the calling thread’s queue, supply the value returned from [EvtGetThreadEventQueue\(\)](#) as the *callerQueue* parameter.

From within the background thread, you then obtain the caller’s queue with [EvtGetReplyEventQueue\(\)](#).

### Attaching to a Running Background Thread

For the case where an application needs to attach to its already running background thread whenever it starts, use the following functions to publish and then retrieve the background process’ event queue by name:

- [EvtPublishEventQueue\(\)](#)
- [EvtLookupEventQueue\(\)](#)

## Atomic Operations

There are a variety of atomic operation APIs that are useful for applications. [SysAtomicAdd32\(\)](#), [SysAtomicAnd32\(\)](#), [SysAtomicOr32\(\)](#) perform an atomic operation of the given type on a 32-bit quantity. [SysAtomicCompareAndSwap32\(\)](#) atomically changes a 32-bit value to a new arbitrary value, but only if the current value is the same as one provided.

## Synchronization

Palm OS Cobalt provides wrappers for the lightweight critical section and condition variable primitives discussed under “[Lightweight Synchronization Primitives](#)” on page 90. The critical section APIs consist of two functions, [SysCriticalSectionEnter\(\)](#) and [SysCriticalSectionExit\(\)](#), that allow you to acquire and release a critical section, respectively.

The condition variable functions include:

- [SysConditionVariableWait\(\)](#) (which can optionally exit and enter a critical section when blocking)
- [SysConditionVariableClose\(\)](#)
- [SysConditionVariableOpen\(\)](#)
- [SysConditionVariableBroadcast\(\)](#)

The above functions do not require creation and destruction functions, as the objects themselves are simple 32-bit values that are initialized to a well-defined constant:

`sysCriticalSectionInitializer` in the case of a critical section, and `sysConditionVariableInitializer` in the case of a condition variable.

## Thread-Specific Data

There is a set of functions in `SysThread.h` for working with thread-specific data (TSD):

- [SysTSDAllocate\(\)](#) allocates a new TSD slot. You can supply an optional destructor function which is called whenever a thread exits to clean up any data associated with

## Threading

### Using the Threading APIs

---

the slot. You can also specify a name; multiple calls to `SysTSDAllocate()` with the same slot name results in the same slot being returned instead of a new one being allocated each time.

- [SysTSDFree\(\)](#) deallocates a previously created TSD slot.

---

**NOTE:** `SysTSDFree()` is very dangerous and shouldn't normally be used.

---

- [SysTSDGet\(\)](#) returns the data in a particular slot.
- [SysTSDSet\(\)](#) changes the data in a particular slot.

## Accessing the User Interface from Outside the Main UI Thread

With just two functions, it is possible for background threads to bring up their own UI, which greatly increases what can be done in them. For example, a background thread could present a progress dialog.

`Window.h` declares functions for performing UI operations outside of the main UI thread: [WinStartThreadUI\(\)](#) and [WinFinishThreadUI\(\)](#). These functions control the lifetime of the calling thread's graphics context. You can nest calls to these functions, so you can use them in a function to ensure that the current thread has a graphics context while inside that function, whether or not there is one from the outside caller.

Once a thread that is not the main UI thread has been bound to a graphics context, it can make use of almost all of the Palm OS UI APIs. There are, however, some limitations placed on them. The most important is that these windows can only be update-based; they do not support back buffering. This carries with it all of the other implications of update-based windows, which are discussed in *Exploring Palm OS: User Interface*. One additional limitation is that these threads cannot call [IOSPoll\(\)](#) for multiplexing UI operations with I/O. Instead, they are expected to spawn additional threads for handling the I/O and can use the new Event Manager APIs described earlier to interact with the UI thread.

## Summary of Threading

---

### Controlling Individual Threads

---

<a href="#"><u>SysCurrentThread()</u></a>	<a href="#"><u>SysThreadChangePriority()</u></a>
<a href="#"><u>SysThreadCreate()</u></a>	<a href="#"><u>SysThreadCreateEZ()</u></a>
<a href="#"><u>SysThreadDelay()</u></a>	<a href="#"><u>SysThreadExit()</u></a>
<a href="#"><u>SysThreadInstallExitCallback()</u></a>	<a href="#"><u>SysThreadRemoveExitCallback()</u></a>
<a href="#"><u>SysThreadResume()</u></a>	<a href="#"><u>SysThreadStart()</u></a>
<a href="#"><u>SysThreadSuspend()</u></a>	

---

### Thread Groups

---

<a href="#"><u>SysThreadGroupCreate()</u></a>	<a href="#"><u>SysThreadGroupDestroy()</u></a>
<a href="#"><u>SysThreadGroupWait()</u></a>	

---

### Thread-Specific Data

---

<a href="#"><u>SysTSDAllocate()</u></a>	<a href="#"><u>SysTSDFree()</u></a>
<a href="#"><u>SysTSDGet()</u></a>	<a href="#"><u>SysTSDSet()</u></a>

---

### Atomic Operations

---

<a href="#"><u>SysAtomicAdd32()</u></a>	<a href="#"><u>SysAtomicAnd32()</u></a>
<a href="#"><u>SysAtomicCompareAndSwap32()</u></a>	<a href="#"><u>SysAtomicOr32()</u></a>

---

### Condition Variables

---

<a href="#"><u>SysConditionVariableBroadcast()</u></a>	<a href="#"><u>SysConditionVariableClose()</u></a>
<a href="#"><u>SysConditionVariableOpen()</u></a>	<a href="#"><u>SysConditionVariableWait()</u></a>

---

## Threading

### *Summary of Threading*

---

---

#### **Critical Sections**

---

[SysCriticalSectionEnter\(\)](#)

[SysCriticalSectionExit\(\)](#)

---

---

#### **Semaphores**

---

[SysSemaphoreCreate\(\)](#)

[SysSemaphoreCreateEZ\(\)](#)

[SysSemaphoreDestroy\(\)](#)

[SysSemaphoreSignal\(\)](#)

[SysSemaphoreSignalCount\(\)](#)

[SysSemaphoreWait\(\)](#)

[SysSemaphoreWaitCount\(\)](#)

---

---

#### **Miscellaneous Functions**

---

[SysGetRunTime\(\)](#)

---

# Power Management

---

Palm OS® differs from a traditional desktop system in that it's never really turned off. Power is constantly supplied to essential subsystems, and the power button is merely a way of bringing the device in or out of low-power mode. The obvious effect of pressing the power button is that the LCD turns on or off. When the user presses the power key to turn the device off, the LCD is disabled, which makes it appear as if power to the entire unit is turned off. In fact, the memory system, real-time clock, and the interrupt generation circuitry are still running, though they are consuming little current.

This chapter looks at Palm OS power management, discussing the following topics:

- [Palm OS Power Modes](#)
- [Guidelines for Application Developers](#)
- [Power Management Calls](#)

## Palm OS Power Modes

To minimize power consumption, the operating system dynamically switches between three different modes of operation: sleep mode, doze mode, and running mode. The System Manager controls transitions between different power modes and provides an API for controlling some aspects of the power management.

- In **sleep mode**, the device looks like it's turned off: the display is blank, the digitizer is inactive, and the main clock is stopped. The only circuits still active are the real-time clock and interrupt generation circuitry.

The device enters this mode when there is no user activity for a number of minutes or when the user presses the power button. The device comes out of sleep mode only when there is an interrupt, for example, when the user presses a button.

## Power Management

### *Palm OS Power Modes*

---

To enter sleep mode, the system puts as many peripherals as possible into low-power mode and sets up the hardware so that an interrupt from any hard key or the real-time clock wakes up the system. When the system gets one of these interrupts while in sleep mode, it quickly checks that the battery is strong enough to complete the wake-up and then takes each of the peripherals, for example, the LCD, serial port, and timers, out of low-power mode.

- In **doze mode**, the main clock is running, the device appears to be turned on, the LCD is on, and the processor's clock is running but it's not executing instructions (that is, it's halted). When the processor receives an interrupt, it comes out of halt and starts processing the interrupt.

The device enters this mode whenever it's on but has no user input to process.

The system can come out of doze mode much faster than it can come out of sleep mode since none of the peripherals need to be woken up. In fact, it takes no longer to come out of doze mode than to process an interrupt. Usually, when the system appears on, it is actually in doze mode and goes into running mode only for short periods of time to process an interrupt or respond to user input like a pen tap or key press.

- In **running mode**, the processor is actually executing instructions.

The device enters this mode when it detects user input (like a tap on the screen) while in doze mode or when it detects an interrupt while in doze or sleep mode. The device stays in running mode only as long as it takes to process the user input (most likely less than a second), then it immediately reenters doze mode. A typical application puts the system into running mode only about 5% of the time.

To maximize battery life, the processor on the Palm Powered™ device is kept out of running mode as much as possible. Any interrupt generated on the device must therefore be capable of “waking” the processor. The processor can receive interrupts from the serial port, the hard buttons on the case, the button on the cradle, the programmable timer, the memory module slot, the real-time clock (for alarms), the low-battery detector, and any built-in peripherals such as a pager or modem.

## Guidelines for Application Developers

Normally, applications don't need to be aware of power management except for a few simple guidelines. When an application calls [EvtGetEvent\(\)](#) to ask the system for the next event to process, the system automatically puts itself into doze mode until there is an event to process. As long as an application uses `EvtGetEvent()`, power management occurs automatically. If there has been no user input for the amount of time determined by the current setting of the auto-off preference, the system automatically enters sleep mode without intervention from the application.

Applications should avoid providing their own delay loops. Instead, they should use [SysTaskDelay\(\)](#), which puts the system into doze mode during the delay to conserve as much power as possible. If an application needs to perform periodic work, it can pass a timeout to `EvtGetEvent()`; this forces the unit to wake up out of doze mode and to return to the application when the timeout expires, even if there is no event to process. Using these mechanisms provides the longest possible battery life.

## Power Management Calls

The system calls `SysSleep()` to put itself immediately into low-power sleep mode. Normally, the system puts itself to sleep when there has been no user activity for the minimum auto-off time or when the user presses the power key.

The [SysSetAutoOffTime\(\)](#) routine changes the auto-off time value. This function is normally only used by the Preferences application and by the system during boot. The Preferences application saves the user preference for the auto-off time in a preferences database, and the system initializes the auto-off time to the value saved in the preferences database during boot. While the auto-off feature can be disabled entirely by calling `SysSetAutoOffTime()` with a timeout value of 0, doing this depletes the battery.

The current battery level and other information can be obtained through the [SysBatteryInfo\(\)](#) routine. This function returns

## **Power Management**

### *Power Management Calls*

---

information about the battery, including the current battery voltage in hundredths of a volt, the warning thresholds for the low-battery alerts, the battery type, and whether external power is applied to the unit. This call can also change the battery warning thresholds and battery type.

# The ROM Serial Number

---

Some Palm™ devices hold a 12-digit serial number that uniquely identifies the device. The serial number is held in a displayable text buffer with no null terminator. The user can view the serial number in the Application Launcher application. The Application Launcher also displays to the user a checksum digit that you can use to validate user entry of the serial number.

To retrieve the ROM serial number programmatically, pass the `sysROMTokenSnum` selector to the `SysGetROMToken()` function. If the `SysGetROMToken()` function returns an error, or if the returned pointer to the buffer is `NULL`, or if the first byte of the text buffer is `0xFF`, then no serial number is available.

The `DrawSerialNumOrMessage()` function shown in [Listing 10.1](#) retrieves the ROM serial number, calculates the checksum, and draws both on the screen at a specified location. If the device has no serial number, this function draws a message you specify. This function accepts as its input a pair of coordinates at which it draws output, and a pointer to the message it draws when a serial number is not available.

## Listing 10.1 DrawSerialNumOrMessage

---

```
static void DrawSerialNumOrMessage(Int16 x, Int16 y, Char*
noNumberMessage)
{
    Char* bufP;
    UInt16* bufLen;
    Err retVal;
    Int16    count;
    UInt8    checkSum;
    Char    checksumStr[2];
           // holds the dash and the checksum digit
```

## The ROM Serial Number

---

```
retval = SysGetROMToken (0, sysROMTokenSnum,
    (UInt8**) &bufP, &bufLen);
if ((!retval) && (bufP) && ((UInt8) *bufP != 0xFF)) {
    // there's a valid serial number!
    // Calculate the checksum: Start with zero, add each
    // digit, then rotate the result one bit to the left
    // and repeat.
    checksum = 0;
    for (count=0; count<bufLen; count++) {
        checksum += bufP[count];
        checksum = (checksum<<1) | ((checksum&0x80)>>7);
    }
    // Add the two hex digits (nibbles) together, +2
    // (range: 2 - 31 ==> 2-9, A-W)
    // By adding 2 before converting to ascii,
    // we eliminate the numbers 0 and 1, which can be
    // difficult to distinguish from the letters O and I.
    checksum = ((checksum>>4) & 0x0F)+(checksum & 0x0F)+2;

    // draw the serial number and find out how wide it was
    WinDrawChars(bufP, bufLen, x, y);
    x += FntCharsWidth(bufP, bufLen);

    // draw the dash and the checksum digit right after it
    checksumStr[0] = '-';
    checksumStr[1] =
        ((checksum < 10) ? (checksum +'0') :
        (checksum -10 +'A'));
    WinDrawChars (checksumStr, 2, x, y);
} else // there's no serial number
    // draw a status message if the caller provided one
    if (noNumberMessage)
        WinDrawChars(noNumberMessage,
            StrLen(noNumberMessage),x, y);
}
```

---

# Time

---

The Palm Powered™ device has a real-time clock and programmable timer. The real-time clock maintains the current time even when the system is in sleep mode (turned off). It's capable of generating an interrupt to wake the device when an alarm is set by the user. The programmable timer is used to generate the system tick count interrupts while the processor is in doze or running mode. The system tick interrupts are required for periodic activity such as polling the digitizer for user input, key debouncing, and so on.

The Date and Time Manager (called Time Manager in this chapter) provides access to the timing resources on the Palm Powered device.

The basic Time Manager API provides support for setting and getting the real-time clock in seconds and for getting the current **system ticks** value (but not for setting it). The System Manager provides more advanced functionality for setting up a timer task that executes periodically or in a given number of system ticks, and includes a number of macros that allow you to convert a system time value (a time interval, in ticks) to and from more natural units such as seconds and milliseconds.

This chapter discusses the following topics:

- [Using Real-Time Clock Functions](#)
- [Using System Ticks Functions](#)

## Using Real-Time Clock Functions

The real-time clock functions of the Time Manager include [TimSetSeconds\(\)](#) and [TimGetSeconds\(\)](#). Real time on the Palm Powered device is measured in seconds from midnight, Jan. 1, 1904. Call [TimSecondsToDateTime\(\)](#) and

## Time

### *Using System Ticks Functions*

---

[TimDateTimeToSeconds\(\)](#) to convert between seconds and a structure specifying year, month, day, hour, minute, and second.

## Using System Ticks Functions

The Palm Powered device maintains a tick count that starts at 0 when the device is reset. This tick increments 1000 times per second when running on the Palm Powered device.

For tick-based timing purposes, applications could use the macro [SysTicksPerSecond\(\)](#), which is conditionally compiled for different platforms. However, the use of a more natural set of units, such as seconds or milliseconds, is preferable. The various `SysTimeIn...` macros, such as [SysTimeInMilliSecs\(\)](#), allow you to convert a system time value (a time interval, in ticks) to more conventional units. To convert from such units back to a system time value, use one of the `SysTimeTo...` macros, such as [SysTimeToMilliSecs\(\)](#).

Use the function [TimGetTicks\(\)](#) to read the current tick count.

Although the `TimGetTicks()` function could be used in a loop to implement a delay, it is recommended that applications use the [SysTaskDelay\(\)](#) function instead. The `SysTaskDelay()` function automatically puts the unit into low-power mode during the delay. Using `TimGetTicks()` in a loop consumes much more current.

# Floating Point

---

Palm OS Cobalt supports IEEE-754 single and double precision floating-point numbers declared with the C types `float` and `double`. Numbers of type `float` occupy four bytes and have an effective range of 1.17549e-38 to 3.40282e+38. Numbers of type `double` occupy eight bytes and have an effective range of 2.22507e-308 to 1.79769e+308. Limited operations are also permitted on values of type **long double**.

You can use basic arithmetic operations to add, subtract, multiply, and divide numbers of type `float` and `double`. Higher-level functions such as those in the standard C header file `math.h` are not part of the core OS; you must either write them yourself, or you must employ a third-party math library.

The standard IEEE-754 special “non-number” values of NaN (not a number), +INF (positive infinity), and -INF (negative infinity) are generated as appropriate if you perform an operation that produces a result outside the range of numbers that can be represented. For instance, dividing a positive number by 0 returns +INF.

The Float Manager contains functions that convert double-precision floating-point numbers to and from various other formats: 32- and 64-bit signed and unsigned integers, as well as floating point values of differing lengths. It also provides a set of basic comparison functions that can be used with values of type `double` and `float`. Finally, it supplies a set of functions that you can use to add, subtract, multiply, and divide values of type `float` and `double`.

Two of the supplied mathematical operations are special: [`FlpCorrectedAdd\(\)`](#) and [`FlpCorrectedSub\(\)`](#) let you add or subtract double values, optionally correcting for least-significant-bit errors when the result should be zero but instead is very close to zero.

In the rare event that you need to work with the binary representation of a `double`, you can use [`FlpBase10Info\(\)`](#) to obtain the mantissa, exponent, and sign (all in base 10) of a `double`

## Floating Point

### *Summary of Float Manager*

---

value. If you only need the exponent, you can use [FlpGetExponent\(\)](#) instead.

See [Chapter 28, “Float Manager,”](#) on page 287 for complete reference information on the functions and macros that make up the Float Manager.

## Summary of Float Manager

---

### Mathematical Operations

---

[FlpAddDouble\(\)](#)

[FlpAddFloat\(\)](#)

[FlpCorrectedAdd\(\)](#)

[FlpCorrectedSub\(\)](#)

[FlpDivDouble\(\)](#)

[FlpDivFloat\(\)](#)

[FlpMulDouble\(\)](#)

[FlpMulFloat\(\)](#)

[FlpNegDouble\(\)](#)

[FlpNegFloat\(\)](#)

[FlpSubDouble\(\)](#)

[FlpSubFloat\(\)](#)

---

### Deconstructing Values

---

[FlpBase10Info\(\)](#)

[FlpGetExponent\(\)](#)

---

### Comparing Values

---

[FlpCompareDoubleEqual\(\)](#)

[FlpCompareDoubleLessThan\(\)](#)

[FlpCompareDoubleLessThanOrEqual\(\)](#)

[FlpCompareFloatEqual\(\)](#)

[FlpCompareFloatLessThan\(\)](#)

[FlpCompareFloatLessThanOrEqual\(\)](#)

---

---

**Converting to and from Float and Double**

---

<a href="#"><u>FlpDoubleToFloat()</u></a>	<a href="#"><u>FlpDoubleToInt32()</u></a>
<a href="#"><u>FlpDoubleToLongDouble()</u></a>	<a href="#"><u>FlpDoubleToLongLong()</u></a>
<a href="#"><u>FlpDoubleToUInt32()</u></a>	<a href="#"><u>FlpDoubleToULongLong()</u></a>
<a href="#"><u>FlpFloatToDouble()</u></a>	<a href="#"><u>FlpFloatToInt32()</u></a>
<a href="#"><u>FlpFloatToLongDouble()</u></a>	<a href="#"><u>FlpFloatToLongLong()</u></a>
<a href="#"><u>FlpFloatToUInt32()</u></a>	<a href="#"><u>FlpFloatToULongLong()</u></a>
<a href="#"><u>FlpInt32ToDouble()</u></a>	<a href="#"><u>FlpInt32ToFloat()</u></a>
<a href="#"><u>FlpLongDoubleToDouble()</u></a>	<a href="#"><u>FlpLongDoubleToFloat()</u></a>
<a href="#"><u>FlpLongLongToDouble()</u></a>	<a href="#"><u>FlpLongLongToFloat()</u></a>
<a href="#"><u>FlpUInt32ToDouble()</u></a>	<a href="#"><u>FlpUInt32ToFloat()</u></a>
<a href="#"><u>FlpULongLongToDouble()</u></a>	<a href="#"><u>FlpULongLongToFloat()</u></a>

---

## **Floating Point**

*Summary of Float Manager*

---

# Debugging Strategies

---

You can use the Palm OS® Error Manager to display debugging information and unexpected runtime errors such as those that typically show up during program development. Final versions of applications or system software won't display the debugging information.

The Error Manager API consists of a set of functions for displaying an alert with an error message, file name, and the line number where the error occurred. If a debugger is connected, it is entered when the error occurs.

Palm OS also provides a **try-and-catch** exception-handling mechanism that applications can use for handling such runtime errors as out of memory conditions, user input errors, and the like.

This section discusses the following topics:

- [Displaying Development Errors](#)
- [The Try-and-Catch Mechanism](#)
- [Summary of Debugging API](#)

This chapter only describes programmatic debugging strategies; to learn how to use the available tools to debug your application, see the book *Palm OS Programming Development Tools Guide*.

## Displaying Development Errors

The Error Manager provides a single function, [ErrFatalErrorInContext\(\)](#), that displays an alert containing a supplied file name, a line number, and a string. You can insert calls to this function into your code in order to alert you to erroneous situations.

Often, you'll want to display certain error messages when you are debugging your code that should never appear in the production version of your program. The Error Manager defines a set of macros that each call `ErrFatalErrorInContext()` and that are conditional, depending on the build type.

## Setting the Build Type

The Error Manager uses the compiler define `BUILD_TYPE` to control the set of error messages displayed. You can set the value of the compiler define to control which level of error checking and display is compiled into the application. Two `BUILD_TYPE` levels are defined in Palm OS Cobalt:

---

<b>If you set <code>BUILD_TYPE</code> to...</b>	<b>The compiler...</b>
<code>BUILD_TYPE_RELEASE</code>	Compiles in only <code>ErrFatalError()</code> and <code>ErrFatalErrorIf()</code> macros.
<code>BUILD_TYPE_DEBUG</code>	Compiles in <code>ErrFatalError()</code> , <code>ErrFatalErrorIf()</code> , <code>DbgOnlyFatalError()</code> , and <code>DbgOnlyFatalErrorIf()</code> macros.

---

During development, it makes sense to set `BUILD_TYPE` to `BUILD_TYPE_DEBUG`. At this level, all Error Manager messages are displayed. Then, when you are ready to release your application, set `BUILD_TYPE` to `BUILD_TYPE_RELEASE`.

---

**NOTE:** Because the `ErrFatalError()` and `ErrFatalErrorIf()` macros display a message regardless of the build type, they should only be used for those unrecoverable situations that can be communicated to the end user. For messages that aid in the debugging process but aren't to be presented to the end user, use `DbgOnlyFatalError()` and `DbgOnlyFatalErrorIf()` instead.

---

## Displaying Error Messages

Rather than calling `ErrFatalErrorInContext()`, most applications make use of the Error Manager's compiler macros instead. These macros employ `ErrFatalErrorInContext()` to display a fatal alert dialog on the screen. There are four macros: `ErrFatalError()`, `ErrFatalErrorIf()`, `DbgOnlyFatalError()`, and `DbgOnlyFatalErrorIf()`.

- `ErrFatalError()` and `DbgOnlyFatalError()` always display the error message on the screen if `BUILD_TYPE` is `BUILD_TYPE_DEBUG`. `DbgOnlyFatalError()` does nothing if `BUILD_TYPE` is `BUILD_TYPE_RELEASE`.
- `ErrFatalErrorIf()` and `DbgOnlyFatalErrorIf()` display the error message only if their first argument is `true` and if `BUILD_TYPE` is `BUILD_TYPE_DEBUG`. `DbgOnlyFatalErrorIf()` does nothing if `BUILD_TYPE` is `BUILD_TYPE_RELEASE`.

The first argument to `ErrFatalErrorIf()` and `DbgOnlyFatalErrorIf()` is a Boolean value that controls whether or not the message (the second parameter) is displayed. Typically, the Boolean parameter is an in-line expression that evaluates to `true` if there is an error condition. As a result, both the expression that evaluates the error condition and the message text are left out of the compiled code when error checking is turned off. You can use either `ErrFatalErrorIf()` or `ErrFatalError()`, but using `ErrFatalErrorIf()` makes your source code cleaner. For example, assume your source code looks like this:

---

```
result = DoSomething();
ErrFatalErrorIf (result < 0,
    "unexpected result from DoSomething");
```

---

With error checking turned on, this code displays an error alert dialog if the result from `DoSomething()` is less than 0. With error checking turned off, both the expression evaluation `result < 0` and the error message text are left out of the compiled code.

The same net result can be achieved by the following code:

---

```
result = DoSomething();
#if BUILD_TYPE != BUILD_TYPE_RELEASE
if (result < 0)
```

---

```
    ErrFatalError ("unexpected result from DoSomething");  
#endif
```

---

However, this solution is longer and requires more work than simply calling `ErrFatalErrorIf()`. It also makes the source code harder to follow.

## The Try-and-Catch Mechanism

The operating system is aware of the machine state of the Palm Powered™ device and can therefore correctly save and restore this state. The built-in try-and-catch of the compiler can't be used because it's machine dependent.

Try-and-catch is basically a neater way of implementing a goto if an error occurs. A typical way of handling errors in the middle of a function is to go to the end of the function as soon as an error occurs and have some general-purpose cleanup code at the end of every function. Errors in nested functions are even trickier because the result code from every subroutine call must be checked before continuing.

When you use the `ErrTry()` and `ErrCatch()` macros, you are providing the compiler with a place to jump to when an error occurs. You can go to that error handling routine at any time by calling `ErrThrow()`. When the compiler sees the `ErrThrow()` call, it performs a goto<sup>1</sup> to your error handling code. The greatest advantage to calling `ErrThrow()`, however, is for handling errors in nested subroutine calls.

Even if `ErrThrow()` is called from a nested subroutine, execution immediately goes to the same error handling code in the higher-level call. The compiler and runtime environment automatically strip off the stack frames that were pushed onto the stack during the nesting process and go to the error-handling section of the higher-level call. You no longer have to check for result codes after calling every subroutine; this can greatly simplify your source code and reduce its size.

---

1. longjmp, actually.

## Using the Try-and-Catch Mechanism

[Listing 13.1](#) illustrates the possible layout for a typical routine using the try-and-catch mechanism.

### Listing 13.1 Try-and-Catch Mechanism Example

---

```
ErrTry {
    p = MemPtrNew(1000);
    if (!p) ErrThrow(errNoMemory);
    MemSet(p, 1000, 0);
    CreateTable(p);
    PrintTable(p);
}

ErrCatch(err) {
    // Recover or cleanup after a failure in the
    // above Try block."err" is an int
    // identifying the reason for the failure.

    // You may call ErrThrow() if you want to
    // jump out to the next Catch block.

    // The code in this Catch block doesn't
    // execute if the above Try block completes
    // without a Throw.

    if (err == errNoMemory)
        ErrFatalError("Out of Memory");
    else
        ErrFatalError("Some other error");
} ErrEndCatch
// You must structure your code exactly as
// above. You can't have an ErrTry without an
//ErrCatch { } ErrEndCatch, or vice versa.
```

---

Any call to [ErrThrow\(\)](#) within the [ErrTry\(\)](#) block results in control passing immediately to the [ErrCatch\(\)](#) block. Even if the subroutine `CreateTable()` called `ErrThrow()`, control would pass directly to the `ErrCatch()` block. If the `ErrTry()` block completes without calling `ErrThrow()`, the `ErrCatch()` block is not executed.

## Debugging Strategies

### Summary of Debugging API

---

You can nest multiple `ErrTry()` blocks. For example, if you wanted to perform some cleanup at the end of `CreateTable()` in case of error,

- Put `ErrTry()` and `ErrCatch()` blocks in `CreateTable()`.
- Clean up in the `ErrCatch()` block first.
- Call [`ErrThrow\(\)`](#) to jump to the top-level `ErrCatch()`.

## Summary of Debugging API

---

### Debugging Functions

---

#### Displaying Errors

[`DbgOnlyFatalError\(\)`](#)

[`DbgOnlyFatalErrorIf\(\)`](#)

[`ErrDisplay\(\)`](#)

[`ErrDisplayFileLineMsg\(\)`](#)

[`ErrFatalDisplay\(\)`](#)

[`ErrFatalDisplayIf\(\)`](#)

[`ErrFatalError\(\)`](#)

[`ErrFatalErrorIf\(\)`](#)

[`ErrFatalErrorInContext\(\)`](#)

[`ErrNonFatalDisplay\(\)`](#)

[`ErrNonFatalDisplayIf\(\)`](#)

#### Throwing and Catching Exceptions

[`ErrCatch\(\)`](#)

[`ErrCatchWithAddress\(\)`](#)

[`ErrEndCatch\(\)`](#)

[`ErrExceptionListAppend\(\)`](#)

[`ErrExceptionListGetByThreadID\(\)`](#)

[`ErrExceptionListRemove\(\)`](#)

[`ErrLongJump\(\)`](#)

[`ErrSetJump\(\)`](#)

[`ErrThrow\(\)`](#)

[`ErrThrowIf\(\)`](#)

[`ErrThrowWithAddress\(\)`](#)

[`ErrThrowWithHandler\(\)`](#)

[`ErrTry\(\)`](#)

---



# Part II

# Reference

This part contains reference documentation for the following:

<a href="#">Alarm Manager</a>	121
<a href="#">Attention Manager</a>	129
<a href="#">Category Manager Sync</a>	161
<a href="#">Common Battery Types</a>	165
<a href="#">Common Error Codes</a>	169
<a href="#">Cyclic Redundancy Check</a>	183
<a href="#">DateTime</a>	187
<a href="#">Debug Manager</a>	219
<a href="#">Desktop Link Server</a>	227
<a href="#">Error Manager</a>	237
<a href="#">ErrTryCatch</a>	247
<a href="#">Expansion Manager</a>	257
<a href="#">Fatal Alert</a>	277
<a href="#">Feature Manager</a>	279
<a href="#">Float Manager</a>	287
<a href="#">Host Control</a>	309
<a href="#">Loader</a>	379
<a href="#">Patch</a>	397
<a href="#">PerfDriver</a>	401
<a href="#">Preferences</a>	405
<a href="#">Sync Manager</a>	421
<a href="#">System Manager</a>	427
<a href="#">SysThread</a>	451
<a href="#">System Utilities</a>	485
<a href="#">Time Manager</a>	505
<a href="#">Trace Manager</a>	509



# Alarm Manager

---

This chapter provides reference material for the Alarm Manager. It is organized into the following sections:

<a href="#">Alarm Manager Structures and Types</a>	122
<a href="#">Alarm Manager Constants</a>	124
<a href="#">Alarm Manager Launch Codes</a>	125
<a href="#">Alarm Manager Functions and Macros</a>	126

The header file `AlarmMgr.h` declares the API that this chapter describes.

You often use the Alarm Manager in conjunction with the [Attention Manager](#). For usage information on both, see [Chapter 1, “Attentions and Alarms,”](#) on page 3.

# Alarm Manager Structures and Types

## AlmProcCmdEnum Typedef

<b>Purpose</b>	
<b>Declared In</b>	AlarmMgr.h
<b>Prototype</b>	typedef Enum16 AlmProcCmdEnum

## SysAlarmTriggeredParamType Struct

<b>Purpose</b>	Contains parameters that accompany a <a href="#">sysAppLaunchCmdAlarmTriggered</a> launch code.
<b>Declared In</b>	AlarmMgr.h
<b>Prototype</b>	<pre>typedef struct SysAlarmTriggeredParamType {     uint32_t ref;     uint32_t alarmSeconds;     Boolean purgeAlarm;     uint8_t padding;     uint16_t padding1; } SysAlarmTriggeredParamType</pre>
<b>Fields</b>	<p><b>ref</b> The caller-defined value specified when the alarm was set with <a href="#">AlmSetAlarm()</a>.</p> <p><b>alarmSeconds</b> The date/time specified when the alarm was set with <a href="#">AlmSetAlarm()</a>. The value is given as the number of seconds since 1/1/1904.</p> <p><b>purgeAlarm</b> In your launch code handler, set this field to <code>true</code> if the alarm should be removed from the alarm table. Use this as an optimization to prevent the application from receiving <a href="#">sysAppLaunchCmdDisplayAlarm</a> if you don't wish to perform any other processing for this alarm. If you do want to receive the launch code, set this field to <code>false</code>.</p> <p><b>padding</b> Padding bytes - not used.</p>

padding1  
Padding bytes - not used.

## **SysDisplayAlarmParamType Struct**

**Purpose** Contains parameters that accompany a [sysAppLaunchCmdDisplayAlarm](#) launch code.

**Declared In** AlarmMgr.h

**Prototype**

```
typedef struct SysDisplayAlarmParamType {  
    uint32_t ref;  
    uint32_t alarmSeconds;  
    Boolean soundAlarm;  
    uint8_t padding;  
    uint16_t padding1;  
} SysDisplayAlarmParamType
```

**Fields**

**ref**  
The caller-defined value specified when the alarm was set with [AlmSetAlarm\(\)](#).

**alarmSeconds**  
The date/time specified when the alarm was set with [AlmSetAlarm\(\)](#). The value is given as the number of seconds since 1/1/1904.

**soundAlarm**  
true if the alarm should be sounded, false otherwise. *This value is currently not used.*

**padding**  
Padding bytes - not used.

**padding1**  
Padding bytes - not used.

# Alarm Manager Constants

## Alarm Manager Error Codes

<b>Purpose</b>	Error codes returned by the various Alarm Manager functions.
<b>Declared In</b>	AlarmMgr.h
<b>Constants</b>	<pre>#define almErrFull (almErrorClass   2)     The alarm table is full.  #define almErrMemory (almErrorClass   1)     There is insufficient memory to perform the requested     operation.</pre>

## AlmProcCmdEnumTag Enum

<b>Purpose</b>	
<b>Declared In</b>	AlarmMgr.h
<b>Constants</b>	<pre>almProcCmdTriggered = 0  almProcCmdReschedule  almProcCmdCustom = 0x8000</pre>

## Alarm Manager Launch Codes

### **sysAppLaunchCmdAlarmTriggered**

<b>Purpose</b>	Indicates that the application should perform a quick action such as scheduling the next alarm or sounding the alarm.
<b>Declared In</b>	<code>CmnLaunchCodes.h</code>
<b>Prototype</b>	<code>#define sysAppLaunchCmdAlarmTriggered 6</code>
<b>Parameters</b>	The launch code's parameter block pointer references a <a href="#">SysAlarmTriggeredParamType</a> structure.
<b>Comments</b>	This launch code is sent as close to the actual alarm time as possible.  Upon receiving this launch code, an application may perform any quick, non-blocking action. The action should be quick: multiple alarms may be pending at the same time for multiple applications, and one alarm display shouldn't block the system and prevent other applications from receiving their alarms in a timely fashion. More time-consuming actions should be performed upon receipt of a <a href="#">sysAppLaunchCmdDisplayAlarm</a> launch code.

### **sysAppLaunchCmdDisplayAlarm**

<b>Purpose</b>	Indicates that the application can perform full, possibly blocking, handling of the alarm.
<b>Declared In</b>	<code>CmnLaunchCodes.h</code>
<b>Prototype</b>	<code>#define sysAppLaunchCmdDisplayAlarm 7</code>
<b>Parameters</b>	The launch code's parameter block pointer references a <a href="#">SysDisplayAlarmParamType</a> structure.
<b>Comments</b>	This is the application's opportunity to handle an alarm in a lengthy or blocking fashion. Alert dialogs are usually displayed when this launch code is received. This work should be done here, not when <a href="#">sysAppLaunchCmdAlarmTriggered</a> is received.

# Alarm Manager Functions and Macros

## AlmEnableNotification Function

**Purpose**  
**Declared In** AlarmMgr.h  
**Prototype** void AlmEnableNotification (Boolean *enable*)  
**Parameters** → *enable*

**Returns**

## AlmGetAlarm Function

**Purpose** Return the date and time for the application's alarm, if one is set.  
**Declared In** AlarmMgr.h  
**Prototype** uint32\_t AlmGetAlarm (DatabaseID *dbID*,  
uint32\_t \**refP*)  
**Parameters** → *dbID*  
Local ID of the application.  
← *refP*  
The alarm's reference value is returned here. This value is passed to the application when the alarm is triggered.  
**Returns** The date and time the alarm will trigger, given in seconds since 1/1/1904; if no alarm is active for the application, 0 is returned for the alarm seconds and the reference value is undefined.  
**See Also** [AlmSetAlarm\(\)](#)

## AlmSetAlarm Function

<b>Purpose</b>	Set or cancel an alarm for the given application.
<b>Declared In</b>	AlarmMgr.h
<b>Prototype</b>	<pre>status_t AlmSetAlarm (DatabaseID dbID,                       uint32_t ref, uint32_t alarmSeconds,                       Boolean quiet)</pre>
<b>Parameters</b>	<p>→ <i>dbID</i> Local ID of the application.</p> <p>→ <i>ref</i> Caller-defined value. This value is passed with the launch code that notifies the application that the alarm has been triggered.</p> <p>→ <i>alarmSeconds</i> Alarm date/time in seconds since 1/1/1904, or 0 to cancel the current alarm (if any).</p> <p>→ <i>quiet</i> Reserved for future use. This value is not currently used.</p>
<b>Returns</b>	Returns <code>errNone</code> if the operation was completed successfully, or one of the following otherwise:  <code>almErrMemory</code> Insufficient memory.  <code>almErrFull</code> Alarm table is full.
<b>Comments</b>	<p>This function sets an alarm for the specified application. An application can have only one alarm set at a time. If an alarm for this application has already been set, it is replaced with the new alarm.</p> <p>The <code>alarmSeconds</code> parameter specifies the time at which the alarm will be triggered. As soon as possible after this time, the alarm manager sends the <a href="#">sysAppLaunchCmdAlarmTriggered</a> launch code to the specified application. If there is another alarm that should be set for this application, you can set it in response to this launch code. Following the <code>sysAppLaunchCmdAlarmTriggered</code> launch code, the alarm manager sends a <a href="#">sysAppLaunchCmdDisplayAlarm</a> launch code. This is where your application should do things such as display a modal dialog indicating that the event has occurred.</p>

## Alarm Manager

*AlmSetAlarm*

---

If your application needs access to any particular value to respond to the alarm, pass a pointer to that value in the `ref` parameter. The system will pass this pointer back to the application in the launch codes' parameter blocks.

**See Also** [AlmGetAlarm\(\)](#)

# Attention Manager

---

This chapter provides reference documentation for the Attention Manager. It is divided into the following sections:

<a href="#">Attention Manager Structures and Types</a> . . . . .	129
<a href="#">Attention Manager Constants</a> . . . . .	138
<a href="#">Attention Manager Launch Codes</a> . . . . .	145
<a href="#">Attention Manager Functions and Macros</a> . . . . .	145

The header file `AttentionMgr.h` declares the API that this chapter describes.

For more information about the attention manager, see [Chapter 1, “Attentions and Alarms,”](#) on page 3.

## Attention Manager Structures and Types

### AttnCommandArgsDrawDetailTag Struct

**Purpose** When `kAttnCommandDrawDetail` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, the application needs to draw the detailed contents of the attention slip. The `AttnCommandArgsDrawDetailTag` structure accompanies the `kAttnCommandDrawDetail` command, and provides the information needed to draw the contents of that slip.

---

**NOTE:** This structure is declared as the `drawDetail` member of the [AttnCommandArgsType](#) union.

---

## Attention Manager

*AttnCommandArgsDrawListTag*

---

**Declared In** AttentionMgr.h

**Prototype**

```
struct AttnCommandArgsDrawDetailTag {
    RectangleType bounds;
    Boolean firstTime;
    uint8_t padding1;
    uint16_t padding2;
    AttnFlagsType flags;
} drawDetail;
```

**Fields** bounds

Contains the window-relative bounding box for the area to draw. The clipping region is also set to the dimensions of this box to prevent accidental drawing outside.

firstTime

Not used in Palm OS Cobalt.

padding1

Padding bytes.

padding2

Padding bytes.

flags

The global user preferences for this attention attempt combined with the custom flags passed in by the developer. Only the lower 16 bits of this field have meaning; use `kAttnFlagsSoundBit`, `kAttnFlagsLEDBit`, `kCustomFlagsVibrateBit`, and `kAttnFlagsCustomEffectBit` (described under "[AttnFlagsType](#)" on page 135) to interpret the contents of this field.

### AttnCommandArgsDrawListTag Struct

**Purpose** When `kAttnCommandDrawList` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, the application needs to draw the appropriate list item in the attention slip. The `AttnCommandArgsDrawListTag` structure accompanies the `kAttnCommandDrawList` command,

and provides the information needed to draw the contents of that slip.

---

**NOTE:** This structure is declared as the `drawList` member of the [AttnCommandArgsType](#) union.

---

<b>Declared In</b>	AttentionMgr.h
<b>Prototype</b>	<pre>struct AttnCommandArgsDrawListTag {     RectangleType bounds;     Boolean firstTime;     uint8_t padding1;     uint16_t padding2;     AttnFlagsType flags;     Boolean selected;     uint8_t padding3;     uint16_t padding4; } drawList;</pre>
<b>Fields</b>	<p><b>bounds</b> Contains the window-relative bounding box for the area to draw. The clipping region is also set to the dimensions of this box to prevent accidental drawing outside.</p> <p><b>firstTime</b> Not used in Palm OS Cobalt.</p> <p><b>padding1</b> Padding bytes.</p> <p><b>padding2</b> Padding bytes.</p> <p><b>flags</b> The global user preferences for this attention attempt combined with the custom flags passed in by the developer. Only the lower 16 bits of this field have meaning; use <code>kAttnFlagsSoundBit</code>, <code>kAttnFlagsLEDBit</code>, <code>kCustomFlagsVibrateBit</code>, and <code>kAttnFlagsCustomEffectBit</code> (described under "<a href="#">AttnFlagsType</a>" on page 135) to interpret the contents of this field.</p> <p><b>selected</b> Not used in Palm OS Cobalt.</p>

## Attention Manager

### *AttnCommandArgsGotItTag*

---

padding3  
Padding bytes.

padding4  
Padding bytes.

## AttnCommandArgsGotItTag Struct

**Purpose** When `kAttnCommandGotIt` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, it is accompanied by an `AttnCommandArgsGotItTag` structure. This structure indicates whether the `kAttnCommandGotIt` command was generated because the user dismissed the attention, or whether the system is simply informing your application that [AttnForgetIt\(\)](#) was called. Your application normally ignores the latter case if your application made the call to `AttnForgetIt()`.

---

**NOTE:** This structure is declared as the `gotIt` member of the [AttnCommandArgsType](#) union.

---

**Declared In** `AttentionMgr.h`

**Prototype**

```
struct AttnCommandArgsGotItTag {  
    Boolean dismissedByUser;  
} gotIt;
```

**Fields** `dismissedByUser`  
`true` indicates that the user dismissed the attention. `false` indicates that the `kAttnCommandGotIt` command was generated by a call to [AttnForgetIt\(\)](#).

## AttnCommandArgsIterateTag Struct

**Purpose** When `kAttnCommandIterate` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, it is accompanied by an `AttnCommandArgsIterateTag` structure. This structure contains any necessary data that the application may need in order to process the launch code.

---

**NOTE:** This structure is declared as the `iterate` member of the [AttnCommandArgsType](#) union.

---

**Declared In** `AttentionMgr.h`

**Prototype**

```
struct AttnCommandArgsIterateTag {  
    uint32_t iterationData;  
} iterate;
```

**Fields** `iterationData`  
Any necessary data that the application may need in order to process the launch code. The value of this field is that which was originally passed to [AttnIterate\(\)](#).

## AttnCommandArgsType Struct

**Purpose** The `AttnCommandArgsType` structure is a union of C structures. How you interpret the union's contents depends on which command it accompanies. Not all commands are accompanied by an `AttnCommandArgsType` structure, as listed in the following table:

---

<b>AttnCommandType</b>	<b>Accompanied By</b>
<code>kAttnCommandDrawDetail</code>	<a href="#">AttnCommandArgsDrawDetailTag</a>
<code>kAttnCommandDrawList</code>	<a href="#">AttnCommandArgsDrawListTag</a>
<code>kAttnCommandPlaySound</code>	nothing
<code>kAttnCommandCustomEffect</code>	nothing
<code>kAttnCommandGoThere</code>	nothing
<code>kAttnCommandGotIt</code>	<a href="#">AttnCommandArgsGotItTag</a>
<code>kAttnCommandSnooze</code>	nothing
<code>kAttnCommandIterate</code>	<a href="#">AttnCommandArgsIterateTag</a>

---

## Attention Manager

*AttnCommandArgsType*

---

**Declared In**    `AttentionMgr.h`

**Prototype**

```
typedef union AttnCommandArgsTag {
    struct AttnCommandArgsDrawDetailTag {
        ...
    } drawDetail;
    struct AttnCommandArgsDrawListTag {
        ...
    } drawList;
    struct AttnCommandArgsGotItTag {
        ...
    } gotIt;
    struct AttnCommandArgsIterateTag {
        ...
    } iterate;
} AttnCommandArgsType
```

**Fields**        `drawDetail`

When `kAttnCommandDrawDetail` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, the application needs to draw the detailed contents of the attention slip. The [AttnCommandArgsDrawDetailTag](#) structure accompanies the `kAttnCommandDrawDetail` command, and provides the information needed to draw the contents of that slip.

`drawList`

When `kAttnCommandDrawList` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, the application needs to draw the appropriate list item in the attention slip. The [AttnCommandArgsDrawListTag](#) structure accompanies the `kAttnCommandDrawList` command, and provides the information needed to draw the contents of that slip.

`gotIt`

When `kAttnCommandGotIt` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, it is accompanied by an [AttnCommandArgsGotItTag](#) structure. This structure indicates whether the `kAttnCommandGotIt` command was generated because the user dismissed the attention, or whether the system is simply informing your application that [AttnForgetIt\(\)](#) was called. Your

application normally ignores the latter case if your application made the call to `AttnForgetIt()`.

#### `iterate`

When `kAttnCommandIterate` is passed to the application as a parameter accompanying the [sysAppLaunchCmdAttention](#) launch code, it is accompanied by an [AttnCommandArgsIterateTag](#) structure. This structure contains any necessary data that the application may need in order to process the launch code.

## **AttnCommandType Typedef**

<b>Purpose</b>	The <code>AttnCommandType</code> typedef specifies the set of possible commands that can be sent to the application requesting the user's attention, as one of the arguments that accompanies a <a href="#">sysAppLaunchCmdAttention</a> launch code.
<b>Declared In</b>	<code>AttentionMgr.h</code>
<b>Prototype</b>	<code>typedef uint16_t AttnCommandType</code>
<b>Comments</b>	See " <a href="#">Commands</a> " on page 143 for the set of values that <code>AttnCommandType</code> can assume.

## **AttnFlagsType Typedef**

<b>Purpose</b>	Pass a value of this type to <a href="#">AttnGetAttention()</a> and <a href="#">AttnUpdate()</a> to specify what means the device should or should not use to get the user's attention. A value of this type is also passed back to your code as part of the structure passed with the <a href="#">sysAppLaunchCmdAttention</a> launch code.
<b>Declared In</b>	<code>AttentionMgr.h</code>
<b>Prototype</b>	<code>typedef uint32_t AttnFlagsType</code>
<b>Comments</b>	See " <a href="#">Attention Flags</a> " on page 141 and " <a href="#">Attention Override Flags</a> " on page 142 for the set of values that this type can assume. Note that more bits may be defined if necessary to accommodate additional hardware.

## Attention Manager

### *AttnLaunchCodeArgsType*

---

These constants can be used in combination. For example, to disable both sound and the LED, use

```
kAttnFlagsNoSound | kAttnFlagsNoLED.
```

If neither `kAttnFlagsAlwaysSound` nor `kAttnFlagsNoSound` is set for a given attention item, a sound plays if and only if the user's preference is to play a sound and the user's preference for alarm volume is non-zero.

## AttnLaunchCodeArgsType Struct

**Purpose** If the Attention Manager needs your code to draw the details of your attention in the attention slip or perform another attention-specific function, it sends a [sysAppLaunchCmdAttention](#) launch code to your application. Along with the launch code, it passes a pointer to the following structure, which indicates both what your code is expected to do and identifies the attention that triggered the launch code:

**Declared In** `AttentionMgr.h`

**Prototype**

```
typedef struct {
    AttnCommandType command;
    uint16_t padding;
    uint32_t userData;
    AttnCommandArgsType *commandArgsP;
} AttnLaunchCodeArgsType
```

**Fields**

- `command`  
Indicates what your code is being requested to do. The complete list of possible commands are described in the definition of [AttnCommandType](#).
- `padding`  
Padding bytes.
- `userData`  
Identifier that distinguishes the particular attention item from others made by this application. This identifier was specified when the attention item was created.

`commandArgsP`

Pointer to command-specific arguments. See the description of each command for a discussion of that command's arguments.

**Comments** When processing the launch code be aware that your application doesn't have application globals available to it; it is important that anything necessary to draw or otherwise display be available through `commandArgsP`.

## **AttnLevelType Typedef**

**Purpose** Attention attempts can either be insistent or subtle. Insistent attention attempts make a serious effort to get the user's attention, by both displaying a slip and possibly by triggering one or more special effects, such as blinking a light, vibrating, or playing a sound. Other alerts are of a less serious nature and shouldn't disrupt the user. Consequently, subtle attention attempts typically present the attention indicator and may trigger one or more special effects, but don't display the Attention Manager slip. The user can then work until a suitable time, at which point they can tap on the indicator to see what needs their attention. Subtle attention attempts might be used for telling the user that they have new e-mail, or perhaps that a holiday or birthday is coming up.

**Declared In** `AttentionMgr.h`

**Prototype** `typedef uint16_t AttnLevelType`

**Comments** "[Attention Levels](#)" on page 144 defines the set of values that this type can assume.

User preferences for the various special effects can't be set separately for subtle and insistent attention attempts. If your application needs to vary the effects based upon the `AttnLevelType`, pass a suitable value for the `flags` parameter in your [AttnGetAttention\(\)](#) call.

## Attention Manager

*AttnNotifyDetailsType*

---

### AttnNotifyDetailsType Struct

<b>Purpose</b>	Structure that accompanied an obsolete Attention Manager notification. Not used in Palm OS Cobalt.
<b>Declared In</b>	<code>AttentionMgr.h</code>
<b>Prototype</b>	<pre>typedef struct {     AttnFlagsType flags; } AttnNotifyDetailsType</pre>

## Attention Manager Constants

### Attention Manager Error Codes

<b>Purpose</b>	Error codes returned by the various Attention Manager functions.
<b>Declared In</b>	<code>AttentionMgr.h</code>
<b>Constants</b>	<pre>#define attnErrMemory (attnErrorClass   1)     Returned by <a href="#">AttnGetAttention()</a> when there is     insufficient memory to perform the requested operation.  #define attnErrItemNotFound (attnErrorClass   2)     The specified attention item is not one of the items being     handled by the Attention Manager. It may have been deleted.</pre>

### Drawing Constants

<b>Purpose</b>	Define the on-screen boundaries of the attention indicator that is displayed when the status bar is not visible and the Attention Manager needs to get the user's attention, and specify how an application should format the list view.
<b>Declared In</b>	<code>AttentionMgr.h</code>
<b>Constants</b>	<pre>#define kAttnIndicatorHeight 15     The height of the attention indicator.  #define kAttnIndicatorLeft 0     The left-hand edge of the attention indicator.  #define kAttnIndicatorTop 0     The top-most edge of the attention indicator.</pre>

```
#define kAttnIndicatorWidth 16
```

The width of the attention indicator.

```
#define kAttnListMaxIconWidth 15
```

Maximum width of the application's icon. If the icon is narrower than this, it should be drawn centered within this width.

```
#define kAttnListTextOffset 17
```

Offset, from the left-hand edge of the drawing bounds, of the textual description of the attention.

**Comments** The `kAttnList...` constants are used when drawing the list view. Applications should use these constants to format the display of information in the Attention Manager's list view. Draw the application's small icon centered within the first `kAttnListMaxIconWidth` pixels of the drawing bounds. Then draw two lines of text describing the attention, left-justified, starting at `kAttnListTextOffset` from the left edge of the drawing bounds.

## Feature Constants

**Purpose** The Attention Manager defines a read-only feature ('attn', 0) that indicates the current user settings and capabilities of the hardware. The upper 16 bits of the feature indicate whether or not the hardware has the capability to perform that sort of alert. The lower 16 bits indicate whether the user has currently enabled that sort of alert.

**Declared In** `AttentionMgr.h`

**Constants**

```
#define kAttnFtrCapabilities 0
```

 Attention Manager feature number.

```
#define kAttnFtrCreator 'attn'
```

 Attention Manager feature creator.

**See Also** [Feature Masks](#), [Feature Flags](#)

## Feature Masks

- Purpose** When working with the value obtained with `FtrGet`, use these constants to separate those bits that contain the user settings from those bits that identify the device's capabilities.
- Declared In** `AttentionMgr.h`
- Constants**
- ```
#define kAttnFlagsCapabilitiesMask  
    (kAttnFlagsAllBits<<16)  
    Mask to isolate those bits that contain the device capabilities.  
  
#define kAttnFlagsUserSettingsMask  
    (kAttnFlagsAllBits)  
    Mask to isolate those bits that contain the user settings.
```
- See Also** [Feature Constants](#), [Feature Flags](#)

## Feature Flags

- Purpose** These constants can be used to interpret the device capabilities (`kAttnFlagsHas...`) and the user settings (`kAttnFlagsUserWants...`), obtained from `FtrGet` (see "[Feature Constants](#)" on page 139)
- Declared In** `AttentionMgr.h`
- Constants**
- ```
#define kAttnFlagsHasCustomEffect  
    (kAttnFlagsCustomEffectBit<<16)  
    Not used.  
  
#define kAttnFlagsHasLED (kAttnFlagsLEDBit<<16)  
    The device has an LED that can be illuminated to indicate an  
    alert.  
  
#define kAttnFlagsHasSound  
    (kAttnFlagsSoundBit<<16)  
    The device is capable of playing a sound to indicate an alert.  
  
#define kAttnFlagsHasVibrate  
    (kAttnFlagsVibrateBit<<16)  
    The device is capable of vibrating to indicate an alert.  
  
#define kAttnFlagsUserWantsCustomEffect  
    (kAttnFlagsCustomEffectBit)  
    Not used.
```

```
#define kAttnFlagsUserWantsLED (kAttnFlagsLEDBit)  
    The user wants the LED illuminated to signal an alert.
```

```
#define kAttnFlagsUserWantsSound  
    (kAttnFlagsSoundBit)  
    The user wants a sound played to signal an alert.
```

```
#define kAttnFlagsUserWantsVibrate  
    (kAttnFlagsVibrateBit)  
    The user wants the device to vibrate to signal an alert.
```

**See Also** [Feature Constants](#), [Feature Masks](#)

## Attention Flags

**Purpose** Specify what means the device should or should not use to get the user's attention

**Declared In** `AttentionMgr.h`

**Constants**

```
#define kAttnFlagsAllBits ((AttnFlagsType)0xFFFF)  
    Uses all available means to get the user's attention.
```

```
#define kAttnFlagsCustomEffectBit  
    ((AttnFlagsType)0x0008)  
    Triggers an application-specific custom effect.
```

```
#define kAttnFlagsLEDBit ((AttnFlagsType)0x0002)  
    Blinks an LED, if the device is so equipped.
```

```
#define kAttnFlagsSoundBit ((AttnFlagsType)0x0001)  
    Plays a sound.
```

```
#define kAttnFlagsVibrateBit  
    ((AttnFlagsType)0x0004)  
    Triggers vibration, if the device is so equipped.
```

```
#define kAttnFlagsUseUserSettings  
    ((AttnFlagsType)0x00000000)  
    System-wide preferences determine what means are used to  
    get the user's attention.
```

**See Also** [AttnFlagsType](#)

## Attention Manager

### Attention Override Flags

---

<b>Purpose</b>	Override the user's settings and force or prevent specific behaviors.
<b>Declared In</b>	AttentionMgr.h
<b>Constants</b>	<pre>#define kAttnFlagsAlwaysCustomEffect (kAttnFlagsCustomEffectBit)     Trigger an application-specific custom effect, regardless of     the user's settings.  #define kAttnFlagsAlwaysLED (kAttnFlagsLEDBit)     Blink an LED, if the device is so equipped, regardless of the     user's settings.  #define kAttnFlagsAlwaysSound (kAttnFlagsSoundBit)     Play a sound, regardless of the user's settings.  #define kAttnFlagsAlwaysVibrate (kAttnFlagsVibrateBit)     Vibrate, if the device is so equipped, regardless of the user's     settings.  #define kAttnFlagsEverything (kAttnFlagsAllBits)     Use every available means to get the user's attention,     regardless of the user's settings.  #define kAttnFlagsNoCustomEffect (kAttnFlagsCustomEffectBit&lt;&lt;16)     Prevent triggering of the application-specific custom effect,     regardless of the user's settings.  #define kAttnFlagsNoLED (kAttnFlagsLEDBit&lt;&lt;16)     Prevent the LED from flashing, regardless of the user's     settings.  #define kAttnFlagsNoSound (kAttnFlagsSoundBit&lt;&lt;16)     Prevent a sound from being played, regardless of the user's     settings.  #define kAttnFlagsNothing (kAttnFlagsAllBits&lt;&lt;16)     Disable all attention-getting mechanisms, regardless of the     user's settings.  #define kAttnFlagsNoVibrate (kAttnFlagsVibrateBit&lt;&lt;16)     Prevent vibration, regardless of the user's settings.</pre>

- Comments** These constants can be used in combination. For example, to disable both sound and the LED, use `kAttnFlagsNoSound | kAttnFlagsNoLED`.
- See Also** [AttnFlagsType](#)

## Commands

- Purpose** The set of possible commands that can be sent to the application requesting the user's attention, as one of the arguments that accompanies a [sysAppLaunchCmdAttention](#) launch code.
- Declared In** `AttentionMgr.h`
- Constants**
- ```
#define kAttnCommandCustomEffect  
((AttnCommandType)4)  
    Indicates that the application needs to perform any  
    application-specific special effects. This command is only  
    sent for attention items that set the  
    kAttnFlagsCustomEffectBit when they call  
    AttnGetAttention, which most applications won't do.
```
- ```
#define kAttnCommandDrawDetail  
((AttnCommandType)1)  
    Indicates that the application needs to draw the detailed  
    contents of the attention slip. The command arguments  
    parameter points to a structure of type  
    AttnCommandArgsDrawDetailTag.
```
- ```
#define kAttnCommandDrawList ((AttnCommandType)2)  
    Indicates that the application needs to draw the appropriate  
    list item in the attention slip. The command arguments  
    parameter points to a structure of type  
    AttnCommandArgsDrawListTag.
```
- ```
#define kAttnCommandGoThere ((AttnCommandType)5)  
    Tells the application to navigate to the item. The command  
    arguments parameter is NULL. An application commonly  
    calls SysAppLaunch\(\) upon receipt of this command to  
    have itself launched.
```
- ```
#define kAttnCommandGotIt ((AttnCommandType)6)  
    Tells the application that the user is dismissing the item. The  
    command arguments parameter points to a structure of type
```

## Attention Manager

### Attention Levels

---

[AttnCommandArgsGotItTag](#). The application may choose to clean up memory at this point.

```
#define kAttnCommandIterate ((AttnCommandType)8)
    This command is passed to the application during the enumeration of attention items. It is particularly useful after HotSync® operations, as it allows the application to examine each item, updating or removing those that are stale or invalid. The command arguments parameter points to a structure of type AttnCommandArgsIterateTag.
```

```
#define kAttnCommandPlaySound ((AttnCommandType)3)
    Indicates that the application needs to play a sound. The command arguments parameter is NULL.
```

```
#define kAttnCommandSnooze ((AttnCommandType)7)
    Indicates to the application that the user is snoozing. The command arguments parameter is NULL. Most applications do nothing upon receipt of this command. This command is passed to the appropriate application for each and every item currently pending, insistent or subtle. Applications with more than one attention item pending are called more than once.
```

**See Also** [AttnCommandType](#)

## Attention Levels

**Purpose** The lengths to which the Attention Manager should go in order to get the user's attention.

**Declared In** `AttentionMgr.h`

**Constants**

```
#define kAttnLevelInsistent ((AttnLevelType)0)
    An insistent attention attempt. Make a serious effort to get the user's attention by displaying a slip and optionally triggering one or more special effects.
```

```
#define kAttnLevelSubtle ((AttnLevelType)1)
    A subtle attention attempt. Notify the user using special effects, but don't disrupt the user with the slip if the device is in use.
```

**See Also** [AttnLevelType](#)

## Attention Manager Launch Codes

### sysAppLaunchCmdAttention

|                    |                                                                                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Sent when the Attention Manager needs your application to draw the details of your attention in the attention slip or perform some other application-specific function. |
| <b>Declared In</b> | <code>CmnLaunchCodes.h</code>                                                                                                                                           |
| <b>Prototype</b>   | <code>#define sysAppLaunchCmdAttention 60</code>                                                                                                                        |
| <b>Parameters</b>  | The launch code's parameter block pointer references a <a href="#">AttnLaunchCodeArgsType</a> structure.                                                                |

## Attention Manager Functions and Macros

### AttnDoSpecialEffects Function

|                    |                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Triggers an Attention Manager special effect set.                                                                                                                                                                                                                                                                                                               |
| <b>Declared In</b> | <code>AttentionMgr.h</code>                                                                                                                                                                                                                                                                                                                                     |
| <b>Prototype</b>   | <code>status_t AttnDoSpecialEffects<br/>(AttnFlagsType flags)</code>                                                                                                                                                                                                                                                                                            |
| <b>Parameters</b>  | → <i>flags</i><br>Specifies the behavior to be exhibited by this special effects request. See <a href="#">AttnFlagsType</a> for the various bits that make up this flag. Note that the behavior is undefined if you set incompatible flags. Supply <code>kAttnFlagsUseUserSettings</code> to have this attention request follow the user's pre-set preferences. |
| <b>Returns</b>     | Returns <code>errNone</code> if no problems were encountered. Returns <code>attnErrMemory</code> if there wasn't enough memory to accommodate the attention request.                                                                                                                                                                                            |
| <b>Comments</b>    | This routine is provided as a convenience for applications that need to trigger special effects. It does the equivalent of one "nag" of an Attention Manager special effect set.                                                                                                                                                                                |

## Attention Manager

### *AttnForgetIt*

---

### **AttnForgetIt Function**

- Purpose** Provides a way for applications to tell the Attention Manager to forget about an attention item.
- Declared In** `AttentionMgr.h`
- Prototype** `Boolean AttnForgetIt (DatabaseID dbID, uint32_t userData)`
- Parameters**
- *dbID*  
Database ID of the application making the request.
  - *userData*  
Identifier that distinguishes the attention attempt from others made by the same application. This identifier can be an integer, a pointer, or any other 32-bit value.
- Returns** Returns `true` if the item was removed, `false` if a matching item was not found.
- Comments** You typically call this function after your application has handled a “Go There” event and the user has read about the item. For example, if there is a subtle attention pending that says “you have three e-mail messages waiting” and you go to the e-mail application on your own and read your e-mail, the subtle notification must disappear. `AttnForgetIt()` allows the application to do this.
- Note that this call can be made when the Attention Manager slip is on-screen (though presumably that is rare, since the application is probably not doing much at this point). If this call removes a list item, then the Attention Manager may call back into other items to redraw the list.
- If this call removes the last item when any indicator is present, the indicator disappears.

### **AttnForgetItV40 Function**

- Purpose** Provides a way for applications to tell the Attention Manager to forget about an attention item.

---

**NOTE:** This function is provided for compatibility purposes only; applications should use `AttnForgetIt()` instead.

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Declared In</b>   | AttentionMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Prototype</b>     | Boolean AttnForgetItV40 (uint16_t <i>cardNo</i> ,<br>LocalID <i>dbID</i> , uint32_t <i>userData</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Parameters</b>    | → <i>cardNo</i><br>Card number on which the application making the request resides.<br>→ <i>dbID</i><br>Database ID of the application making the request.<br>→ <i>userData</i><br>Identifier that distinguishes the attention attempt from others made by the same application. This identifier can be an integer, a pointer, or any other 32-bit value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Returns</b>       | Returns <code>true</code> if the item was removed, <code>false</code> if a matching item was not found.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Comments</b>      | You typically call this function after your application has handled a “Go There” event and the user has read about the item. For example, if there is a subtle attention pending that says “you have three e-mail messages waiting” and you go to the e-mail application on your own and read your e-mail, the subtle notification must disappear. <code>AttnForgetItV40 ( )</code> allows the application to do this. Note that this call can be made when the Attention Manager slip is on-screen (though presumably that is rare, since the application is probably not doing much at this point). If this call removes a list item, then the Attention Manager may call back into other items to redraw the list.<br><br>If this call removes the last item when any indicator is present, the indicator disappears. |
| <b>Compatibility</b> | This function is provided for compatibility purposes only; applications should use <code>AttnForgetIt ( )</code> —which omits the obsolete <code>cardNo</code> parameter—instead.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Attention Manager

### *AttnGetAttention*

---

## AttnGetAttention Function

- Purpose** Requests the user's attention.
- Declared In** `AttentionMgr.h`
- Prototype** `status_t AttnGetAttention (DatabaseID dbID, uint32_t userData, AttnLevelType level, AttnFlagsType flags, uint16_t nagRateInSeconds, uint16_t nagRepeatLimit)`
- Parameters**
- *dbID*  
Database ID of the application making the request.
  - *userData*  
Application-specific data that is included in what is passed along with a [sysAppLaunchCmdAttention](#) launch code. *userData* can be an integer, a pointer, or any other 32-bit value as needed by your application. Most applications pass the unique ID or other key for the record which caused the attention request. *userData* is also used to distinguish a given attention attempt from others made by the same application.
  - *level*  
Indicates the annoyance level. Pass one of the values defined for [AttnLevelType](#).
  - *flags*  
Behavior override, if necessary, for this attention request. This override allows, for instance, silent alarms or noisy alarms. See [AttnFlagsType](#) for the various bits that make up this flag. Note that the behavior is undefined if you set incompatible flags. Supply `kAttnFlagsUseUserSettings` to have this attention request follow the user's pre-set preferences.
  - *nagRateInSeconds*  
How long to wait before nagging.
  - *nagRepeatLimit*  
How many times to nag, excluding the first attempt.
- Returns** Returns `errNone` if no problems were encountered. Returns `attnErrMemory` if there wasn't enough memory to accommodate the attention request.

**Comments** The combination of `dbID` and `userData` uniquely identify an attention-getting attempt. If another call is made to `AttnGetAttention` with identical values for these arguments, an error is reported. To update or delete an existing attention item, pass these same values to [AttnUpdate\(\)](#) or [AttnForgetIt\(\)](#), respectively.

In response to `AttnGetAttention`, the behavior of the operating system or application depends on whether there already are other demands and on the annoyance level passed in the `AttnGetAttention` call.

- No other demands, insistent attention request:

The Attention Manager puts up a slip that details the current attempt to get the user's attention.

- Other demands exist, insistent attention request:

The Attention Manager adds a summary of the current attempt to get the user's attention to a list of things that need attention. If the slip is currently in detail form—which is the case if just one other demand exists—the view is refreshed, changing from detail to list form.

- Subtle attention request:

The Attention Manager adds the item to its list for later display and presents the attention indicator, unless the slip is currently being displayed. In this event, the new subtle attention item simply appears in the list (switching to list mode if necessary).

**See Also** [AttnUpdate\(\)](#)

## **AttnGetAttentionV40 Function**

**Purpose** Requests the user's attention.

---

**NOTE:** This function is provided for compatibility purposes only; applications should use [AttnGetAttention\(\)](#) instead.

---

## Attention Manager

### *AttnGetAttentionV40*

---

**Declared In**    `AttentionMgr.h`

**Prototype**    `status_t AttnGetAttentionV40 (uint16_t cardNo,  
                  LocalID dbID, uint32_t userData,  
                  AttnLevelType level, AttnFlagsType flags,  
                  uint16_t nagRateInSeconds,  
                  uint16_t nagRepeatLimit)`

**Parameters**

- *cardNo*  
Card number on which the application making the request resides.
- *dbID*  
Database ID of the application making the request.
- *userData*  
Application-specific data that is included in what is passed along with a [sysAppLaunchCmdAttention](#) launch code. *userData* can be an integer, a pointer, or any other 32-bit value as needed by your application. Most applications pass the unique ID or other key for the record which caused the attention request. *userData* is also used to distinguish a given attention attempt from others made by the same application.
- *level*  
Indicates the annoyance level. Pass one of the values defined for [AttnLevelType](#).
- *flags*  
Behavior override, if necessary, for this attention request. This override allows, for instance, silent alarms or noisy alarms. See [AttnFlagsType](#) for the various bits that make up this flag. Note that the behavior is undefined if you set incompatible flags. Supply `kAttnFlagsUseUserSettings` to have this attention request follow the user's pre-set preferences.
- *nagRateInSeconds*  
How long to wait before nagging.
- *nagRepeatLimit*  
How many times to nag, excluding the first attempt.

**Returns**    Returns `errNone` if no problems were encountered. Returns `attnErrMemory` if there wasn't enough memory to accommodate the attention request.

**Comments** The combination of `cardNo`, `dbID` and `userData` uniquely identify an attention-getting attempt. If another call is made to `AttnGetAttention` with identical values for these arguments, an error is reported. To update or delete an existing attention item, pass these same values to [AttnUpdate\(\)](#) or [AttnForgetIt\(\)](#), respectively.

In response to `AttnGetAttention`, the behavior of the operating system or application depends on whether there already are other demands and on the annoyance level passed in the `AttnGetAttention` call.

- No other demands, insistent attention request:

The Attention Manager puts up a slip that details the current attempt to get the user's attention.

- Other demands exist, insistent attention request:

The Attention Manager adds a summary of the current attempt to get the user's attention to a list of things that need attention. If the slip is currently in detail form—which is the case if just one other demand exists—the view is refreshed, changing from detail to list form.

- Subtle attention request:

The Attention Manager presents the attention indicator, and adds the item to its list for later display, unless the slip is currently being displayed in list mode. In this event, the new subtle attention item simply appears in the list.

**Compatibility** This function is provided for compatibility purposes only; applications should use [AttnGetAttention\(\)](#)—which omits the obsolete `cardNo` parameter—instead.

**See Also** [AttnUpdate\(\)](#)

## Attention Manager

### *AttnGetCounts*

---

### **AttnGetCounts Function**

- Purpose** Returns the number of attention items that are currently pending.
- Declared In** `AttentionMgr.h`
- Prototype**  
`uint16_t AttnGetCounts (DatabaseID dbID,  
uint16_t *insistentCountP,  
uint16_t *subtleCountP)`
- Parameters**
- *dbID*  
If this value is zero, counts pending attention items from all applications. Otherwise, counts only pending attention items from applications with the specified database ID.
  - ← *insistentCountP*  
Pointer to a 16-bit unsigned value that is filled in with the number of insistent items pending. Pass NULL for this parameter if you don't need to know the number of insistent items that are pending.
  - ← *subtleCountP*  
Pointer to a 16-bit unsigned value that is filled in with the number of subtle items pending. Pass NULL for this parameter if you don't need to know the number of subtle items that are pending.
- Returns** Returns the total number of items, both insistent and subtle, that are currently pending.
- Comments** Call this function if you need to exhibit different behavior if attention items are already pending.

### **AttnGetCountsV40 Function**

- Purpose** Returns the number of attention items that are currently pending.

---

**NOTE:** This function is provided for compatibility purposes only; applications should use [AttnGetCounts\(\)](#) instead.

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Declared In</b>   | AttentionMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Prototype</b>     | <pre>uint16_t AttnGetCountsV40 (uint16_t cardNo,<br/>    LocalID dbID, uint16_t *insistentCountP,<br/>    uint16_t *subtleCountP)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Parameters</b>    | <p>→ <i>cardNo</i><br/>If this value is zero, counts pending attention items from applications on all cards. Otherwise, counts only pending attention items from applications on the specified card.</p> <p>→ <i>dbID</i><br/>If this value is zero, counts pending attention items from all applications. Otherwise, counts only pending attention items from applications with the specified database ID.</p> <p>← <i>insistentCountP</i><br/>Pointer to a 16-bit unsigned value that is filled in with the number of insistent items pending. Pass NULL for this parameter if you don't need to know the number of insistent items that are pending.</p> <p>← <i>subtleCountP</i><br/>Pointer to a 16-bit unsigned value that is filled in with the number of subtle items pending. Pass NULL for this parameter if you don't need to know the number of subtle items that are pending.</p> |
| <b>Returns</b>       | Returns the total number of items, both insistent and subtle, that are currently pending.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Comments</b>      | Call this function if you need to exhibit different behavior if attention items are already pending.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Compatibility</b> | This function is provided for compatibility purposes only; applications should use <a href="#">AttnGetCounts()</a> —which omits the obsolete <i>cardNo</i> parameter—instead.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

## Attention Manager

*AttnIndicatorEnable*

---

### AttnIndicatorEnable Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Enables and disables the on-screen attention indicator displayed when the status bar is not visible.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Declared In</b> | <code>AttentionMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Prototype</b>   | <code>void AttnIndicatorEnable (Boolean <i>enableIt</i>)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Parameters</b>  | → <i>enableIt</i><br>true to enable the attention indicator, false to disable it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Returns</b>     | Returns nothing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Comments</b>    | <p>This function is used by applications to enable or disable the on-screen attention indicator that appears in the upper left corner of the screen when the status bar is not visible. The indicator appears here only when all of the following are true:</p> <ul style="list-style-type: none"><li>• The status bar is not visible.</li><li>• The indicator is enabled.</li><li>• The Attention Manager needs to get the user's attention.</li><li>• The operating system isn't using the display in such a way as to prevent the attention indicator from showing, such as when the menu bar is being displayed or when a modal dialog is on top of the form.</li></ul> <p>The attention indicator is enabled by default. If your application controls the upper portion of the screen and needs to prevent the attention indicator from being displayed, call <a href="#">AttnIndicatorEnable()</a> and pass it a value of false.</p> <p>This function has no effect on the attention indicator that is displayed in the status bar.</p> |
| <b>See Also</b>    | <a href="#">AttnIndicatorEnabled()</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## AttnIndicatorEnabled Function

|                    |                                                                                                                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Returns whether the on-screen attention indicator displayed when the status bar is not visible is currently enabled.                                                                                          |
| <b>Declared In</b> | <code>AttentionMgr.h</code>                                                                                                                                                                                   |
| <b>Prototype</b>   | <code>Boolean AttnIndicatorEnabled (void)</code>                                                                                                                                                              |
| <b>Parameters</b>  | None.                                                                                                                                                                                                         |
| <b>Returns</b>     | Returns <code>true</code> if the on-screen attention indicator (displayed in the upper left hand corner of the screen when the status bar is not visible) is currently enabled, <code>false</code> otherwise. |
| <b>See Also</b>    | <a href="#">AttnIndicatorEnable()</a>                                                                                                                                                                         |

## AttnIterate Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Instructs the Attention Manager to check each attention item currently pending and, for those that match the specified database ID, send the <a href="#">sysAppLaunchCmdAttention</a> launch code to the application that made the attention request.                                                                                                                                                                                                                                                                                                                                                    |
| <b>Declared In</b> | <code>AttentionMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Prototype</b>   | <code>void AttnIterate (DatabaseID dbID,<br/>uint32_t iterationData)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Parameters</b>  | <code>→ dbID</code><br>Database ID of the application that made the request.<br><code>→ iterationData</code><br>Any necessary data that the application may need in order to process the launch code.                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Returns</b>     | Returns nothing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Comments</b>    | This function iterates through all of the attention requests made by this application and uses the launch code for each to inform the application about the attention request. When an application receives a <a href="#">sysAppLaunchCmdSyncNotify</a> launch code, signifying that a HotSync occurred that affected that application's databases, the application typically calls <code>AttnIterate</code> so it can remove attention requests for records that may have been removed during the HotSync. Applications can also call <a href="#">AttnGetAttention()</a> after a HotSync, if necessary. |

## Attention Manager

### *AttnIterateV40*

---

Note that you can call [AttnForgetIt\(\)](#) inside the iteration since it only marks the record for deletion and thus doesn't confuse the iteration.

## AttnIterateV40 Function

**Purpose** Instructs the Attention Manager to check each attention item currently pending and, for those that match the specified card number and database ID, send the [sysAppLaunchCmdAttention](#) launch code to the application that made the attention request.

---

**NOTE:** This function is provided for compatibility purposes only; applications should use [AttnIterate\(\)](#) instead.

---

**Declared In** `AttentionMgr.h`

**Prototype** `void AttnIterateV40 (uint16_t cardNo,  
LocalID dbID, uint32_t iterationData)`

**Parameters**

- *cardNo*  
Card number on which the application that made the request resides.
- *dbID*  
Database ID of the application that made the request.
- *iterationData*  
Any necessary data that the application may need in order to process the launch code.

**Returns** Returns nothing.

**Comments** This function iterates through all of the attention requests made by this application and uses the launch code for each to inform the application about the attention request. When an application receives a [sysAppLaunchCmdSyncNotify](#) launch code, signifying that a HotSync occurred that affected that application's databases, the application typically calls `AttnIterate` so it can remove attention requests for records that may have been removed during the HotSync. Applications can also call [AttnGetAttention\(\)](#) after a HotSync, if necessary.

Note that you can call [AttnForgetIt\(\)](#) inside the iteration since it only marks the record for deletion and thus doesn't confuse the iteration.

**Compatibility** This function is provided for compatibility purposes only; applications should use [AttnIterate\(\)](#)—which omits the obsolete `cardNo` parameter—instead.

## AttnListOpen Function

**Purpose** Displays the attention slip in list mode and, after the user has dismissed it, acts accordingly based on how it was dismissed.

**Declared In** `AttentionMgr.h`

**Prototype** `void AttnListOpen (void)`

**Parameters** None.

**Returns** Returns nothing.

**Comments** This function allows applications that do not provide the on screen attention indicator (the one that is displayed when the status bar is not visible) to open the list, if necessary.

## AttnUpdate Function

**Purpose** Updates one or more aspects of a specified attention item.

**Declared In** `AttentionMgr.h`

**Prototype** `Boolean AttnUpdate (DatabaseID dbID,  
uint32_t userData, AttnFlagsType *flagsP,  
uint16_t *nagRateInSecondsP,  
uint16_t *nagRepeatLimitP)`

**Parameters**

- `dbID`  
Database ID of the application that made the request.
- `userData`  
Application-specific data that is included in what is passed along with a [sysAppLaunchCmdAttention](#) launch code. `userData` can be an integer, a pointer, or any other 32-bit value. Most applications pass the unique ID or other key for the record which caused the attention request. The value of

## Attention Manager

### *AttnUpdate*

---

the `userData` parameter is also used to distinguish a given attention attempt from others made by the same application.

→ *flagsP*

Pointer to a set of flags that can be used to override user-specified attention behavior; for instance, to force silent or noisy alarms. See [AttnFlagsType](#) for the various bits that make up this flag, and note that the behavior is undefined if you set incompatible flags. Pass `NULL` to leave the current flag settings unchanged.

→ *nagRateInSecondsP*

Pointer to the length of time to wait before nagging. Pass `NULL` to leave the “nag rate” unchanged.

→ *nagRepeatLimitP*

Pointer to the maximum number of times the user should be nagged. Pass `NULL` to leave the nag repeat limit unchanged.

**Returns** Returns `true` if the update was successful, `false` if no matching attention item was found.

**Comments** This call may result in the sending of the [sysAppLaunchCmdAttention](#) launch code to your application. It may also result in this launch code being sent to other pending attention requests.

You call `AttnUpdate` to tell the Attention Manager to update, forcing it to call into all of its clients to redraw. This provides a way for an application to update the text of an attention item without tearing down and re-opening the Attention Manager slip. For example, `AttnUpdate` could be used to update an existing email notification to say “You have three new email messages” when additional messages are received.

Although `AttnUpdate` may cause a given attention item to redraw, it does not rerun the special effects (if any) that occurred when that attention item was added. If you want to trigger Attention Manager effects for a particular item, call [AttnForgetIt\(\)](#) followed by [AttnGetAttention\(\)](#).

**See Also** [AttnGetAttention\(\)](#)

## AttnUpdateV40 Function

**Purpose** Updates one or more aspects of a specified attention item.

---

**NOTE:** This function is provided for compatibility purposes only; applications should use [AttnUpdate\(\)](#) instead.

---

**Declared In** AttentionMgr.h

**Prototype** Boolean AttnUpdateV40 (uint16\_t cardNo,  
LocalID dbID, uint32\_t userData,  
AttnFlagsType \*flagsP,  
uint16\_t \*nagRateInSecondsP,  
uint16\_t \*nagRepeatLimitP)

**Parameters**

- *cardNo*  
Card number on which the application that made the request resides.
- *dbID*  
Database ID of the application that made the request.
- *userData*  
Application-specific data that is included in what is passed along with a [sysAppLaunchCmdAttention](#) launch code. *userData* can be an integer, a pointer, or any other 32-bit value. Most applications pass the unique ID or other key for the record which caused the attention request. The value of the *userData* parameter is also used to distinguish a given attention attempt from others made by the same application.
- *flagsP*  
Pointer to a set of flags that can be used to override user-specified attention behavior; for instance, to force silent or noisy alarms. See [AttnFlagsType](#) for the various bits that make up this flag, and note that the behavior is undefined if you set incompatible flags. Pass NULL to leave the current flag settings unchanged.
- *nagRateInSecondsP*  
Pointer to the length of time to wait before nagging. Pass NULL to leave the “nag rate” unchanged.
- *nagRepeatLimitP*  
Pointer to the maximum number of times the user should be nagged. Pass NULL to leave the nag repeat limit unchanged.

## Attention Manager

*AttnUpdateV40*

---

- Returns** Returns `true` if the update was successful, `false` if no matching attention item was found.
- Comments** This call may result in the sending of the [sysAppLaunchCmdAttention](#) launch code to your application. It may also result in this launch code being sent to other pending attention requests.
- You call `AttnUpdate` to tell the Attention Manager to update, forcing it to call into all of its clients to redraw. This provides a way for an application to update the text of an attention item without tearing down and re-opening the Attention Manager slip. For example, `AttnUpdate` could be used to update an existing email notification to say “You have three new email messages” when additional messages are received.
- Although `AttnUpdate` may cause a given attention item to redraw, it does not rerun the special effects (if any) that occurred when that attention item was added. If you want to trigger Attention Manager effects for a particular item, call [AttnForgetIt\(\)](#) followed by [AttnGetAttention\(\)](#).
- Compatibility** This function is provided for compatibility purposes only; applications should use [AttnUpdate\(\)](#)—which omits the obsolete `cardNo` parameter—instead.
- See Also** [AttnGetAttention\(\)](#)

# Category Manager Sync

---

The Category Manager synchronization functions allow you to obtain category change tracking information.

---

**NOTE:** The APIs described in this chapter aren't generally used by Palm OS applications.

---

Because `CatMgrSync.h` only declares functions, this chapter is composed of a single section only:

[Category Manager Sync Functions and Macros](#) . . . . 161

The header file `CatMgrSync.h` declares the API that this chapter describes.

## Category Manager Sync Functions and Macros

### CatMgrSyncGetModifiedCategories Function

|                    |                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Retrieves the ID of each modified category in a Schema database.                                                                                            |
| <b>Declared In</b> | <code>CatMgrSync.h</code>                                                                                                                                   |
| <b>Prototype</b>   | <pre>status_t CatMgrSyncGetModifiedCategories (DmOpenRef dbP, const DbSyncCounterType *counterP, CategoryID **categoriesPP, uint32_t *numCategoriesP)</pre> |
| <b>Parameters</b>  | <p>→ <i>dbP</i><br/> DmOpenRef to an open database.</p>                                                                                                     |

## Category Manager Sync

### *CatMgrSyncGetModifiedCategories*

---

→ *counterP*

Pointer to the desktop counter.

← *categoriesPP*

Pointer to an array of category IDs, where each identifies a category that has changed.

← *numCategoriesP*

Pointer to a variable that receives the number of elements in *categoriesPP*.

**Returns** Returns `errNone` if the category IDs were successfully retrieved, or one of the following otherwise:

`catmErrInvalidParam`

*dbP* is NULL.

`dmErrNotSchemaDatabase`

The specified database is not a Schema database.

`catmErrMemError`

A memory error occurred.

**Comments** This function returns a list of categories that have a higher sync counter than the *counterP* parameter. That is, it returns all categories that were modified since the database had the same sync counter value as *counterP*. The Category Manager allocates the category ID list returned in *categoriesPP*; your code is responsible for freeing the list by calling [CatMgrSyncReleaseStorage\(\)](#).

**Example** The following code excerpt shows how you could use this function to retrieve a list of modified categories by ID:

---

```
DbSyncCounterType cover;
CategoryID categoriesP;
uint32_t numCategories;

dbRef = DbOpenDatabase(dbID, dmModeReadWrite, dbShareNone,
    dbDefaultSortID);

cover = 0;
CatMgrSyncGetModifiedCategories(dbRef, &cover, &categoriesP,
    &numCategories);
//
// do something with the category IDs in categoriesP here
//
CatMgrSyncReleaseStorage(categoriesP);
```

---

## CatMgrSyncGetPurgeCounter Function

- Purpose** Retrieve the category purge counter for a specified database.
- Declared In** `CatMgrSync.h`
- Prototype** `status_t CatMgrSyncGetPurgeCounter (DmOpenRef dbP,  
DbSyncCounterType *purgeCounterP)`
- Parameters**
- *dbP*  
DmOpenRef to an open database.
  - ← *purgeCounterP*  
Pointer to a DbSyncCounterType variable that receives the row purge count.
- Returns** Returns `errNone` if the category purge counter was successfully retrieved, or one of the following otherwise:
- `catmErrInvalidParam`  
*dbP* is NULL.
  - `dmErrNotSchemaDatabase`  
The specified database is not a Schema database.
  - `PrvLockCatMgrInfo`  
The specified database has no defined categories.
  - `catmErrMemError`  
A memory error occurred.

## CatMgrSyncReleaseStorage Function

- Purpose** Release a dynamic heap chunk that was allocated by the Category Manager when [CatMgrSyncGetModifiedCategories\(\)](#) is called.
- Declared In** `CatMgrSync.h`
- Prototype** `status_t CatMgrSyncReleaseStorage (DmOpenRef dbP,  
MemPtr bufferP)`
- Parameters**
- *dbP*  
DmOpenRef to an open database.

## Category Manager Sync

*CatMgrSyncReleaseStorage*

---

→ *bufferP*

Pointer to the chunk to be released. This should only be a chunk that was allocated for you by [CatMgrSyncGetModifiedCategories\(\)](#).

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`catmErrInvalidParam`

*dbP* is NULL or the specified buffer has zero length.

`dmErrNotSchemaDatabase`

The specified database is not a Schema database.

`catmErrInvalidStoragePtr`

The *bufferP* parameter is invalid.

`memErrInvalidParam`

The *bufferP* parameter is invalid.

# Common Battery Types

---

The header file `CmnBatteryTypes.h` simply defines battery types and battery states. The material in this chapter is divided up as follows:

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| <a href="#">Common Battery Types Structures and Types</a> . . . . . | 165 |
| <a href="#">Common Battery Types Constants</a> . . . . .            | 166 |

## Common Battery Types Structures and Types

### **SysBatteryKind Typedef**

|                    |                                                                                        |
|--------------------|----------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains a value indicating the type of battery used by the handheld.                  |
| <b>Declared In</b> | <code>CmnBatteryTypes.h</code>                                                         |
| <b>Prototype</b>   | <code>typedef Enum8 SysBatteryKind</code>                                              |
| <b>Comments</b>    | See <a href="#">SysBatteryKindTag</a> for the set of values that this type can assume. |

### **SysBatteryState Typedef**

|                    |                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains a value indicating the current state of the handheld's battery.                |
| <b>Declared In</b> | <code>CmnBatteryTypes.h</code>                                                          |
| <b>Prototype</b>   | <code>typedef Enum8 SysBatteryState</code>                                              |
| <b>Comments</b>    | See <a href="#">SysBatteryStateTag</a> for the set of values that this type can assume. |

## Common Battery Types Constants

### **SysBatteryKindTag Enum**

|                    |                                                                                       |
|--------------------|---------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Identify the type of battery used in a Palm Powered handheld.                         |
| <b>Declared In</b> | <code>CmnBatteryTypes.h</code>                                                        |
| <b>Constants</b>   | <code>sysBatteryKindAlkaline = 0</code><br>Alkaline.                                  |
|                    | <code>sysBatteryKindNiCad</code><br>Nickel-Cadmium (NiCad).                           |
|                    | <code>sysBatteryKindLiIon</code><br>Lithium Ion.                                      |
|                    | <code>sysBatteryKindRechAlk</code><br>Rechargeable alkaline.                          |
|                    | <code>sysBatteryKindNiMH</code><br>Nickel-Metal-Hydride.                              |
|                    | <code>sysBatteryKindLiIon1400</code><br>Lithium-Ion, 1400 mA.                         |
|                    | <code>sysBatteryKindFuelCell</code><br>Fuel cell.                                     |
|                    | <code>sysBatteryKindPlutonium237</code><br>Future power source.                       |
|                    | <code>sysBatteryKindAntiMatter</code><br>Future power source.                         |
|                    | <code>sysBatteryKindLast = 0xFF</code><br>The upper limit of the battery type values. |

## **SysBatteryStateTag Enum**

|                    |                                                                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Identify the state of the handheld's battery.                                                                                                                                                               |
| <b>Declared In</b> | <code>CmnBatteryTypes.h</code>                                                                                                                                                                              |
| <b>Constants</b>   | <code>sysBatteryStateNormal = 0</code><br>The battery is in a normal state, able to provide sufficient power for normal handheld operation.                                                                 |
|                    | <code>sysBatteryStateLowBattery</code><br>Battery power is low; the battery should be recharged. The battery can still provide sufficient power for normal handheld operation, but only for a limited time. |
|                    | <code>sysBatteryStateCritBattery</code><br>The battery state is critical; it should be recharged immediately. Normal operations will be curtailed.                                                          |
|                    | <code>sysBatteryStateShutdown</code><br>The battery cannot provide sufficient power to run the display. The handheld will be, or is already, shut down.                                                     |

## Common Battery Types

*SysBatteryStateTag*

---

# Common Error Codes

---

The header file `CmnErrors.h` declares the error code classes—base values from which all Palm OS error codes are defined—as well as many error codes that are commonly used throughout Palm OS.

Because `CmnErrors.h` declares only constants, this chapter consists of a single section:

[Common Error Codes Constants](#). . . . . 169

## Common Error Codes Constants

### Error Code Classes

|                    |                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | These constants define base values from which error code values are derived.                                                                                                                                                                                                                                                                     |
| <b>Declared In</b> | <code>CmnErrors.h</code>                                                                                                                                                                                                                                                                                                                         |
| <b>Constants</b>   | <pre>#define actvErrorClass 0x80002000     Activation application errors  #define almErrorClass 0x80000900     Alarm Manager errors  #define amErrorClass 0x80003D00  #define appErrorClass 0x80008000     Application-defined errors  #define appMgrErrorClass 0x80004000  #define attnErrorClass 0x80002E00     Attention Manager errors</pre> |

## Common Error Codes

### *Error Code Classes*

---

```
#define azmErrorClass 0x80003C00

#define bltErrorClass 0x80002300
    Blitter Driver errors

#define blthErrorClass 0x80003100
    Bluetooth Library errors

#define bndErrorClass 0x80004A00
    Binder Errors.

#define catmErrorClass 0x80004700

#define certErrorClass 0x80004800

#define cmpErrorClass 0x80000D00
    Connection Manager (HotSync) errors

#define cncErrorClass 0x80001F00
    Connection Manager (serial communication) errors

#define cpmErrorClass 0x80003800
    Cryptographic Provider Manager errors

#define dalErrorClass 0x8000FF00
    DAL Errors.

#define dirErrorClass 0x80003E00

#define dispErrorClass 0x80002200
    Display Driver errors

#define dlkErrorClass 0x80000E00
    Desktop Link Manager errors

#define dmErrorClass 0x80000200
    Data Manager errors

#define drvrErrorClass 0x80004500

#define em68kErrorClass 0x80003400

#define emuErrorClass 0x80001C00
    Emulator Control Manager errors
```

```
#define errInfoClass 0x80007F00

#define evtErrorClass 0x80000700
    System Event Manager errors
#define exgErrorClass 0x80001500
    Exchange Manager errors
#define expErrorClass 0x80002900
    Expansion Manager errors
#define fileErrorClass 0x80001600
    File Stream Manager errors
#define flpErrorClass 0x80000680
    Floating Point Library errors
#define flshErrorClass 0x80001D00
    Flash Manager errors
#define fplErrorClass 0x80000600
    Old Floating Point Library errors
#define ftrErrorClass 0x80000C00
    Feature Manager errors
#define grfErrorClass 0x80001000
    Graffiti Manager errors
#define grmlErrorClass 0x80003500

#define halErrorClass 0x80003B00

#define hsExgErrorClass 0x80004E00

#define htalErrorClass 0x80001300
    HTAL Library errors
#define HttpErrorClass 0x80004C00

#define hwrErrorClass 0x80003000
    Handwriting Recognition Manager errors
#define inetErrorClass 0x80001400
    INet Library errors
```

## Common Error Codes

### *Error Code Classes*

---

```
#define intlErrorClass 0x80002C00
    International Manager errors
#define iosErrorClass 0x80004200

#define IrCommErrorClass 0x80003700

#define IrErrorClass 0x80003600

#define kalErrorClass 0x80003A00

#define lmErrorClass 0x80002B00
    Locale Manager errors
#define lz77ErrorClass 0x80002700
    LZ77 Library errors
#define mdmErrorClass 0x80001100
    Modem Manager errors
#define mediaErrorClass 0x80004600
    Media Manager Errors.
#define memErrorClass 0x80000100
    Memory Manager errors
#define menuErrorClass 0x80002600
    Menu Manager errors
#define netErrorClass 0x80001200
    Net Library errors
#define oemErrorClass 0x80007000
    OEM/Licensee errors
#define omErrorClass 0x80002500
    Overlay Manager errors
#define padErrorClass 0x80000F00
    PAD Manager errors
#define pdiErrorClass 0x80002D00
    PDI Library errors
#define penErrorClass 0x80000B00
    Pen Manager errors
```

```
#define perfErrorClass 0x80004400

#define pinsErrorClass 0x80005000

#define posixErrorClass 0x80005300

#define pppErrorClass 0x80004F00

#define pwrErrorClass 0x80001E00
    Power Manager errors

#define radioErrorClass 0x80002100
    Radio Manager (library) errors

#define ralErrorClass 0x80004100

#define regexpErrorClass 0x80005200

#define rfutErrorClass 0x80001700
    RFUT Library errors

#define secErrorClass 0x80001B00
    Security Library errors

#define secSvcErrorClass 0x80004900

#define serErrorClass 0x80000300
    Serial Manager errors

#define signErrorClass 0x80004300

#define slkErrorClass 0x80000400
    Serial Link Manager errors

#define smsErrorClass 0x80002800
    SMS Library errors

#define sndErrorClass 0x80000800
    Sound Manager errors

#define sslErrorClass 0x80003900
    SSL errors
```

## Common Error Codes

### *Error Code Classes*

---

```
#define statErrorClass 0x80005100

#define svcErrorClass 0x80003F00

#define syncMgrErrorClass 0x80004B00

#define sysErrorClass 0x80000500
    System Manager errors
#define telErrorClass 0x80002F00
    Telephony Manager errors
#define timErrorClass 0x80000A00
    Time Manager errors
#define tlsErrorClass 0x80003300

#define tsmErrorClass 0x80001900
    Text Services Manager errors
#define txtErrorClass 0x80001800
    Text Manager errors
#define udaErrorClass 0x80003200
    UDA Manager errors
#define vfsErrorClass 0x80002A00
    Virtual Filesystem Manager and Filesystem Library errors
#define webErrorClass 0x80001A00
    Web Library errors
#define winErrorClass 0x80002400
    Window Manager errors
#define xSyncErrorClass 0x80004D00
```

**Purpose**

**Declared In** CmnErrors.h

**Constants**

```
#define errResponseBreak68K 0x00000004

#define errResponseBreakBoth
    (errResponseBreakNative + errResponseBreak68K)

#define errResponseBreakNative 0x00000002

#define errResponseDefaultSet
    (errResponseBreakNative |
     errResponseKillProcess | errResponseSoftReset |
     errResponseShutdown )

#define errResponseIgnore 0x00000008

#define errResponseKillProcess 0x00000010

#define errResponseShutdown 0x00000040

#define errResponseSoftReset 0x00000020

#define errResponseTryHighLevel 0x00000001
```

**Purpose**

**Declared In** CmnErrors.h

**Constants**

```
#define errNone 0x00000000
    No error.
```

## Common Error Codes

### Binder Errors

---

```
#define errorClassMask 0xFFFFFFFF00
    AND this mask with an error code value to determine the
    error's class. See "Error Code Classes" on page 169 for a list
    of common error classes.

#define errReportStripContext 0x00000001
```

## Binder Errors

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Declared In</b> | CmnErrors.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Constants</b>   | <pre>#define bndErrBadInterface     ((status_t)(bndErrorClass   7))  #define bndErrBadTransact     ((status_t)(bndErrorClass   5))  #define bndErrBadType ((status_t)(bndErrorClass       2))  #define bndErrDead ((status_t)(bndErrorClass   3))  #define bndErrMissingArg ((status_t)(bndErrorClass       1))  #define bndErrOutOfStack ((status_t)(bndErrorClass       10))  #define bndErrTooManyLoopers     ((status_t)(bndErrorClass   6))  #define bndErrUnknownMethod     ((status_t)(bndErrorClass   8))</pre> |

```
#define bndErrUnknownProperty  
((status_t)(bndErrorClass | 9))
```

```
#define bndErrUnknownTransact  
((status_t)(bndErrorClass | 4))
```

## **DAL Errors**

**Purpose**

**Declared In** CmnErrors.h

**Constants** #define kDALError ((status\_t)(dalErrorClass |  
0x00FF))

```
#define kDALTimeout ((status_t)(sysErrorClass |  
1))
```

## **Media Manager Errors**

**Purpose** Errors generated by the Media Manager.

**Declared In** CmnErrors.h

**Constants** #define mediaErrAlreadyConnected  
((status\_t)(mediaErrorClass | 4))

```
#define mediaErrAlreadyVisited  
((status_t)(mediaErrorClass | 2))
```

```
#define mediaErrFormatMismatch  
((status_t)(mediaErrorClass | 1))
```

```
#define mediaErrNoBufferSource  
((status_t)(mediaErrorClass | 6))
```

## Common Error Codes

### Media Manager Errors

---

```
#define mediaErrNotConnected  
    ((status_t)(mediaErrorClass | 5))
```

```
#define mediaErrStreamExhausted  
    ((status_t)(mediaErrorClass | 3))
```

#### Purpose

**Declared In** CmnErrors.h

#### Constants

```
#define regexpErrCorruptedOpcode  
    ((status_t)(regexpErrorClass | 16))
```

```
#define regexpErrCorruptedPointers  
    ((status_t)(regexpErrorClass | 15))
```

```
#define regexpErrCorruptedProgram  
    ((status_t)(regexpErrorClass | 13))
```

```
#define regexpErrInternalError  
    ((status_t)(regexpErrorClass | 10))
```

```
#define regexpErrInvalidBracketRange  
    ((status_t)(regexpErrorClass | 8))
```

```
#define regexpErrJunkOnEnd  
    ((status_t)(regexpErrorClass | 4))
```

```
#define regexpErrMemoryCorruption  
    ((status_t)(regexpErrorClass | 14))
```

```
#define regexpErrNestedStarQuestionPlus  
    ((status_t)(regexpErrorClass | 6))
```

```
#define regexpErrQuestionPlusStarFollowsNothing  
    ((status_t)(regexpErrorClass | 11))
```

```
#define regExpErrStarPlusOneOperandEmpty
    ((status_t)(regExpErrorClass | 5))

#define regExpErrTooBig
    ((status_t)(regExpErrorClass | 2))

#define regExpErrTooManyParenthesis
    ((status_t)(regExpErrorClass | 3))

#define regExpErrTrailingBackslash
    ((status_t)(regExpErrorClass | 12))

#define regExpErrUnmatchedBracket
    ((status_t)(regExpErrorClass | 9))

#define regExpErrUnmatchedParenthesis
    ((status_t)(regExpErrorClass | 1))
```

## **System Errors**

|                    |                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | General system errors.                                                                                                                                                                                                                                                                                                                               |
| <b>Declared In</b> | CmnErrors.h                                                                                                                                                                                                                                                                                                                                          |
| <b>Constants</b>   | <pre>#define sysErrBufTooSmall     ((status_t)(sysErrorClass   13))  #define sysErrCPUArchitecture     ((status_t)(sysErrorClass   40))     The program requires a different CPU architecture to run.  #define sysErrDynamicLinkerError     ((status_t)(sysErrorClass   38))  #define sysErrInternalError     ((status_t)(sysErrorClass   37))</pre> |

## Common Error Codes

### *System Errors*

---

```
#define sysErrInvalidSignature
    ((status_t)(sysErrorClass | 36))

#define sysErrLibNotFound
    ((status_t)(sysErrorClass | 10))

#define sysErrModuleFound68KCode
    ((status_t)(sysErrorClass | 33))

#define sysErrModuleIncompatible
    ((status_t)(sysErrorClass | 32))

#define sysErrModuleInvalid
    ((status_t)(sysErrorClass | 31))

#define sysErrModuleNotFound sysErrLibNotFound

#define sysErrModuleRelocationError
    ((status_t)(sysErrorClass | 34))

#define sysErrNoFreeLibSlots
    ((status_t)(sysErrorClass | 9))

#define sysErrNoFreeRAM ((status_t)(sysErrorClass
    | 4))

#define sysErrNoFreeResource
    ((status_t)(sysErrorClass | 3))

#define sysErrNoGlobalStructure
    ((status_t)(sysErrorClass | 35))

#define sysErrNotAllowed ((status_t)(sysErrorClass
    | 5))
```

```
#define sysErrNotInitialized
    ((status_t)(sysErrorClass | 30))

#define sysErrOSVersion ((status_t)(sysErrorClass
    | 53))
    The program requires a higher version of the operating
    system in order to run.

#define sysErrOutOfOwnerIDs
    ((status_t)(sysErrorClass | 8))

#define sysErrParamErr ((status_t)(sysErrorClass |
    2))

#define sysErrPrefNotFound
    ((status_t)(sysErrorClass | 14))

#define sysErrRAMModuleNotAllowed
    ((status_t)(sysErrorClass | 39))
    A RAM-based module cannot be loaded when the device is
    booted into ROM-only mode

#define sysErrRomIncompatible
    ((status_t)(sysErrorClass | 12))

#define sysErrTimeout ((status_t)(sysErrorClass |
    1))

#define sysErrWeakRefGone
    ((status_t)(sysErrorClass | 52))
    The function is holding a weak reference, but the referred-to
    object is gone.

#define sysNotifyErrBroadcastBusy
    ((status_t)(sysErrorClass | 19))

#define sysNotifyErrBroadcastCancelled
    ((status_t)(sysErrorClass | 20))
```

## Common Error Codes

---

```
#define sysNotifyErrDuplicateEntry
    ((status_t)(sysErrorClass | 17))

#define sysNotifyErrEntryNotFound
    ((status_t)(sysErrorClass | 16))

#define sysNotifyErrNoServer
    ((status_t)(sysErrorClass | 21))

#define sysNotifyErrNoStackSpace
    ((status_t)(sysErrorClass | 29))

#define sysNotifyErrQueueEmpty
    ((status_t)(sysErrorClass | 28))

#define sysNotifyErrQueueFull
    ((status_t)(sysErrorClass | 27))
```

### **Purpose**

**Declared In** CmnErrors.h

**Constants** #define ECANCEL 5

#define ENOTSUP 45

#define EWOULDBLOCK EAGAIN

# Cyclic Redundancy Check

---

This chapter provides reference material for the CRC (Cyclic Redundancy Check) functions. It is divided into the following sections:

[CRC Functions and Macros](#) . . . . . 183

The header file `Crc.h` declares the API that this chapter describes.

## CRC Functions and Macros

### Crc16CalcBigBlock Function

**Purpose** Calculate the 16-bit CRC (Cyclic Redundancy Check) of a large data block (> 64K bytes) using the table lookup method. A CRC is one of many mathematical ways of checking data for corruption, similar to a checksum but mathematically more complex.

**Declared In** `Crc.h`

**Prototype** `uint16_t Crc16CalcBigBlock (void *bufP,  
uint32_t count, uint16_t crc)`

**Parameters**

- *bufP*  
Pointer to the data buffer.
- *count*  
Number of bytes in the buffer.
- *crc*  
Seed CRC value.

**Returns** A 16-bit CRC for the data buffer.

**See Also** [Crc16CalcBlock](#), [Crc32CalcBlock](#)

## Cyclic Redundancy Check

*Crc16CalcBlock*

---

### Crc16CalcBlock Function

**Purpose** Calculate the 16-bit CRC (Cyclic Redundancy Check) of a data block using the table lookup method. A CRC is one of many mathematical ways of checking data for corruption, similar to a checksum but mathematically more complex.

**Declared In**

**Declared In** `Crc.h`

**Prototype** `uint16_t Crc16CalcBlock (const void *bufP,  
uint16_t count, uint16_t crc)`

**Parameters**

- *bufP*  
Pointer to the data buffer.
- *count*  
Number of bytes in the buffer.
- *crc*  
Seed CRC value.

**Returns** A 16-bit CRC for the data buffer.

**Comments** The data block must be less than or equal to 64K bytes in length. For larger data blocks, use [Crc16CalcBigBlock](#). To obtain a 32-bit CRC value, use [Crc32CalcBlock](#).

### Crc32CalcBlock Function

**Purpose** Calculate the 32-bit CRC (Cyclic Redundancy Check) of a data block using the table lookup method. A CRC is one of many mathematical ways of checking data for corruption, similar to a checksum but mathematically more complex.

**Declared In** `Crc.h`

**Prototype** `uint32_t Crc32CalcBlock (const void *bufP,  
uint16_t count, uint32_t crc)`

**Parameters**

- *bufP*  
Pointer to the data buffer.
- *count*  
Number of bytes in the buffer.

→ *crc*

Seed CRC value.

**Returns** A 32-bit CRC for the data buffer.

**Comments** The data block must be less than or equal to 64K bytes in length.

**See Also** [Crc16CalcBigBlock](#), [Crc16CalcBlock](#)

## Cyclic Redundancy Check

*Crc32CalcBlock*

---

# DateTime

---

This chapter provides reference material for those APIs used to store and manipulate date and time values. It is organized as follows:

[DateTime Structures and Types](#) . . . . . 188

[DateTime Constants](#) . . . . . 191

[DateTime Functions and Macros](#). . . . . 202

The header file `DateTime.h` declares the API that this chapter describes. For more information on using the DateTime APIs, see [Chapter 11, “Time,”](#) on page 107.

## DateTime Structures and Types

### DateFormatType Typedef

|                    |                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains a <a href="#">DateFormatTag</a> value, which specifies a display format for date values. |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                           |
| <b>Prototype</b>   | <code>typedef Enum8 DateFormatType</code>                                                         |

### DateTimeType Struct

|                    |                                                                                                                                                                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Represents a date and time value.                                                                                                                                                                                 |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                                                                           |
| <b>Prototype</b>   | <pre>typedef struct {     int16_t second;     int16_t minute;     int16_t hour;     int16_t day;     int16_t month;     int16_t year;     int16_t weekDay; } DateTimeType typedef DateTimeType *DateTimePtr</pre> |

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| <b>Fields</b> | <code>second</code><br>The number of seconds. This is a value between 0 and 59. |
|               | <code>minute</code><br>The number of minutes. This is a value between 0 and 59. |
|               | <code>hour</code><br>The number of hours. This is a value between 0 and 23.     |
|               | <code>day</code><br>The day number. This is a value between 1 and 31.           |
|               | <code>month</code><br>The month number. This is a value between 1 and 12.       |
|               | <code>year</code><br>The year number.                                           |

weekDay

The day number. This represents the number of days since Sunday and is thus a value between 0 and 6.

## DateType Struct

|                    |                                                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Represents a date value.                                                                                                                                                              |
| <b>Declared In</b> | DateTime.h                                                                                                                                                                            |
| <b>Prototype</b>   | <pre>typedef struct {     uint16_t day :5;     uint16_t month :4;     uint16_t year :7; } DateType typedef DateType *DatePtr</pre>                                                    |
| <b>Fields</b>      | <p>day<br/>The day number. This is a value between 1 and 31.</p> <p>month<br/>The month number. This is a value between 1 and 12.</p> <p>year<br/>The number of years since 1904.</p> |

## DaylightSavingsTypes Typedef

|                    |                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains one of the forms of daylight savings times defined by the <a href="#">DaylightSavingsTag</a> enum. |
| <b>Declared In</b> | DateTime.h                                                                                                  |
| <b>Prototype</b>   | <pre>typedef Enum8 DaylightSavingsTypes</pre>                                                               |

## DayOfMonthType

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| <b>Purpose</b>     | Creates an alias for the <a href="#">DayOfWeekType</a> typedef. |
| <b>Declared In</b> | DateTime.h                                                      |
| <b>Prototype</b>   | <pre>#define DayOfWeekType DayOfMonthType</pre>                 |

### **DayOfWeekType Typedef**

- Purpose** Contains one of the day-of-the-week numeric values returned by the [DayOfMonth\(\)](#) function and enumerated in [DayOfWeekTag](#).
- Declared In** `DateTime.h`
- Prototype** `typedef Enum8 DayOfWeekType`

### **TimeFormatType Typedef**

- Purpose** Contains one of the different display formats for time values enumerated in [TimeFormatTag](#).
- Declared In** `DateTime.h`
- Prototype** `typedef Enum8 TimeFormatType`

### **TimeType Struct**

- Purpose** Represents a time value.
- Declared In** `DateTime.h`
- Prototype**

```
typedef struct {
    uint16_t hours : 8;
    uint16_t minutes : 8;
} TimeType
typedef TimeType *TimePtr
```
- Fields**
- `hours`  
The number of hours. This is a value between 0 and 23.
  - `minutes`  
The number of minutes. This is value between 0 and 59.

## DateTime Constants

### String Lengths

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | These constants represent the maximum lengths of strings returned by the date and time formatting routines <a href="#">DateToAscii()</a> , <a href="#">DateToDOWDMFormat()</a> , and <a href="#">TimeToAscii()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Constants</b>   | <pre>#define dateStringLength 15     Maximum length of the string returned by DateToAscii()     for short date formats.  #define dowDateStringLength 31     Maximum length of the string returned by     DateToDOWDMFormat() for short date formats.  #define dowLongDateStrLength 47     Maximum length of the string returned by     DateToDOWDMFormat() for both medium and long date     formats.  #define longDateStrLength 31     Maximum length of the string returned by DateToAscii()     for medium and long date formats.  #define timeStringLength 15     Maximum length of the string returned by TimeToAscii().  #define timeZoneStringLength 50     Maximum length of a descriptive string for a time zone as     returned by <a href="#">TimeZoneToAscii()</a> or <a href="#">TimeZoneToAsciiV50()</a>.</pre> |

### Months

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <b>Purpose</b>     | Constants that represent the months of the year. |
| <b>Declared In</b> | <code>DateTime.h</code>                          |
| <b>Constants</b>   | <pre>#define april 4 #define august 8</pre>      |

## DateTime

### Days

---

```
#define december 12
#define february 2
#define january 1
#define july 7
#define june 6
#define march 3
#define may 5
#define november 11
#define october 10
#define september 9
```

### Days

|                    |                                                                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Constants that represent the days of the week.                                                                                            |
| <b>Declared In</b> | DateTime.h                                                                                                                                |
| <b>Constants</b>   | <pre>#define friday 5 #define monday 1 #define saturday 6 #define sunday 0 #define thursday 4 #define tuesday 2 #define wednesday 3</pre> |

### Conversions

|                    |                                                                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Constants that define various intervals in terms of a smaller unit.                                               |
| <b>Declared In</b> | DateTime.h                                                                                                        |
| <b>Constants</b>   | <pre>#define daysInFourYears (daysInLeapYear + 3 *     daysInYear)</pre> <p>The number of days in four years.</p> |

```
#define daysInLeapYear 366
    The number of days in a leap year.
#define daysInSeconds (0x15180)
    The number of seconds in a day.
#define daysInWeek 7
    The number of days in a week.
#define daysInYear 365
    The number of days in a non-leap year.
#define hoursInMinutes 60
    The number of minutes in an hour.
#define hoursInSeconds (hoursInMinutes *
    minutesInSeconds)
    The number of seconds in an hour.
#define hoursPerDay 24
    The number of hours in a day.
#define minutesInSeconds 60
    The number of seconds in a minute.
#define monthsInYear 12
    The number of months in a year.
#define secondsInSeconds 1
    The number of seconds in a second.
```

## Template Formatting Characters

**Purpose** Characters that are used in conjunction with single-digit value types (declared in the [Template Value Types](#) enum) to construct formatting substrings for use with [DateTemplateToAscii\(\)](#). See that function for a complete description of how you specify date formatting in template strings.

**Declared In** `DateTime.h`

**Constants**

```
#define dateTemplateChar chrCircumflexAccent
    Character that marks the beginning of a formatting substring.
#define dateTemplateLeadZeroModifier 'z'
    Modifier that adds a leading zero to the formatted numeric value.
```

## DateTime

### Miscellaneous DateTime Constants

---

```
#define dateTemplateLongModifier 'l'
    Modifier that formats the value in long form,
#define dateTemplateRegularModifier 'r'
    Modifier that formats the value in regular form.
#define dateTemplateShortModifier 's'
    Modifier that formats the value in short form.
```

## Miscellaneous DateTime Constants

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The <code>DateTime.h</code> file also declares these constants.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Constants</b>   | <pre>#define firstYear 1904     The year upon which the year values in <a href="#">DateType</a> structures     are based.  #define lastYear (firstYear + numberOfYears - 1)     The greatest year that can be represented by a <a href="#">DateType</a>     structure.  #define maxDays ((uint32_t) numberOfYears / 4 *     daysInFourYears - 1)     The number of days in numberOfYears.  #define maxSeconds ((uint32_t) (maxDays + 1) *     daysInSeconds - 1)     The number of seconds in numberOfYears.  #define noTime -1     A time value that represents “no time.” This value is used, for     instance, when you create an appointment in the Date Book     application and specify “No Time” for the time of the     appointment.  #define numberOfYears 128     The <a href="#">DateType</a> structure uses 7 bits to represent the year (as     an offset from <code>firstYear</code>); this constant is the largest value     that can be represented by those 7 bits.</pre> |

## DateFormatTag Enum

|                    |                                                                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Specify the different display formats for date values. These values are typically contained within <a href="#">DateFormatType</a> variables.           |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                |
| <b>Constants</b>   | <code>dfMDYWithSlashes</code><br>The month, day, and year numbers separated by slashes. For example: 12/31/95. This is considered a short format.      |
|                    | <code>dfDMYWithSlashes</code><br>The day, month, and year numbers separated by slashes. For example: 31/12/95. This is considered a short format.      |
|                    | <code>dfDMYWithDots</code><br>The day, month, and year numbers separated by dots. For example, 31.12.95. This is considered a short format.            |
|                    | <code>dfDMYWithDashes</code><br>The day, month, and year numbers separated by dashes. For example, 31-12-95. This is considered a short format.        |
|                    | <code>dfYMDWithSlashes</code><br>The year, month, and day numbers separated by slashes. For example, 95/12/31. This is considered a short format.      |
|                    | <code>dfYMDWithDots</code><br>The year, month, and day numbers separated by dots. For example, 95.12.31. This is considered a short format.            |
|                    | <code>dfYMDWithDashes</code><br>The year, month, and day numbers separated by dashes. For example, 95-12-31. This is considered a short format.        |
|                    | <code>dfMDYLongWithComma</code><br>The month, day, and year in long format, with a comma. For example, Dec 31, 1995. This is considered a long format. |
|                    | <code>dfDMYLong</code><br>The month, day, and year in long format. For example, 31 Dec 1995. This is considered a long format.                         |
|                    | <code>dfDMYLongWithDot</code><br>The month, day, and year in long format, with a dot. For example, 31. Dec 1995. This is considered a long format.     |

## DateTime

### Template Value Types

---

#### dfDMYLongNoDay

The month and year in long format. For example, Dec 1995. This is considered a long format.

#### dfDMYLongWithComma

The day, month, and year in long format, with a comma. For example, 31 Dec, 1995. This is considered a long format.

#### dfYMDLongWithDot

The year, month, and day in long format with dot separators. For example, 1995.12.31. This is considered a long format.

#### dfYMDLongWithSpace

The year, month, and day in long format with space separators. For example, 1995 Dec 31. This is considered a long format.

#### dfMYMed

The month in long format with the two-digit year, preceded by an apostrophe. For example, Dec '95. This is considered a medium format.

#### dfMYMedNoPost

The month in long format with the two-digit year. For example, Dec 95. This is considered a medium format.

#### dfMDYWithDashes

The month, day, and year numbers separated by dashes. For example, 12-31-95. This is considered a short format.

### Compatibility

These values are provided for compatibility purposes only. ARM-native application developers shouldn't specify the format of dates directly (and thus shouldn't use these enum values). Instead, use the preference selected by the user (such as `prefDateFormat`; see [SystemPreferencesChoice](#)). If you need to use a format appropriate for a specific locale, ask the Locale Manager for that format. If you need more control than that, format the date yourself in such a way as to allow it to be localized.

## Template Value Types Enum

### Purpose

Values that specify portions of a date, used along with [Template Formatting Characters](#) to construct formatting substrings for use

with [DateTemplateToAscii\(\)](#). See that function for a complete description of how you specify date formatting in template strings.

**Declared In** `DateTime.h`

**Constants**

- `dateTemplateDayNum = '0'`  
The day number. For example, "1", "01", "23", or "31".
- `dateTemplateDOWName`  
The day name. For example "Tue" or "Tuesday".
- `dateTemplateMonthName`  
The month name. For example, "May", "Aug", or "August".
- `dateTemplateMonthNum`  
The number of the month. For example, "4", "04", or "11".
- `dateTemplateYearNum`  
The year. For example, "97" or "1997".

## DaylightSavingsTag Enum

**Purpose** The `DaylightSavingsTypes` enum specifies the different forms of daylight savings times that you can specify for date and time values. Use [DaylightSavingsTypes](#) variables to contain daylight savings types values.

**Declared In** `DateTime.h`

**Constants**

- `dsNone`  
No DST (daylight savings time)
- `dsUSA`  
U.S.A. DST
- `dsAustralia`  
Australian DST
- `dsWesternEuropean`  
Western European DST
- `dsMiddleEuropean`  
Middle European DST
- `dsEasternEuropean`  
Eastern European DST

## DateTime

### DayOfWeekTag

---

dsGreatBritain  
Great Britain and Eire DST

dsRumania  
Rumanian DST

dsTurkey  
Turkish DST

dsAustraliaShifted  
Australian DST, with the 1986 shift

**Comments** Palm OS represents daylight savings time as an integer value that gives the number of minutes to add to the current time for daylight savings time.

## DayOfWeekTag Enum

**Purpose** Specifies the different day-of-the-week numeric values that are returned by the [DayOfMonth\(\)](#) function. These values are used to represent repeating appointments that occur on specific days of the month; for example, the first Friday or the third Tuesday of each month. Variables that contain these values should be declared as [DayOfWeekType](#).

**Declared In** DateTime.h

**Constants** dom1stSun  
The first Sunday of the month.

dom1stMon  
The first Monday of the month.

dom1stTue  
The first Tuesday of the month.

dom1stWen  
The first Wednesday of the month.

dom1stThu  
The first Thursday of the month.

dom1stFri  
The first Friday of the month.

dom1stSat  
The first Saturday of the month.

dom2ndSun  
The second Sunday of the month.

dom2ndMon  
The second Monday of the month.

dom2ndTue  
The second Tuesday of the month.

dom2ndWen  
The second Wednesday of the month.

dom2ndThu  
The second Thursday of the month.

dom2ndFri  
The second Friday of the month.

dom2ndSat  
The second Saturday of the month.

dom3rdSun  
The third Sunday of the month.

dom3rdMon  
The third Monday of the month.

dom3rdTue  
The third Tuesday of the month.

dom3rdWen  
The third Wednesday of the month.

dom3rdThu  
The third Thursday of the month.

dom3rdFri  
The third Friday of the month.

dom3rdSat  
The third Saturday of the month.

dom4thSun  
The fourth Sunday of the month.

dom4thMon  
The fourth Monday of the month.

dom4thTue  
The fourth Tuesday of the month.

## DateTime

### TimeFormatTag

---

|                         |                                    |
|-------------------------|------------------------------------|
| <code>dom4thWen</code>  | The fourth Wednesday of the month. |
| <code>dom4thThu</code>  | The fourth Thursday of the month.  |
| <code>dom4thFri</code>  | The fourth Friday of the month.    |
| <code>dom4thSat</code>  | The fourth Saturday of the month.  |
| <code>domLastSun</code> | The last Sunday of the month.      |
| <code>domLastMon</code> | The last Monday of the month.      |
| <code>domLastTue</code> | The last Tuesday of the month.     |
| <code>domLastWen</code> | The last Wednesday of the month.   |
| <code>domLastThu</code> | The last Thursday of the month.    |
| <code>domLastFri</code> | The last Friday of the month.      |
| <code>domLastSat</code> | The last Saturday of the month.    |

## TimeFormatTag Enum

|                    |                                                                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Specifies the different display formats for time values. Variables that contain these values should be declared as <a href="#">TimeFormatType</a> .                                                                                                             |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                                                                                                                         |
| <b>Constants</b>   | <code>tfColon</code><br>The hour and minutes separated by a colon character. For example, <code>1:00</code> .<br><code>tfColonAMPM</code><br>The hour and minutes separated by a colon and followed by an AM/PM indication. For example, <code>1:00 pm</code> . |

`tfColon24h`

The 24-hour time with the hour and minutes separated by a colon character. For example, 13:00.

`tfDot`

The hour and minutes separated by a dot character. For example, 1.00.

`tfDotAMPM`

The hour and minutes separated by a period and followed by an AM/PM indication. For example, 1.00 pm.

`tfDot24h`

The 24-hour time with the hour and minutes separated by a dot character. For example, 13.00.

`tfHoursAMPM`

The hour value followed by an AM/PM indication. For example, 1 pm.

`tfHours24h`

The 24-hour value. For example, 13.

`tfComma24h`

The 24-hour time with the hour and minutes separated by a comma character. For example, 13,00.

**Compatibility**

These values are provided for compatibility purposes only. ARM-native application developers shouldn't specify the format of times directly (and thus shouldn't use these enum values). Instead, use the preference selected by the user (such as `prefTimeFormat`; see [SystemPreferencesChoice](#)). If you need to use a format appropriate for a specific locale, ask the Locale Manager for that format. If you need more control than that, format the time yourself in such a way as to allow it to be localized.

## DateTime Functions and Macros

### DateAdjust Function

- Purpose** Add or subtract a specified number of days from a given date.
- Declared In** `DateTime.h`
- Prototype** `void DateAdjust (DatePtr dateP,  
int32_t adjustment)`
- Parameters**
- $\leftrightarrow$  *dateP*  
A pointer to a [DateType](#) structure with the date to be adjusted.
  - $\rightarrow$  *adjustment*  
The number of days by which to adjust the date.
- Returns** Returns nothing. Upon return, *dateP* contains the adjusted date.
- Comments** This function adjusts the date by the specified number of days and manages month and year wrapping conditions.
- See Also** [TimAdjust\(\)](#)

### DateDaysToDate Function

- Purpose** Converts a date specified as the number of days since January 1, 1904 to a [DateType](#) structure.
- Declared In** `DateTime.h`
- Prototype** `void DateDaysToDate (uint32_t days, DatePtr date)`
- Parameters**
- $\rightarrow$  *days*  
The number of days since 1/1/1904.
  - $\leftarrow$  *date*  
A pointer to a [DateType](#) structure that receives the computed date values.
- Returns** Returns nothing. Upon return, the date information is returned in the structure referenced by *date*.
- See Also** [DateSecondsToDate\(\)](#), [DateToDays\(\)](#)

## DateSecondsToDate Function

- Purpose** Converts a date specified as the number of seconds since January 1, 1904 to a [DateType](#) structure.
- Declared In** `DateTime.h`
- Prototype**

```
void DateSecondsToDate (uint32_t seconds,
                        DatePtr date)
```
- Parameters**
- *seconds*  
The number of seconds since 1/1/1904.
  - ← *date*  
A pointer to a [DateType](#) structure that receives the computed date values.
- Returns** Returns nothing. Upon return, the date information is returned in the structure referenced by *date*.
- See Also** [DateDaysToDate\(\)](#), [DateToDays\(\)](#)

## DateTemplateToAscii Function

- Purpose** Convert the specified date values into a string that is formatted according to a formatting template specification.
- Declared In** `DateTime.h`
- Prototype**

```
uint16_t DateTemplateToAscii
(const char *templateP, uint8_t months,
 uint8_t days, uint16_t years, char *stringP,
 int16_t stringLen)
```
- Parameters**
- *templateP*  
A pointer to the template string used to format the date. See the Comments section below for details on how to specify date formatting in this template string.
  - *months*  
The month number, which must be a value between 1 and 12.
  - *days*  
The day number, which must be a value between 1 and 31.
  - *years*  
The four-digit year number. For example, 1995.

## DateTime

### DateTemplateToAscii

---

← *stringP*

A pointer to a string that is updated with the result. If *stringP* is NULL, this function does not write an output string; however, it does return the length required for the output string. If *stringP* is not NULL, this function writes up to *stringLen* bytes into *stringP*.

→ *stringLen*

The size of the *stringP* buffer.

**Returns** The length of the formatted string (whether or not *stringP* is NULL), up to but not including the null terminator.

**Comments** This function is intended as a replacement for the [DateToAscii\(\)](#) and [DateToDOWDMFormat\(\)](#) functions.

This function uses the formatting template referenced by *templateP* to create a formatted string from the date values that you pass in.

You specify a series of formatting substrings in *templateP*. Each substring has the form:

`^<valueType><formatModifier>`

Each substring has three components:

- The ^ character begins a substring.
- The <valueType> component is a single-digit value that specifies the value type. The [Template Value Types](#) enum declares constants that represent these values.
- The <formatModifier> component is a single-letter value that specifies how you want that value formatted. See [“Template Formatting Characters”](#) on page 193 for a set of constants that correspond to these formatting characters.

The following is an example of a template specification with three substrings:

```
^0z ^2l ^4r
```

[Table 20.1](#) shows the values you can specify for the <valueType> component. Note that the formatted result depends on the <modifier> value.

**Table 20.1 Template value types for the DateTemplateToAscii function**

| Value | Value type   | Formatted examples |
|-------|--------------|--------------------|
| 0     | Day number   | 1, 01, 23, 31      |
| 1     | Day name     | Tue, Tuesday       |
| 2     | Month name   | May, Aug, August   |
| 3     | Month number | 4, 04, 11          |
| 4     | Year number  | 97, 1997           |

[Table 20.2](#) shows the values you can specify for the <modifier> component of each template substring.

**Table 20.2 Template modifier types for the DateTemplateToAscii function**

| Modifier | Description                                        |
|----------|----------------------------------------------------|
| s        | Formats the value in short form                    |
| r        | Formats the value in regular form                  |
| l        | Formats the value in long form                     |
| z        | Adds a leading zero to the formatted numeric value |

Finally, [Table 20.3](#) shows examples of each value type formatted with each modifier type.

**Table 20.3 Examples of formatted values**

| Value type     | Raw value | s (Short format) | r (Regular format) | l (Long format) | z (Zero format) |
|----------------|-----------|------------------|--------------------|-----------------|-----------------|
| 0 (Day number) | 2         | 2                | 2                  | 2               | 02              |
| 1 (Day name)   | 2         | T                | Tue                | Tuesday         | n/a             |

## DateTime

### DateToAscii

---

**Table 20.3 Examples of formatted values (*continued*)**

| Value type       | Raw value | s (Short format) | r (Regular format) | l (Long format) | z (Zero format) |
|------------------|-----------|------------------|--------------------|-----------------|-----------------|
| 2 (Month name)   | 11        | N                | Nov                | November        | n/a             |
| 3 (Month number) | 11        | 11               | 11                 | 11              | 11              |
| 4 (Year number)  | 2000      | 00               | 2000               | 2000            | n/a             |

**Example** Calling `DateTemplateToAscii` as follows:

```
DateTemplateToAscii("^0z ^2l ^4r", 2, 7, 2000, myStr, 20)
```

Produces the following formatted string:

```
07 February 2000
```

**See Also** `DateToAscii()`, `DateToDOWDMFormat()`

## DateToAscii Function

**Purpose** Convert the passed date to a string using the format specified by the `dateFormat` parameter.

**Declared In** `DateTime.h`

**Prototype**

```
void DateToAscii (uint8_t months, uint8_t days, uint16_t years, DateFormatType dateFormat, char *pString)
```

**Parameters**

- *months*  
The month number, which must be a value between 1 and 12.
- *days*  
The day number, which must be a value between 1 and 31.
- *years*  
The four-digit year number. For example, 1995.

→ *dateFormat*

Any [DateFormatType](#) format.

→ *pString*

A pointer to string that is updated with the result. This string must be of length `dateStringLength` for short formats or `longDateStrLength` for medium or long formats. Note that these lengths include the terminating null byte. For more information about required string lengths, see “[DateTime Constants](#)” on page 191.

**Returns** Returns nothing. The string reference by *pString* is updated with the formatted string.

**Comments** If you are using a debug ROM, the string buffer is filled with either `dateStringLength` or `longStrLength` debugging bytes, depending on the value of the `dateFormat` parameter.

It is important to allocate enough space for your string buffer. Finding buffer overflow errors can be difficult when using a debug ROM. One common situation is when you pass a buffer that is too small from a form, for an element such as a label or title. Then, the buffer overflow can clobber objects that follow the form in memory. When a form element’s location information is corrupted, it disappears from the display.

**See Also** [TimeToAscii\(\)](#), [DateToDOWDMFormat\(\)](#),  
[DateTemplateToAscii\(\)](#)

## DateToDays Function

**Purpose** Convert the [DateType](#) structure to the number of days elapsed from January 1, 1904.

**Declared In** `DateTime.h`

**Prototype** `uint32_t DateToDays (DateType date)`

**Parameters** → *date*

A [DateType](#) structure.

**Returns** Returns the number of days elapsed from January 1, 1904 to the specified date.

**See Also** [DateDaysToDate\(\)](#)

## DateTime

### DateToDOWDMFormat

---

## DateToDOWDMFormat Function

- Purpose** Convert a date to a formatted string using a specified format. The resultant string includes the name of the day of the week.
- Declared In** `DateTime.h`
- Prototype**  
`void DateToDOWDMFormat (uint8_t months,  
uint8_t days, uint16_t years,  
DateFormatType dateFormat, char *pString)`
- Parameters**
- *months*  
The month number, which must be a value between 1 and 12.
  - *days*  
The day number, which must be a value between 1 and 31.
  - *years*  
The four-digit year number. For example, 1995.
  - *dateFormat*  
Any [DateFormatType](#) format.
  - ← *pString*  
A pointer to a string that is updated with the result. The string must be of length `dowDateStringLength` for short formats or `dowLongDateStrLength` for medium or long date formats.
- Returns** Returns nothing. The string referenced by `pString` is updated with the formatted string.
- Comments** The values of some of the [DateTime Constants](#) that specify the required string buffer lengths do change from time to time. You should always use the constants or verify the required lengths by checking the `DateTime.h` file.
- It is important to allocate enough space for your string buffer. Finding buffer overflow errors can be difficult when using a debug ROM. One common situation is when you pass a buffer that is too small from a form, for an element such as a label or title. Then, the buffer overflow can clobber objects that follow the form in memory. When a form element's location information is corrupted, it disappears from the display.
- See Also** [DateToAscii\(\)](#), [DateTemplateToAscii\(\)](#)

## DateToInt Macro

|                    |                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Convert a date in a <a href="#">DateType</a> structure to an unsigned integer.                         |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                |
| <b>Prototype</b>   | <code>#define DateToInt (date)</code>                                                                  |
| <b>Parameters</b>  | <code>→ date</code><br>A <a href="#">DateType</a> structure containing the date value to be converted. |
| <b>Returns</b>     | The date as an unsigned 16-bit integer of type <code>uint16_t</code> .                                 |
| <b>See Also</b>    | <a href="#">TimeToInt()</a>                                                                            |

## DayOfMonth Function

|                    |                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Determine the day of a month on which the specified date occurs. The value returned by this function represents a quantity such as “First Monday” or “Third Friday” as is used for repeating appointments in the Datebook.                  |
| <b>Declared In</b> | <code>DateTime.h</code>                                                                                                                                                                                                                     |
| <b>Prototype</b>   | <code>int16_t DayOfMonth (int16_t month, int16_t day, int16_t year)</code>                                                                                                                                                                  |
| <b>Parameters</b>  | <code>→ month</code><br>The month number, which must be a value between 1 and 12.<br><code>→ day</code><br>The day number, which must be a value between 1 and 31.<br><code>→ year</code><br>The four-digit year number. For example, 1995. |
| <b>Returns</b>     | Returns one of the <a href="#">DayOfWeekType</a> values that represents the day of the month.                                                                                                                                               |
| <b>Comments</b>    | The returns value can be used to specify on which day of the month an appointment repeats.                                                                                                                                                  |

## DayOfWeek Function

- Purpose** Determine the day of the week value for a specified date.
- Declared In** `DateTime.h`
- Prototype** `int16_t DayOfWeek (int16_t month, int16_t day, int16_t year)`
- Parameters**
- *month*  
The month number, which must be a value between 1 and 12.
  - *day*  
The day number, which must be a value between 1 and 31.
  - *year*  
The four-digit year number. For example, 1995.
- Returns** Returns one of the values listed under “[Days](#)” on page 192.

## DaysInMonth Function

- Purpose** Return the number of days in the month.
- Declared In** `DateTime.h`
- Prototype** `int16_t DaysInMonth (int16_t month, int16_t year)`
- Parameters**
- *month*  
The month number, which must be a value between 1 and 12.
  - *year*  
The four-digit year number. For example, 1995.
- Returns** Returns the number of days in the month for the specified year.

## TimAdjust Function

- Purpose** Add or subtract a specified number of seconds from a given time and date.
- Declared In** `DateTime.h`
- Prototype** `void TimAdjust (DateTimePtr dateTimeP,  
int32_t adjustment)`
- Parameters**  $\leftrightarrow$  *dateTimeP*  
A pointer to a [DateType](#) structure containing the time and date to be adjusted.
- $\rightarrow$  *adjustment*  
The number of seconds by which to adjust the time.
- Returns** Returns nothing. The structure referenced by *dateTimeP* is modified to contain the updated date and time.
- Comments** This function advances the time by the specified number of seconds and takes care of any wraparound conditions.
- See Also** `DateAdjust()`

## TimDateTimeToSeconds Function

- Purpose** Determine the number of seconds elapsed from 12:00 A.M. on January 1, 1904 to the specified date and time.
- Declared In** `DateTime.h`
- Prototype** `uint32_t TimDateTimeToSeconds  
(DateTimePtr dateTimeP)`
- Parameters**  $\rightarrow$  *dateTimeP*  
A pointer to a [DateTimeType](#) structure containing a date and time.
- Returns** The number of seconds elapsed from 12:00 A.M. on January 1, 1904 to the date referenced by *dateTimeP*.
- See Also** `TimSecondsToDateTime()`

## TimeGetFormatSeparator Function

- Purpose** Get the time format separator (such as ':') used by a specified time format.
- Declared In** `DateTime.h`
- Prototype** `wchar32_t TimeGetFormatSeparator  
(TimeFormatType timeFormat)`
- Parameters** → *timeFormat*  
The time format.
- Returns** Returns the separator character that the specified time format uses.
- Comments** If the time format uses a multi-character time separator, this function returns only the first character of the separator.
- See Also** [TimeGetFormatSuffix\(\)](#), [TimeIs24HourFormat\(\)](#)

## TimeGetFormatSuffix Function

- Purpose** Obtain the time format suffix (such as "am" or "pm) that's appropriate given a time format and a time.
- Declared In** `DateTime.h`
- Prototype** `Boolean TimeGetFormatSuffix  
(TimeFormatType timeFormat, uint8_t hours,  
char *suffixStr)`
- Parameters** → *timeFormat*  
The time format.
- *hours*  
The time, in hours.
- ← *suffixStr*  
Pointer to a character buffer into which the time format suffix is written. If the specified time format doesn't have a suffix, a single null-terminator character is written.
- Returns** Returns `true` if a valid suffix string is returned (that is, one that is not simply a null-terminator). Otherwise, this function returns `false`.
- See Also** [TimeGetFormatSeparator\(\)](#), [TimeIs24HourFormat\(\)](#)

## TimeIs24HourFormat Function

- Purpose** Returns whether the specified time format is a 24-hour format, as opposed to a 12-hour, AM/PM format.
- Declared In** `DateTime.h`
- Prototype** `Boolean TimeIs24HourFormat (TimeFormatType timeFormat)`
- Parameters** → *timeFormat*  
The time format.
- Returns** Returns `true` if the specified time format is a 24-hour format, `false` otherwise.
- See Also** [TimeGetFormatSeparator\(\)](#)

## TimeToAscii Function

- Purpose** Convert the time to a string that is formatted according to the specified time format.
- Declared In** `DateTime.h`
- Prototype** `void TimeToAscii (uint8_t hours, uint8_t minutes, TimeFormatType timeFormat, char *pString)`
- Parameters**
- *hours*  
The number of hours. This must be a value between 0 and 23.
  - *minutes*  
The number of minutes. This must be a value between 0 and 59.
  - *timeFormat*  
The time format for the resultant string. This must be one of the [TimeFormatType](#) values.
  - *pString*  
A pointer to a string that is updated with the resultant string. This string must be of length `timeStringLength`. See “[DateTime Constants](#)” on page 191 for information on string buffer lengths.
- Returns** Returns nothing. The string referenced by *pString* is updated with the formatted string.

## DateTime

### TimeToInt

---

**Comments** It is important to allocate enough space for your string buffer. Finding buffer overflow errors can be difficult when using a debug ROM. One common situation is when you pass a buffer that is too small from a form, for an element such as a label or title. Then, the buffer overflow can clobber objects that follow the form in memory. When a form element's location information is corrupted, it disappears from the display.

**See Also** [DateToAscii\(\)](#)

## TimeToInt Macro

**Purpose** Convert a time in a [TimeType](#) structure to a signed integer.

**Declared In** `DateTime.h`

**Prototype** `#define TimeToInt (time)`

**Parameters** `→ time`  
A [TimeType](#) structure containing the time value to be converted.

**Returns** The time as a signed 16-bit integer of type `int16_t`.

**See Also** [DateToInt\(\)](#)

## TimeZoneToAscii Function

**Purpose** Convert a time zone ID to an ASCII string.

**Declared In** `DateTime.h`

**Prototype** `void TimeZoneToAscii (const char *timeZoneID, char *string)`

**Parameters** `→ timeZoneID`  
Time zone ID. This is one of the time zone ID strings found in the UI Library's `TimeZoneSet.xrd` file. For instance, "Asia/Kabul".

`← string`  
A pointer to a string in which to return the result. This string must be of length `timeZoneStringLength`.

**Returns** Returns nothing.

**Comments** This function returns a descriptive string for the specified time zone. This string identifies the time zone first by its country, such as “USA (Mountain)” or “Canada (Eastern).” If the function cannot find a time zone that matches the specified GMT offset and country, it returns a string containing the time zone as a numeric offset from the GMT (for example, “GMT+9:00”).

## TimeZoneToAsciiV50 Function

**Purpose** Convert a time zone to a string.

**Declared In** `DateTime.h`

**Prototype** `void TimeZoneToAsciiV50 (int16_t timeZone,  
const LmLocaleType *localeP, char *string)`

**Parameters** → *timeZone*

The time zone, given as minutes east of Greenwich Mean Time (GMT).

→ *localeP*

A pointer to a locale (see `LmLocaleType`) that identifies the time zone country. You can use the constant `lmAnyLanguage` as the value for the language field of the structure pointed to by this parameter.

← *string*

A pointer to a string in which to return the result. This string must be of length `timeZoneStringLength`.

**Returns** Returns nothing.

**Comments** This function returns a descriptive string for the specified time zone. This string identifies the time zone first by its country, such as “USA (Mountain)” or “Canada (Eastern).” If the function cannot find a time zone that matches the specified GMT offset and country, it returns a string containing the time zone as a numeric offset from the GMT (for example, “GMT+9:00”).

**Compatibility** This function is provided for compatibility purposes only; applications should use [TimeZoneToAscii\(\)](#) instead.

## DateTime

### *TimSecondsToDateTime*

---

#### TimSecondsToDateTime Function

- Purpose** Converts a date specified as the number of seconds since January 1, 1904 to a [DateTimeType](#) structure.
- Declared In** `DateTime.h`
- Prototype**  
`void TimSecondsToDateTime (uint32_t seconds,  
DateTimePtr dateTimeP)`
- Parameters**
- *seconds*  
A date specified as the number of seconds elapsed from 12:00 A.M. on January 1, 1904 to the date
  - ← *dateTimeP*  
A pointer to a [DateTimeType](#) structure that is updated with the date and time values.
- Returns** Returns nothing. The structure referenced by `dateTimeP` is updated with the date and time computed for the number of seconds since 12:00 A.M. on January 1, 1904.
- See Also** [TimDateTimeToSeconds\(\)](#)

#### TimTimeZoneToUTC Function

- Purpose** Converts a date and time from a given time zone to Universal Coordinated Time (UTC). UTC is also known as Greenwich Mean Time (GMT).
- Declared In** `DateTime.h`
- Prototype**  
`uint32_t TimTimeZoneToUTC (uint32_t seconds,  
int16_t timeZone,  
int16_t daylightSavingAdjustment)`
- Parameters**
- *seconds*  
The number of seconds since 12:00 A.M. on January 1, 1904.
  - *timeZone*  
The time zone, given as the number of minutes east of UTC. For time zones west of UTC but before the international dateline, this is a negative number.
  - *daylightSavingAdjustment*  
The number of minutes to add to the current time for daylight savings time in this time zone.

**Returns** Returns the same time as `seconds` but in the Universal Coordinated Time. The value is still given as the number of seconds since 12:00 A.M. on January 1, 1904.

**Comments** The returned value is not necessarily the time in Greenwich because Greenwich may be observing daylight saving time.

**Example** You can use this function to convert the local time to UTC. The time zone and the daylight savings adjustment are system preferences that can be retrieved using [PrefGetPreference\(\)](#). For example, the following code converts the current local time to UTC:

```
int16_t timeZone =
    PrefGetPreference(prefTimeZone);
int16_t daylightSavingAdjustment =
    PrefGetPreference(
        prefDaylightSavingAdjustment);
uint32_t utcTime =
    TimTimeZoneToUTC(TimGetSeconds(), timeZone,
        daylightSavingAdjustment);
```

**See Also** [TimUTCToTimeZone\(\)](#)

## TimUTCToTimeZone Function

**Purpose** Converts a date and time from Universal Coordinated Time (UTC) to the specified time zone. UTC is also known as Greenwich Mean Time (GMT).

**Declared In** `DateTime.h`

**Prototype** `uint32_t TimUTCToTimeZone (uint32_t seconds,  
int16_t timeZone,  
int16_t daylightSavingAdjustment)`

**Parameters** → *seconds*  
The number of seconds since 12:00 A.M. on January 1, 1904 in UTC.

→ *timeZone*  
The time zone, given as the number of minutes east of UTC. For time zones west of UTC before the international dateline, this is a negative number.

## DateTime

*TimUTCToTimeZone*

---

→ *daylightSavingAdjustment*

The number of minutes to add to the current time for daylight savings time in this time zone.

**Returns** Returns the same time as *seconds* but in the specified time zone. The value is still given as the number of seconds since 12:00 A.M. on January 1, 1904.

**Comments** The *seconds* value is not necessarily the time in Greenwich because Greenwich may be observing daylight saving time.

**See Also** [TimTimeZoneToUTC\(\)](#)

# Debug Manager

---

This chapter provides reference material for the Debug Manager. It is organized as follows:

[Debug Manager Functions and Macros](#) . . . . . 219

The header file `DebugMgr.h` declares the API that this chapter describes.

## Debug Manager Functions and Macros

### DbgBreak Function

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Connects to the external debugger and halts execution.                                                    |
| <b>Declared In</b> | <code>DebugMgr.h</code>                                                                                   |
| <b>Prototype</b>   | <code>void DbgBreak (void)</code>                                                                         |
| <b>Parameters</b>  | None.                                                                                                     |
| <b>Returns</b>     | Returns nothing.                                                                                          |
| <b>Comments</b>    | This function currently halts the entire device. Future implementations may halt only the calling thread. |
| <b>See Also</b>    | <a href="#">DbgBreakMessage()</a> , <a href="#">DbgBreakMessageIf()</a>                                   |

### **DbgBreakMessage Macro**

- Purpose** Sends a null-terminated string to the current debug device, then connects to the external debugger and halts execution.
- Declared In** `DebugMgr.h`
- Prototype** `#define DbgBreakMessage (message)`
- Parameters** `→ message`  
The string to be sent to the current debug device.
- Returns** Returns nothing.
- Comments** This macro uses [DbgMessage\(\)](#) to send the string to the current debug device and [DbgBreak\(\)](#) to halt execution.

### **DbgBreakMessageIf Macro**

- Purpose** If a specified condition evaluates to `true`, sends a null-terminated string to the current debug device, then connects to the external debugger and halts execution.
- Declared In** `DebugMgr.h`
- Prototype** `#define DbgBreakMessageIf (condition, message)`
- Parameters** `→ condition`  
If this argument evaluates to `true`, the macro sends a null-terminated string to the current debug device, then connects to the external debugger and halts execution. Otherwise, this macro does nothing.
- `→ message`  
The string to be sent to the current debug device.
- Returns** Returns nothing.
- Comments** This macro uses [DbgMessage\(\)](#) to send the string to the current debug device and [DbgBreak\(\)](#) to halt execution.

## DbgFatalErrorInContext Function

|                    |                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                                                                                       |
| <b>Declared In</b> | DebugMgr.h                                                                                                                                                            |
| <b>Prototype</b>   | <pre>uint32_t DbgFatalErrorInContext     (const char *fileName, uint32_t lineNumber,      const char *errMsg, uint32_t options,      uint32_t allowedResponses)</pre> |
| <b>Parameters</b>  | <p>→ <i>fileName</i></p> <p>→ <i>lineNum</i></p> <p>→ <i>errMsg</i></p> <p>→ <i>options</i></p> <p>→ <i>allowedResponses</i></p>                                      |
| <b>Returns</b>     |                                                                                                                                                                       |
| <b>Comments</b>    | Low-level error reporting: this gives a few options to the developer such as ignore, break, etc.                                                                      |

## DbgGetChar Function

|                    |                                                                                |
|--------------------|--------------------------------------------------------------------------------|
| <b>Purpose</b>     | Reads a single character from the debug device.                                |
| <b>Declared In</b> | DebugMgr.h                                                                     |
| <b>Prototype</b>   | <pre>char DbgGetChar (void)</pre>                                              |
| <b>Parameters</b>  | None.                                                                          |
| <b>Returns</b>     | Returns the character read from the debug device.                              |
| <b>Comments</b>    | On some devices, this may block all threads until the character has been read. |

### DbgIsPresent Function

- Purpose** Determine whether the debugger is connected.
- Declared In** `DebugMgr.h`
- Prototype** `Boolean DbgIsPresent (void)`
- Parameters** None.
- Returns** Returns `true` if the debugger is connected, `false` otherwise.

### DbgLookupSymbol Function

- Purpose** Obtain the raw symbol name for a given function.
- Declared In** `DebugMgr.h`
- Prototype** `status_t DbgLookupSymbol (const void *addr, int32_t maxLen, char *outSymbol, void **outAddr)`
- Parameters**
- *addr*  
The address of the function for which the raw symbol name is to be returned.
  - *maxLen*  
The size of the buffer into which the raw symbol name is to be written.
  - ← *outSymbol*  
The raw symbol name of the specified function.
  - ← *outAddr*
- Returns**
- Compatibility** This function is only implemented for the Palm OS Simulator.
- See Also** [DbgUnmangleSymbol\(\)](#)

## DbgMessage Function

- Purpose** Sends a null-terminated string to the current debug device.
- Declared In** `DebugMgr.h`
- Prototype** `void DbgMessage (const char *iStringP)`
- Parameters** `→ iStringP`  
The string to be sent to the current debug device.
- Returns** Returns nothing.
- Comments** The string is *not* guaranteed to be sent atomically; concurrent `DbgMessage()` calls may result in interleaved output unless some mutual exclusion is provided by the caller.
- See Also** [DbgPrintf\(\)](#), [DbgVPrintf\(\)](#)

## DbgOutputSync Function

- Purpose** Forces the output stream to be flushed if the output device is buffered.
- Declared In** `DebugMgr.h`
- Prototype** `void DbgOutputSync (void)`
- Parameters** None.
- Returns** Returns nothing.
- Comments** The call does not return until all pending output has been written out. If the output device is not buffered, this call is a no-op.

## DbgPrintf Function

- Purpose** Sends a string to the current debug device. The string is constructed using `sprintf()`; accordingly, you pass this function a format string and a set of arguments.
- Declared In** `DebugMgr.h`
- Prototype** `long DbgPrintf (const char *iFormat, ...)`
- Parameters** `→ iFormat`  
Format string, as specified by `sprintf()`.

## Debug Manager

### *DbgRestartMallocProfiling*

---

→ ...

Zero or more argument to be written, formatted according to *iFormat*, to the current debug device.

**Returns** Returns the number of bytes written to the current debug device.

**Comments** This function builds a string, up to 200 characters in length, using `sprintf()`, and then sends it out using [DbgMessage\(\)](#). Formatted strings in excess of 200 characters will be truncated.

**See Also** [DbgMessage\(\)](#), [DbgVPrintf\(\)](#)

## DbgRestartMallocProfiling Function

**Purpose** Clear out all `malloc()` profiling statistics that have been collected so far for the current process.

**Declared In** `DebugMgr.h`

**Prototype** `void DbgRestartMallocProfiling (void)`

**Parameters** None.

**Returns** Nothing.

**See Also** [DbgSetMallocProfiling\(\)](#)

## DbgSetMallocProfiling Function

**Purpose**

**Declared In** `DebugMgr.h`

**Prototype** `void DbgSetMallocProfiling (Boolean enabled,  
int32_t dumpPeriod, int32_t maxItems,  
int32_t stackDepth)`

**Parameters** → *enabled*

→ *dumpPeriod*

→ *maxItems*

→ *stackDepth*

**Returns**

**See Also** [DbgRestartMallocProfiling\(\)](#)

## DbgUnmangleSymbol Function

**Purpose** Get the corresponding unmangled name for a symbol.

**Declared In** `DebugMgr.h`

**Prototype** `status_t DbgUnmangleSymbol (char *symbol,  
int32_t maxLen, char *outName)`

**Parameters** → *symbol*  
The name of the symbol.

→ *maxLen*  
The size of the buffer into which the unmangled name is to be written.

← *outName*  
The unmangled name of the symbol.

**Returns**

**Compatibility** This function is only implemented for the Palm OS Simulator.

**See Also** [DbgLookupSymbol\(\)](#)

## DbgVPrintf Function

**Purpose** Sends a string to the current debug device. The string is constructed using `vsprintf()`; accordingly, you pass this function a format string and a set of arguments.

**Declared In** `DebugMgr.h`

**Prototype** `long DbgVPrintf (const char *iFormat,  
va_list arglist)`

**Parameters** → *iFormat*  
Format string, as specified by `vsprintf()`.

## Debug Manager

### *DbgWalkStack*

---

→ *arglist*

Set of arguments (specified *varargs*-style) to be written, formatted according to *iFormat*, to the current debug device.

**Returns** Returns the number of bytes written to the current debug device.

**Comments** This function builds a string, up to 200 characters in length, using `vsprintf()`, and then sends it out via [DbgMessage\(\)](#). Formatted strings in excess of 200 characters will be truncated.

**See Also** [DbgMessage\(\)](#), [DbgPrintf\(\)](#)

## DbgWalkStack Function

**Purpose** Retrieve a stack crawl for the current thread.

**Declared In** `DebugMgr.h`

**Prototype** `int32_t DbgWalkStack (int32_t ignoreDepth,  
int32_t maxResults, void **outAddresses)`

**Parameters** → *ignoreDepth*

Allows you to skip functions at the top of the stack.

→ *maxResults*

The maximum number of available slots to return.

← *outAddresses*

Filled in with the address of each function on the stack, from the immediate caller down.

**Returns** Returns the number of functions actually found.

**Compatibility** This function is only implemented for the Palm OS Simulator.

**See Also** [DbgLookupSymbol\(\)](#), [DbgUnmangleSymbol\(\)](#)

# Desktop Link Server

---

|                                                          |     |
|----------------------------------------------------------|-----|
| <a href="#">Desktop Link Server Structures and Types</a> | 227 |
| <a href="#">Desktop Link Server Constants</a>            | 228 |
| <a href="#">Desktop Link Server Functions and Macros</a> | 231 |

The header file `DLServer.h` declares the API that this chapter describes.

## Desktop Link Server Structures and Types

### DlkCallAppReplyParamType Struct

|                    |                                                                                                                                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                                                                                                                                                                       |
| <b>Declared In</b> | <code>DLServer.h</code>                                                                                                                                                                                                                               |
| <b>Prototype</b>   | <pre>typedef struct DlkCallAppReplyParamType {     uint16_t pbSize;     uint16_t padding;     uint32_t dwResultCode;     const void *resultP;     uint32_t dwResultSize;     void *dlRefP;     uint32_t dwReserved1; } DlkCallAppReplyParamType</pre> |
| <b>Fields</b>      | <p><code>pbSize</code><br/>Size of this parameter block. Set it to <code>sizeof(DlkCallAppReplyParamType)</code>.</p> <p><code>padding</code><br/>Padding bytes.</p>                                                                                  |

## Desktop Link Server

*DlkCtlEnum*

---

`dwResultCode`  
Result code to be returned to the remote caller.

`resultP`  
Pointer to result data.

`dwResultSize`  
Size of result data block (number of bytes).

`dlRefP`  
Desktop Link reference pointer from  
`SysAppLaunchCmdHandleSyncCallAppType`.

`dwReserved1`  
Reserved. Set to `NULL`.

### DlkCtlEnum Typedef

**Purpose**

**Declared In** `DLServer.h`

**Prototype** `typedef Enum8 DlkCtlEnum`

## Desktop Link Server Constants

**Purpose**

**Declared In** `DLServer.h`

**Constants** `#define dlkErrIncompatibleProducts (dlkErrorClass  
| 8)`

`#define dlkErrInterrupted (dlkErrorClass | 6)`

`#define dlkErrLostConnection (dlkErrorClass | 5)`

`#define dlkErrMemory (dlkErrorClass | 2)`

```
#define dlkErrNoSession (dlkErrorClass | 3)

#define dlkErrNPOD (dlkErrorClass | 9)

#define dlkErrParam (dlkErrorClass | 1)

#define dlkErrSizeErr (dlkErrorClass | 4)

#define dlkErrUserCan (dlkErrorClass | 7)
```

## Miscellaneous Desktop Link Server Constants

|                    |                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                         |
| <b>Declared In</b> | DLServer.h                                                                                              |
|                    | <pre>#define dlkMaxUserNameLength (40)  #define dlkUserNameBufSize (dlkMaxUserNameLength +     1)</pre> |

## DlkCtlEnumTag Enum

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                |
| <b>Declared In</b> | DLServer.h                                                                                     |
| <b>Constants</b>   | <pre>dlkCtlFirst = 0  dlkCtlGetPCHostName  dlkCtlSetPCHostName  dlkCtlGetCondFilterTable</pre> |

## Desktop Link Server

*DlkSyncStateType*

---

`dlkCtlSetCondFilterTable`

`dlkCtlGetLANSync`

`dlkCtlSetLANSync`

`dlkCtlGetHSTCPPort`

`dlkCtlSetHSTCPPort`

`dlkCtlSendCallAppReply`

`dlkCtlGetPCHostAddr`

`dlkCtlSetPCHostAddr`

`dlkCtlGetPCHostMask`

`dlkCtlSetPCHostMask`

`dlkCtlGetPostSyncErr`

`dlkCtlSetPostSyncErr`

`dlkCtlLAST`

### **DlkSyncStateType Enum**

|                    |                                                                                 |
|--------------------|---------------------------------------------------------------------------------|
| <b>Purpose</b>     | State information returned by <a href="#">DlkGetSyncInfo</a> .                  |
| <b>Declared In</b> | <code>DLServer.h</code>                                                         |
| <b>Constants</b>   | <code>dlkSyncStateNeverSynced = 0</code><br>The handheld has never been synced. |
|                    | <code>dlkSyncStateInProgress</code><br>A sync is currently in progress.         |

- `dlkSyncStateLostConnection`  
The connection was lost during sync.
- `dlkSyncStateLocalCan`  
Sync was cancelled by the user on the handheld.
- `dlkSyncStateRemoteCan`  
Sync was cancelled by the user from the desktop.
- `dlkSyncStateLowMemoryOnTD`  
Sync ended due to a low memory condition on the handheld.
- `dlkSyncStateAborted`  
Sync was aborted for some other reason.
- `dlkSyncStateCompleted`  
Sync completed normally.
- `dlkSyncStateIncompatibleProducts`  
Sync ended because the desktop HotSync product is incompatible with this version of the handheld HotSync.
- `dlkSyncStateNPOD`  
The sync could not take place because the handheld has a 4.0-style password but the desktop hasn't yet been updated to a compatible version.

## Desktop Link Server Functions and Macros

### DlkControl Function

- Purpose** Perform an operation at the behest of the desktop software. Among other things, this function is used to return values to the conduit during the handling of [sysAppLaunchCmdHandleSyncCallApp](#).
- Declared In** `DLServer.h`
- Prototype** `status_t DlkControl (DlkCtlEnum op, void *param1P, void *param2P)`
- Parameters** → *op*  
Desktop Link control code. Use `dlkCtlSendCallAppReply` when sending a result back to the conduit while handling a `sysAppLaunchCmdHandleSyncCallApp` launch code.

## Desktop Link Server

*DlkControl*

---

↔ *param1P*

Pointer to the first parameter (operation-specific). For `dlkCtlSendCallAppReply`, this parameter should point to a [DlkCallAppReplyParamType](#) structure.

↔ *param2P*

Pointer to the second parameter (operation-specific). For `dlkCtlSendCallAppReply`, this parameter should be set to NULL.

**Returns** `errNone` if no error, or an error code if there was a problem during the call to `DlkControl`. In either case, place the value returned from `DlkControl` into the `replyErr` field of the `SysAppLaunchCmdHandleSyncCallAppType` structure when handling `sysAppLaunchCmdHandleSyncCallApp`.

**Comments** This function is needed to return data back to a conduit during the handling of [sysAppLaunchCmdHandleSyncCallApp](#). Set `param1P` to point to a [DlkCallAppReplyParamType](#) structure, as described below. See the [Example](#) on page 232 for an illustration of how to handle `sysAppLaunchCmdHandleSyncCallApp`.

**Example** The `SysAppLaunchCmdHandleSyncCallAppType` structure that accompanies the [sysAppLaunchCmdHandleSyncCallApp](#) launch code contains all of the information passed into `SyncCallRemoteModule` on the desktop as well as the necessary fields to pass the result pack to the desktop. At the end of your `sysAppLaunchCmdHandleSyncCallApp` launch code handler, you'll need to send a [DlkCallAppReplyParamType](#) reply structure back to the device using `DlkControl`.

---

```
#include <DLServer.h>
...
case sysAppLaunchCmdHandleSyncCallApp:
{
    SysAppLaunchCmdHandleSyncCallAppType *theCommandPtr;
    DlkCallAppReplyParamType theReplyParams;
    CharPtr theReplyBuffer = "SUCCESS";

    // Cast the cmdPBP to a SysAppLaunchCmdHandleSyncCallAppType
    // pointer so that we can work with it.
    theCommandPtr = (SysAppLaunchCmdHandleSyncCallAppType*)cmdPBP;

    // Do whatever work is necessary here. If you set the m_wActionCode
    // field in your CCallModuleParams class on the desktop, then you
    // can handle that code by looking at the action field of theCommandPtr
```

```
// (i.e.) if (theCommandPtr->action == 1)

// Create the reply to send back to the desktop

// First clear out all the fields. This is necessary so that the reserved
// fields are set to NULL.
MemSet( &theReplyParams, sizeof(DlkCallAppReplyParamType), 0 );

// Set the size of the reply. This is required.
theReplyParams.pbSize = sizeof(DlkCallAppReplyParamType);

// Set the result code. Normally this will be set to zero unless you want
// to send an error code back to the desktop.
theReplyParams.dwResultCode = 0;

// Fill in the reply buffer and buffer length
theReplyParams.resultP = theReplyBuffer;
theReplyParams.dwResultSize = StrLen(theReplyBuffer) + 1;

// Fill in the DL reference pointer. This is required.
theReplyParams.dlRefP = theCommandPtr->dlRefP;

// Set the handled field to true. This is required to let the desktop
// know that the sysAppLaunchCmdHandleSyncCallApp was handled. If you
// don't set this to true, the call to SyncCallRemoteModule will return
// SYNCERR_UNKNOWN_REQUEST.
theCommandPtr->handled = true;

// Finally, set the replyErr field by passing the reply parameters to
// DlkControl. This is required for the DLServer to properly handle the
// reply request.
theCommandPtr->replyErr = DlkControl (dlkCtlSendCallAppReply,
    &theReplyParams, NULL);

break;
}
```

---

[Table 22.1](#) and [Table 22.2](#) list some important mappings from the `CCallModuleParams` class on the desktop to the `SysAppLaunchCmdHandleSyncCallAppType` and `DlkCallAppReplyParamType` structures on the handheld.

**Table 22.1 CCallModuleParams to SysAppLaunchCmdHandleSyncCallAppType mapping**

| <b>CCallModuleParams</b> | <b>SysAppLaunchCmdHandleSyncCallAppType</b> |
|--------------------------|---------------------------------------------|
| m_wActionCode            | action                                      |
| m_dwParamSize            | dwParamSize                                 |
| m_pParam                 | paramP                                      |

**Table 22.2 CCallModuleParams to DlkCallAppReplyParamType mapping**

| <b>CCallModuleParams</b> | <b>DlkCallAppReplyParamType</b> |
|--------------------------|---------------------------------|
| m_dwResultBufSize        | dwResultSize                    |
| m_pResultBuf             | resultP                         |
| m_dwResultCode           | dwResultCode                    |

## DlkGetSyncInfo Function

- Purpose** Get the sync info managed by Desktop Link. This function is often used to obtain the user name on the handheld.
- Declared In** DLServer.h
- Prototype**  
`status_t DlkGetSyncInfo (uint32_t *succSyncDateP,  
uint32_t *lastSyncDateP,  
DlkSyncStateType *syncStateP, char *nameBufP,  
char *logBufP, int32_t *logLenP)`
- Parameters** ← *succSyncDateP*  
Pointer to the location where the date of the last successful sync is stored. Supply NULL for this parameter if this date isn't needed.

← *lastSyncDateP*

Pointer to the location where the date of the last sync, successful or otherwise, is stored. Supply NULL for this parameter if this date isn't needed.

← *syncStateP*

Pointer to a `DlkSyncStateType` enum into which the state of the last sync is stored. Supply NULL for this parameter if the state information isn't needed. See the Comments, below, for a description of this enum.

← *nameBufP*

Pointer to a string buffer into which the null-terminated handheld user name is stored. This string buffer must have been preallocated to be at least `dlkUserNameBufSize` bytes in length. Supply NULL for this parameter if the user name isn't needed.

← *logBufP*

Pointer to a string buffer into which the sync log text, null-terminated, is stored. Supply NULL for this parameter if the log text isn't needed. If you supply a valid pointer for this parameter, you must specify the preallocated buffer length using the `logLenP` parameter; the returned log text will be truncated, if necessary, to fit within the buffer.

↔ *logLenP*

Pointer to the log buffer size. If `logBufP` is not NULL, on entry you must set this value to the size of the `logBufP` buffer. When this function returns, this value indicates the actual length of the log text, not counting the null terminator.

**Returns** Returns `errNone` if no error, or `dlkErrMemory` if the Desktop Link preferences resource couldn't be locked.

**Comments** The state information returned through `syncStateP` has one of the values defined by the [DlkSyncStateType](#) enum.

**Example** This function is most often used to obtain the handheld user name. The following code excerpt shows how to do this (for clarity, error-checking has been omitted):

---

```
MemHandle nameH;  
char *nameP;  
  
// Allocate a buffer for the user name  
nameH = MemHandleNew(dlkUserNameBufSize);
```

## Desktop Link Server

### *DlkSetLogEntry*

---

```
nameP = MemHandleLock(nameH);

// Obtain the user's name
DlkGetSyncInfo(NULL, NULL, NULL, nameP, NULL, NULL);

// ... Do something with the user name here ...

// Now that we're done with the user name, free the buffer
MemPtrUnlock(nameP);
```

---

## DlkSetLogEntry Function

### **Purpose**

**Declared In** DLServer.h

**Prototype** void DlkSetLogEntry (const char \*textP,  
int16\_t textLen, Boolean append)

**Parameters** textP

→ *textLen*

→ *append*

### **Returns**

# Error Manager

---

The Error Manager consists of a set of functions and macros that allow you to conditionally display debugging messages when working with debug ROMs.

[ErrorManager Constants](#) . . . . . 237

[Error Manager Functions and Macros](#) . . . . . 238

The header file `ErrorMgr.h` declares the API that this chapter describes.

For tips on using the Error Manager APIs, see [Chapter 13](#), “[Debugging Strategies](#),” on page 113.

## ErrorManager Constants

### ErrDlgResultType Enum

|                    |                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Possible return types for <a href="#">ErrAlert()</a> .                                                                                                                                                                                                                                                                                                         |
| <b>Declared In</b> | <code>Form.h</code>                                                                                                                                                                                                                                                                                                                                            |
| <b>Constants</b>   | <p><code>errDlgResOK</code><br/>The user tapped the OK button.</p> <p><code>errDlgResCancel</code><br/>The user tapped the Cancel button.</p> <p><code>errDlgResRetry</code><br/>The user tapped the Retry button.</p> <p><code>errDlgResYes</code><br/>The user tapped the Yes button.</p> <p><code>errDlgResNo</code><br/>The user tapped the No button.</p> |
| <b>Comments</b>    | In Palm OS Cobalt, <code>ErrAlert()</code> only displays an OK button.                                                                                                                                                                                                                                                                                         |

# Error Manager Functions and Macros

## DbgOnlyFatalError Macro

|                    |                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Display an error alert dialog if you are not doing a release build. For release builds, this macro does nothing.                                                                                                                                                                                                                                |
| <b>Declared In</b> | <code>ErrorMgr.h</code>                                                                                                                                                                                                                                                                                                                         |
| <b>Prototype</b>   | <code>#define DbgOnlyFatalError (errMsg)</code>                                                                                                                                                                                                                                                                                                 |
| <b>Parameters</b>  | <code>→ errMsg</code><br>Error message text as a string.                                                                                                                                                                                                                                                                                        |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                                                                                                                        |
| <b>Comments</b>    | This macro displays a fatal error message, source code filename, and line number in a dialog. The dialog is cleared only when the user resets the system by responding to the dialog.<br><br>This macro is compiled into the code only if the <code>BUILD_TYPE</code> compiler define is something other than <code>BUILD_TYPE_RELEASE</code> . |
| <b>See Also</b>    | <a href="#">DbgOnlyFatalErrorIf()</a> , <a href="#">ErrFatalError()</a>                                                                                                                                                                                                                                                                         |

## DbgOnlyFatalErrorIf Macro

|                    |                                                                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Display an error alert dialog if you are not doing a release build and the specified condition is true.                                                                                                                                              |
| <b>Declared In</b> | <code>ErrorMgr.h</code>                                                                                                                                                                                                                              |
| <b>Prototype</b>   | <code>#define DbgOnlyFatalErrorIf (condition, errMsg)</code>                                                                                                                                                                                         |
| <b>Parameters</b>  | <code>→ condition</code><br>A boolean value. If true, display the error.<br><br><code>→ errMsg</code><br>Error message text as a string.                                                                                                             |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                             |
| <b>Comments</b>    | This macro displays a fatal error message, source code filename, and line number in a dialog. The alert is displayed only if <code>condition</code> is true. The dialog is cleared only when the user resets the system by responding to the dialog. |

This macro is compiled into the code only if the `BUILD_TYPE` compiler define is something other than `BUILD_TYPE_RELEASE`.

**See Also** [DbgOnlyFatalError\(\)](#), [ErrFatalErrorIf\(\)](#)

## ErrAlert Function

**Purpose** Displays an alert dialog for runtime errors.

**Declared In** `Form.h`

**Prototype** `uint16_t ErrAlert (DmOpenRef appDbRef,  
                  status_t errCode)`

**Parameters**  $\rightarrow$  `appDbRef`  
Open database containing the string list resource.

$\rightarrow$  `errCode`  
An error code. This is used as an index into a string list resource in a database. See Comments for more information.

**Returns** Zero, which indicates that the OK button has been clicked to dismiss the dialog.

**Comments** This function is intended for use by applications that are likely to receive runtime errors when the application itself is not at fault. For example, a networking application might use it to display an alert if the remote server cannot be found.

The error message displayed on the dialog is stored in a string list resource. A string list resource contains strings that can be looked up by index. The `errCode` parameter is used as the index into this list.

To use application-defined error codes in `ErrAlert()`, make sure that all of your error codes are greater than or equal to `appErrorClass`. This way, the error manager looks up the code in the application's string list resource number 0. All other error codes are taken from string list resources stored in the system.

**See Also** [ErrGetErrorMsg\(\)](#)

### ErrDisplay Macro

|                      |                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>       | Display an error alert.                                                                                                                                                                                              |
| <b>Declared In</b>   | <code>ErrorMgr.h</code>                                                                                                                                                                                              |
| <b>Prototype</b>     | <code>#define ErrDisplay (msg)</code>                                                                                                                                                                                |
| <b>Parameters</b>    | → <i>msg</i><br>Error message text as a string.                                                                                                                                                                      |
| <b>Returns</b>       | Nothing.                                                                                                                                                                                                             |
| <b>Comments</b>      | Call this macro to display an error message, source code filename, and line number.                                                                                                                                  |
| <b>Compatibility</b> | This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to use the (currently equivalent) <a href="#">ErrFatalError()</a> macro instead. |

### ErrDisplayFileLineMsg Macro

|                    |                                                                                                                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Display a dialog with an error message. Do not allow the user to exit the dialog or continue.                                                                                                                                                                                                                                                  |
| <b>Declared In</b> | <code>ErrorMgr.h</code>                                                                                                                                                                                                                                                                                                                        |
| <b>Prototype</b>   | <code>#define ErrDisplayFileLineMsg (a, b, c)</code>                                                                                                                                                                                                                                                                                           |
| <b>Parameters</b>  | → <i>a</i><br>Source code filename.<br>→ <i>b</i><br>Line number in the source code file.<br>→ <i>c</i><br>Message to display.                                                                                                                                                                                                                 |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                                                                                                                       |
| <b>Comments</b>    | Called by <a href="#">ErrFatalDisplayIf()</a> and <a href="#">ErrNonFatalDisplayIf()</a> . This function is useful when the application is already on the device and being tested by users.<br><br>On Japanese systems, the system displays a generic message indicating that an error has occurred instead of displaying the English message. |

**Compatibility** This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to directly call the [ErrFatalErrorInContext\(\)](#) function instead.

## **ErrFatalDisplay Macro**

**Purpose** Display an error alert dialog.

**Declared In** `ErrorMgr.h`

**Prototype** `#define ErrFatalDisplay (msg)`

**Parameters** `→ msg`  
Error message text as a string.

**Returns** Nothing.

**Comments** Call this macro to display a fatal error message, source code filename, and line number. The dialog is cleared only when the user resets the system by responding to the dialog.

**Compatibility** This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to use the (currently equivalent) [ErrFatalError\(\)](#) macro instead.

## **ErrFatalDisplayIf Macro**

**Purpose** Display an error alert dialog if the specified condition is true.

**Declared In** `ErrorMgr.h`

**Prototype** `#define ErrFatalDisplayIf (condition, msg)`

**Parameters** `→ condition`  
A boolean value. If true, display the error.

`→ msg`  
Error message text as a string.

**Returns** Nothing.

**Comments** Call this macro to display a fatal error message, source code filename, and line number. The alert is displayed only if `condition` is true. The dialog is cleared only when the user resets the system by responding to the dialog.

## Error Manager

### *ErrFatalError*

---

**Compatibility** This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to use the (currently equivalent) [ErrFatalErrorIf\(\)](#) macro instead.

### **ErrFatalError Macro**

**Purpose** Display an error alert dialog.

**Declared In** `ErrorMgr.h`

**Prototype** `#define ErrFatalError (errMsg)`

**Parameters** → *errMsg*  
Error message text as a string.

**Returns** Nothing.

**Comments** Call this macro to display a fatal error message, source code filename, and line number. The dialog is cleared only when the user resets the system by responding to the dialog.

**See Also** [DbgOnlyFatalError\(\)](#), [ErrFatalErrorIf\(\)](#)

### **ErrFatalErrorIf Macro**

**Purpose** Display an error alert dialog if the specified condition is true.

**Declared In** `ErrorMgr.h`

**Prototype** `#define ErrFatalErrorIf (condition, errMsg)`

**Parameters** → *condition*  
A boolean value. If true, display the error.

→ *errMsg*  
Error message text as a string.

**Returns** Nothing.

**Comments** Call this macro to display a fatal error message, source code filename, and line number. The alert is displayed only if *condition* is true. The dialog is cleared only when the user resets the system by responding to the dialog.

**See Also** [DbgOnlyFatalErrorIf\(\)](#), [ErrFatalError\(\)](#)

## **ErrFatalErrorInContext Function**

- Purpose** Display a dialog with an error message. Do not allow the user to exit the dialog or continue.
- Declared In** `ErrorMgr.h`
- Prototype** `void ErrFatalErrorInContext (const char *fileName, uint32_t lineNumber, const char *errMsg)`
- Parameters**
- *fileName*  
Source code filename.
  - *lineNum*  
Line number in the source code file.
  - *errMsg*  
Message to display.
- Returns** Nothing.
- Comments** Called by all of the macros documented in this chapter. This function is useful when the application is already on the device and being tested by users.

On Japanese systems, the system displays a generic message indicating that an error has occurred instead of displaying the English message.

## **ErrGetErrorMsg Function**

- Purpose** Looks up the error message for the specified error code.
- Declared In** `Form.h`
- Prototype** `status_t ErrGetErrorMsg (DmOpenRef appDbRef, status_t errCode, char *errMsgP, int32_t errMsgLen)`
- Parameters**
- *appDbRef*  
Open database containing the string list resource.
  - *errCode*  
An error code. This is used as an index into a string list resource in a database. See Comments for more information.
  - ← *errMsgP*  
The error message corresponding to *errCode*.

## Error Manager

### *ErrNonFatalDisplay*

---

→ *errMsgLen*

The size in bytes of the *errMsgP* parameter.

**Returns** `errNone` upon success or one of the following:

`sysErrParamErr`

Invalid *errMsgP* or *errMsgLen* parameter.

**Comments** This function is intended to be used under the same circumstances as [ErrAlert\(\)](#). You might use this function if you want to display the error message in a dialog of your own design.

The error message is stored in a string list resource. A string list resource contains strings that can be looked up by index. The *errCode* parameter is used as the index into this list.

To use application-defined error codes in `ErrGetErrorMsg()`, make sure that all of your error codes are greater than or equal to `appErrorClass`. Palm OS looks up application-specific codes in the application's string list resource number 0. All other error codes are taken from string list resources stored in the system.

**Example** The following code looks up an error code in the system library's database and displays it in an alert.

---

```
DmOpenRef syslibdbP = DmOpenDatabaseByTypeCreator
    (sysFileTLibrary, sysFileCSystem, dmModeReadOnly);
int32_t errMsgSize = 50;

myErrMsg = (char *)MemPtrNew(50);
ErrGetErrorMsg(syslibdbP, memErrNotEnoughSpace, myErrMsg,
    errMsgSize);
FrmCustomAlert(myAppdbP, MyAlertResourceID, myErrMsg,
    "", "");
```

---

## ErrNonFatalDisplay Macro

**Purpose** Display an error alert dialog if you are not doing a release build. For release builds, this macro does nothing.

**Declared In** `ErrorMgr.h`

**Prototype** `#define ErrNonFatalDisplay (msg)`

**Parameters** → *msg*

Error message text as a string.

|                      |                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Returns</b>       | Nothing.                                                                                                                                                                                                                 |
| <b>Comments</b>      | Call this macro to display a nonfatal error message, source code filename, and line number. The alert dialog is cleared when the user selects to continue (or resets the system).                                        |
| <b>Compatibility</b> | This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to use the (currently equivalent) <a href="#">DbgOnlyFatalError()</a> macro instead. |

## **ErrNonFatalDisplayIf Macro**

|                      |                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>       | Display an error alert dialog if the specified condition is true and you are not doing a release build. For release builds, this macro does nothing.                                                                                                     |
| <b>Declared In</b>   | <code>ErrorMgr.h</code>                                                                                                                                                                                                                                  |
| <b>Prototype</b>     | <code>#define ErrNonFatalDisplayIf (<i>condition</i>, <i>msg</i>)</code>                                                                                                                                                                                 |
| <b>Parameters</b>    | <code>→ <i>condition</i></code><br>A boolean value. If <code>true</code> , display the error.<br><code>→ <i>msg</i></code><br>Error message text as a string.                                                                                            |
| <b>Returns</b>       | Nothing.                                                                                                                                                                                                                                                 |
| <b>Comments</b>      | Call this macro to display a nonfatal error message, source code filename, and line number. The alert is displayed only if <i>condition</i> is <code>true</code> . The alert dialog is cleared when the user selects to continue (or resets the system). |
| <b>Compatibility</b> | This macro is provided for compatibility with applications ported from earlier releases of Palm OS; new applications are encouraged to use the (currently equivalent) <a href="#">DbgOnlyFatalErrorIf()</a> macro instead.                               |

## **Error Manager**

*ErrNonFatalDisplayIf*

---

# ErrTryCatch

---

This chapter describes the Palm OS “Try/Catch” exception-handling mechanism. It is organized into the following sections:

|                                                            |     |
|------------------------------------------------------------|-----|
| <a href="#">ErrTryCatch Structures and Types</a> . . . . . | 247 |
| <a href="#">ErrTryCatch Functions and Macros</a> . . . . . | 248 |

The header file `ErrTryCatch.h` declares the API that this chapter describes.

For instructions on using these APIs to throw and handle exceptions, see “[The Try-and-Catch Mechanism](#)” on page 116.

## ErrTryCatch Structures and Types

### ErrExceptionType Struct

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | An <code>ErrExceptionType</code> structure is created for each <code>ErrTry</code> and <code>ErrCatch()</code> block. At any point in the program, there is a linked list of these structures. An <code>ErrExceptionType</code> structure stores information about the state of the machine (register values, an error code, and the address at which the error occurred) at the start of the <code>ErrTry</code> block. |
| <b>Declared In</b> | <code>ErrTryCatch.h</code>                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Prototype</b>   | <pre>typedef struct ErrExceptionType {     struct ErrExceptionType *nextP;     ErrJumpBuf state;     int32_t err;     VAddr errVAddr; } ErrExceptionType typedef ErrExceptionType *ErrExceptionPtr</pre>                                                                                                                                                                                                                 |
| <b>Fields</b>      | <p><code>nextP</code><br/>Next <code>ErrExceptionType</code> structure in the linked list.</p>                                                                                                                                                                                                                                                                                                                           |

## ErrTryCatch

### ErrJumpBuf

---

**state** Environment storage for [ErrSetJump\(\)](#)/[ErrLongJump\(\)](#).  
**err** Error code.  
**errVAddr** Address reference that caused the fault.

### ErrJumpBuf Typedef

**Purpose** Contains state information for [ErrSetJump\(\)](#)/[ErrLongJump\(\)](#). Used in the [ErrExceptionType](#) structure.  
**Declared In** `ErrTryCatch.h`  
**Prototype** `typedef long *ErrJumpBuf[16]`

## ErrTryCatch Functions and Macros

### ErrCatch Macro

**Purpose** Marks the end of an [ErrTry\(\)](#) block and the beginning of an `ErrCatch` block.  
**Declared In** `ErrTryCatch.h`  
**Prototype** `#define ErrCatch (inErr)`  
**Parameters**  $\rightarrow$  *inErr*  
An exception code identifying the reason for the failure. This is the value supplied to the [ErrThrow\(\)](#) call that caused the jump to this `ErrCatch` block.  
**Returns** Returns nothing.  
**Comments** `ErrCatch` can only be used in conjunction with [ErrTry\(\)](#) and `ErrEndCatch`. See the comments under `ErrTry` for usage instructions.  
`ErrTry`, `ErrCatch` and `ErrThrow` are based on `set jmp` ([ErrSetJump\(\)](#)) and `long jmp` ([ErrLongJump\(\)](#)). At the beginning of an `ErrTry` block, `set jmp` saves the machine registers. `ErrThrow` calls `long jmp`, which restores the registers and jumps to

the beginning of the `ErrCatch` block. Therefore, any changes in the `ErrTry` block to variables stored in registers aren't retained when entering the `ErrCatch` block.

The solution is to declare variables that you want to use in both the `ErrTry` and `ErrCatch` blocks as "volatile". For example:

---

```
volatile long x = 1; // Declare volatile local variable
ErrTry {
    x = 100; // Set local variable in Try
    ErrThrow(-1);
}
ErrCatch(inErr) {
    if (x > 1) { // Use local variable in Catch
        SysBeep(1);
    }
} ErrEndCatch
```

---

If you have many local variables after the `ErrCatch` you may want to put the `ErrTry` and `ErrCatch` in a separate enclosing function.

## ErrCatchWithAddress Macro

|                    |                                                                                                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Marks the end of an <a href="#">ErrTry()</a> block and the beginning of an <code>ErrCatch</code> block. Unlike <a href="#">ErrCatch()</a> , this macro works with <a href="#">ErrThrowWithAddress()</a> to give the address reference that caused fault, if such an address.                                                    |
| <b>Declared In</b> | <code>ErrTryCatch.h</code>                                                                                                                                                                                                                                                                                                      |
| <b>Prototype</b>   | <code>#define ErrCatchWithAddress (inErr, inErrVAddr)</code>                                                                                                                                                                                                                                                                    |
| <b>Parameters</b>  | <p>→ <i>inErr</i><br/>An exception code identifying the reason for the failure. This is the value supplied to the <a href="#">ErrThrowWithAddress()</a> call that caused the jump to this <code>ErrCatch</code> block.</p> <p>→ <i>inErrVAddr</i><br/>Address reference that caused fault, if such an address is available.</p> |
| <b>Returns</b>     | Returns nothing.                                                                                                                                                                                                                                                                                                                |
| <b>Comments</b>    | If the exception was thrown with <a href="#">ErrThrow()</a> or <a href="#">ErrThrowIf()</a> , <i>inErrVAddr</i> is set to <code>0xffffffff</code> .                                                                                                                                                                             |

## ErrTryCatch

### ErrEndCatch

---

ErrCatch can only be used in conjunction with [ErrTry\(\)](#) and [ErrEndCatch](#). See the comments under [ErrTry](#) for usage instructions. Also see the comments under [ErrCatch](#) for information on changes to variables stored in registers.

## ErrEndCatch Macro

|                    |                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Marks the end of an <a href="#">ErrCatch()</a> block.                                                                                                                                                                                       |
| <b>Declared In</b> | <a href="#">ErrTryCatch.h</a>                                                                                                                                                                                                               |
| <b>Prototype</b>   | <code>#define ErrEndCatch</code>                                                                                                                                                                                                            |
| <b>Parameters</b>  | None.                                                                                                                                                                                                                                       |
| <b>Returns</b>     | Returns nothing.                                                                                                                                                                                                                            |
| <b>Comments</b>    | <a href="#">EndErrCatch</a> can only be used in conjunction with <a href="#">ErrTry()</a> and <a href="#">ErrCatch()</a> (or <a href="#">ErrCatchWithAddress()</a> ). See the comments under <a href="#">ErrTry</a> for usage instructions. |

## ErrExceptionListAppend Function

|                    |                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Appends an exception to the exception list.                                                   |
| <b>Declared In</b> | <a href="#">ErrTryCatch.h</a>                                                                 |
| <b>Prototype</b>   | <code>void ErrExceptionListAppend<br/>(ErrExceptionType *errExceptionTypeP)</code>            |
| <b>Parameters</b>  | <code>→ errExceptionTypeP</code><br>Pointer to the exception to be appended.                  |
| <b>Returns</b>     | Returns nothing.                                                                              |
| <b>See Also</b>    | <a href="#">ErrExceptionListGetByThreadID()</a> ,<br><a href="#">ErrExceptionListRemove()</a> |

## ErrExceptionListGetByThreadID Function

- Purpose** Gets the head of the exception list for a given thread, without changing the list.
- Declared In** ErrTryCatch.h
- Prototype** `ErrExceptionType *ErrExceptionListGetByThreadID  
(SysHandle iThreadID)`
- Parameters** → *iThreadID*  
ID of the thread for which the exception list is being obtained.
- Returns** Returns the exception at the head of the exception list.
- Comments** This function is designed to be used by a keeper thread.
- See Also** [ErrExceptionListAppend\(\)](#), [ErrExceptionListRemove\(\)](#)

## ErrExceptionListRemove Function

- Purpose** Removes a specified exception from the exception list.
- Declared In** ErrTryCatch.h
- Prototype** `void ErrExceptionListRemove  
(ErrExceptionType *errExceptionTypeP)`
- Parameters** → *errExceptionTypeP*  
Pointer to the exception to be removed.
- Returns** Returns nothing.
- See Also** [ErrExceptionListAppend\(\)](#),  
[ErrExceptionListGetByThreadID\(\)](#)

## ErrLongJump Function

- Purpose** Restores the environment saved in the specified buffer by a call to [ErrSetJump\(\)](#).
- Declared In** `ErrTryCatch.h`
- Prototype** `void ErrLongJump (ErrJumpBuf buf, int32_t result)`
- Parameters**
- `→ buf`  
Pointer to a buffer containing the environment to be restored ( program counter and stack pointer).
  - `→ result`  
Value that will appear to have been returned from [ErrSetJump\(\)](#) once the environment has been restored.
- Returns** Returns nothing.

## ErrSetJump Function

- Purpose** Saves a copy of the current environment (program counter and the current pointer to the top of the stack) for later restoration by [ErrLongJump\(\)](#).
- Declared In** `ErrTryCatch.h`
- Prototype** `int32_t ErrSetJump (ErrJumpBuf buf)`
- Parameters**
- `→ buf`  
Pointer to a buffer in which the program counter and stack pointer are to be stored.
- Returns** Returns `errNone`. Note, however, that when a call to [ErrLongJump\(\)](#) is made, the saved environment is restored, causing program execution to resume as if `ErrSetJump` had just returned; the value “returned “at that time is the one specified by the call to `ErrLongJump`.

## ErrThrow Macro

|                    |                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Cause a jump to the nearest catch block.                                                                                                    |
| <b>Declared In</b> | <code>ErrTryCatch.h</code>                                                                                                                  |
| <b>Prototype</b>   | <code>#define ErrThrow (err)</code>                                                                                                         |
| <b>Parameters</b>  | <code>→ err</code><br>Error code.                                                                                                           |
| <b>Returns</b>     | Never returns.                                                                                                                              |
| <b>Comments</b>    | Use the macros <a href="#">ErrTry()</a> , <a href="#">ErrCatch()</a> , and <a href="#">ErrEndCatch()</a> in conjunction with this function. |
| <b>See Also</b>    | <a href="#">ErrFatalDisplayIf()</a> , <a href="#">ErrNonFatalDisplayIf()</a> , <a href="#">ErrDisplay()</a>                                 |

## ErrThrowIf Macro

|                    |                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | If the error code parameter is other than <code>errNone</code> , cause a jump to the nearest catch block.                                   |
| <b>Declared In</b> | <code>ErrTryCatch.h</code>                                                                                                                  |
| <b>Prototype</b>   | <code>#define ErrThrowIf (err)</code>                                                                                                       |
| <b>Parameters</b>  | <code>→ err</code><br>Error code.                                                                                                           |
| <b>Returns</b>     | Never returns.                                                                                                                              |
| <b>Comments</b>    | Use the macros <a href="#">ErrTry()</a> , <a href="#">ErrCatch()</a> , and <a href="#">ErrEndCatch()</a> in conjunction with this function. |
| <b>See Also</b>    | <a href="#">ErrFatalDisplayIf()</a> , <a href="#">ErrNonFatalDisplayIf()</a> , <a href="#">ErrDisplay()</a>                                 |

## ErrThrowWithAddress Function

|                |                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b> | Throw an error to the first handler on the exception list for this thread, causing a jump to the nearest catch block. Unlike |
|----------------|------------------------------------------------------------------------------------------------------------------------------|

## ErrTryCatch

*ErrThrowWithHandler*

---

[ErrThrow\(\)](#) and [ErrThrowIf\(\)](#), you can use this function to pass the address reference that caused the fault.

**Declared In** ErrTryCatch.h

**Prototype** void ErrThrowWithAddress (int32\_t err,  
VAddr errVAddr)

**Parameters** → *err*  
Error code.  
→ *errVAddr*  
Address to be reported to [ErrCatchWithAddress\(\)](#).

**Returns** Never returns.

**Comments** Use the macros [ErrTry\(\)](#), [ErrCatchWithAddress\(\)](#), and [ErrEndCatch\(\)](#) in conjunction with this function.

**See Also** [ErrFatalDisplayIf\(\)](#), [ErrNonFatalDisplayIf\(\)](#), [ErrDisplay\(\)](#)

## ErrThrowWithHandler Function

**Purpose** This function is what the keeper thread should make the faulted thread run when it resumes. This function will do the [ErrLongJump\(\)](#) to make the faulted thread clear its stack and jump to its [ErrCatch\(\)](#) handler.

**Declared In** ErrTryCatch.h

**Prototype** void ErrThrowWithHandler (int32\_t err,  
VAddr errVAddr, ErrExceptionType \*tryP)

**Parameters** → *err*  
Error code.  
→ *errVAddr*  
Address to be reported to [ErrCatchWithAddress\(\)](#).  
→ *tryP*  
Pointer to the exception state to be handled by [ErrCatch\(\)](#) or [ErrCatchWithAddress\(\)](#).

**Returns** Never returns.

**Comments** This function is designed to be used by a keeper thread.

**See Also** [ErrFatalDisplayIf\(\)](#), [ErrNonFatalDisplayIf\(\)](#),  
[ErrDisplay\(\)](#)

## ErrTry Macro

**Purpose** Marks the beginning of a try/catch block.

**Declared In** `ErrTryCatch.h`

**Prototype** `#define ErrTry`

**Parameters** None.

**Returns** Returns nothing.

**Comments** An exception raised by a call to [ErrThrow\(\)](#)—even from within a nested subroutine—causes program execution to switch to the beginning of the [ErrCatch\(\)](#) block. If the end of the block enclosed by `ErrTry` is encountered without a call to `ErrThrow`, execution jumps to the line of code following the [ErrEndCatch\(\)](#) macro. See “[The Try-and-Catch Mechanism](#)” on page 116 for a more thorough description of how this mechanism works.

**Example** You must structure your code exactly as shown here. You can't use `ErrTry` without `ErrCatch` and `ErrEndCatch`, or vice versa.

---

```
ErrTry {
    // Do something which may fail. Call ErrThrow to signal
    // failure and force a jump to the following ErrCatch
    // block.
}
ErrCatch(inErr) {
    // Recover or cleanup after a failure in the above ErrTry
    // block. "inErr" is an exception code identifying the
    // reason for the failure.

    // Call ErrThrow if you want to jump out to the next
    // ErrCatch block.

    // The code in this block doesn't execute if the above
    // ErrTry block completes without a call to ErrThrow.
} ErrEndCatch
```

---

## **ErrTryCatch**

*ErrTry*

---

# Expansion Manager

---

This chapter contains reference material for the Expansion Manager, which applications use to work with the handheld's expansion slots. Note that to perform filesystem operations on the media in the slot you use the VFS Manager, which is documented in *Exploring Palm OS: Memory, Databases, and Files*.

This chapter is organized into the following sections:

[Expansion Manager Structures and Types](#). . . . . 257

[Expansion Manager Constants](#) . . . . . 260

[Expansion Manager Functions and Macros](#) . . . . . 265

The header file `ExpansionMgr.h` declares the API that this chapter describes.

For basic Expansion Manager concepts and instructions on using the Expansion Manager APIs, see [Chapter 5, "Expansion,"](#) on page 53.

## Expansion Manager Structures and Types

### CardMetricsType Struct

**Purpose** Contains information about the physical structure of the card: the type of information that may be needed by a file system in order to

## Expansion Manager

### *CardMetricsType*

---

format volumes on the card, for instance. This structure is passed as a parameter in [ExpCardMetrics\(\)](#).

**Declared In** ExpansionMgr.h

**Prototype**

```
typedef struct CardMetricsTag {
    uint32_t totalSectors;
    uint16_t bytesPerSector;
    uint16_t sectorsPerHead;
    uint16_t headsPerCylinder;
    uint16_t reserved1;
    uint8_t sectorsPerBlock;
    uint8_t partitionType;
    uint8_t bootIndicator;
    uint8_t reserved2;
    uint32_t partitionStart;
    uint32_t partitionSize;
} CardMetricsType, *CardMetricsPtr
```

**Fields** totalSectors

The total number of sectors accessible. Some media may contain extra sectors in case one goes bad, or for storing configuration information, but they are handled internally to the block device driver, and not accessible to applications.

bytesPerSector

The number of bytes in one sector. Currently for Palm OS, this must be 512.

sectorsPerHead

The number of Sectors per Head as given by guidelines in the specification for this media type. Even though all Palm OS disk accesses are LBA, this is for compatibility when filling out MBRs and PBRs. If the media guidelines don't specify, this value is set to 0.

headsPerCylinder

The number of Heads per Cylinder as given by guidelines in the specification for this media type. Even though all Palm OS disk accesses are LBA, this is for compatibility when filling out MBRs and PBRs. If the media guidelines don't specify, this value is set to 0.

reserved1

Reserved

**sectorsPerBlock**

A suggested number of Sectors per Block (Cluster) as given by guidelines in the specification for this media type. If the media guidelines don't specify, this value is set to 0.

**partitionType**

The suggested partition type (System ID) of the first partition as given by guidelines in the specification for this media type. If the media guidelines don't specify, this value is set to `slotDrvNonBootablePartition`. See "[Partition Type Flags](#)" on page 263 for a list of defined partition types.

**bootIndicator**

The suggested bootability of the first partition as given by guidelines in the specification for this media type. Generally, `0x80` (`slotDrvBootablePartition`) is bootable, while the default boot partition `0x00` (`slotDrvNonBootablePartition`) is not-bootable. If the media guidelines don't specify, this value is set to `0xFF`.

**reserved2**

Reserved

**partitionStart**

The suggested starting sector of the first partition as given by guidelines in the specification for this media type. If this value is set to zero, and the partition size value is non-zero, the media guidelines suggest to not use an MBR, and only use a PBR at sector 0. If the media guidelines don't specify, the partition size value is set to 0.

**partitionSize**

The suggested size of the first partition as given by guidelines in the specification for this media type. If the media guidelines don't specify, this value is set to 0, and the partition start parameter is ignored.

## **ExpCardInfoType Struct**

**Purpose** The `ExpCardInfoType` declaration defines a structure that is passed to [ExpCardInfo\(\)](#). This structure is used to determine the characteristics of the card loaded in the slot. It is initialized by the underlying block device driver with the following information.

## Expansion Manager

### Expansion Manager Constants

---

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Declared In</b> | ExpansionMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Prototype</b>   | <pre>typedef struct ExpCardInfoTag {     uint32_t capabilityFlags;     char manufacturerStr[expCardInfoStringMaxLen+1];     char productStr[expCardInfoStringMaxLen+1];     char deviceClassStr[expCardInfoStringMaxLen+1];     char deviceUniqueIDStr[expCardInfoStringMaxLen+1]; } ExpCardInfoType, *ExpCardInfoPtr</pre>                                                                                                                                                                                                                                                                                                                   |
| <b>Fields</b>      | <p><b>capabilityFlags</b><br/>Describes the capabilities of the card. See <a href="#">“Capability Flags”</a> on page 263 for the set of flags that are currently supported.</p> <p><b>manufacturerStr</b><br/>Names the manufacturer of the card. For example “Palm” or “Motorola”.</p> <p><b>productStr</b><br/>Name of the product. For example “SafeBackup 32MB”.</p> <p><b>deviceClassStr</b><br/>Describes the type of card, for example, “Backup” or “Ethernet”.</p> <p><b>deviceUniqueIDStr</b><br/>Unique identifier for the product. A serial number for example. This value is set to the empty string if no identifier exists.</p> |

## Expansion Manager Constants

### Expansion Manager Error Codes

|                    |                                                                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error codes returned by the various Expansion Manager functions.                                                                                        |
| <b>Declared In</b> | ExpansionMgr.h                                                                                                                                          |
| <b>Constants</b>   | <pre>#define expErrCardBadSector (expErrorClass   8)</pre> <p>The card supports the block device driver block read/write API but the sector is bad.</p> |

```
#define expErrCardNoSectorReadWrite (expErrorClass
| 6)
    The card does not support the block device driver block
    read/write API.

#define expErrCardNotPresent (expErrorClass | 3)
    There is no card present in the given slot.

#define expErrCardProtectedSector (expErrorClass |
9)
    The card supports the block device driver block read/write
    API but the sector is protected.

#define expErrCardReadOnly (expErrorClass | 7)
    The card supports the block device driver block read/write
    API but the card is read only.

#define expErrEnumerationEmpty (expErrorClass |
13)
    There are no values remaining to enumerate.

#define expErrIncompatibleAPIVer (expErrorClass |
14)
    The API version of the underlying block device driver is not
    supported by this version of Expansion Manager.

#define expErrInvalidSlotRefNum (expErrorClass |
4)
    The slot reference number is not valid.

#define expErrNotEnoughPower (expErrorClass | 2)
    The required power is not available.

#define expErrNotOpen (expErrorClass | 10)
    The block device driver library has not been opened.

#define expErrSlotDeallocated (expErrorClass | 5)
    The slot reference number is within the valid range, but the
    slot has been deallocated.

#define expErrStillOpen (expErrorClass | 11)
    The block device driver library is still open; it may have been
    opened more than once.

#define expErrUnimplemented (expErrorClass | 12)
    The call is unimplemented.
```

## Expansion Manager

### Defined Media Types

---

```
#define expErrUnsupportedOperation (expErrorClass  
    | 1)  
    The operation is unsupported or undefined.
```

## Defined Media Types

**Purpose** The following media types are defined by the Expansion Manager. These media types are used with the function [VFSVolumeInfo\(\)](#) in the `VolumeInfoType.mediaType` field. The media type is also passed as a parameter to the [VFSRegisterDefaultDirectory\(\)](#) and [VFSUnregisterDefaultDirectory\(\)](#) functions.

**Declared In** `ExpansionMgr.h`

**Constants**

```
#define expMediaType_Any 'wild'  
    Matches all media types when looking up a default directory  
  
#define expMediaType_CompactFlash 'cfsh'  
    Compact Flash  
  
#define expMediaType_MacSim 'PSim'  
    Host file system emulated by the Mac Simulator  
  
#define expMediaType_MemoryStick 'mstk'  
    Memory stick  
  
#define expMediaType_MultiMediaCard 'mmcd'  
    MultiMedia Card  
  
#define expMediaType_PoserHost 'pose'  
    Host file system emulated by the Palm OS® Emulator  
  
#define expMediaType_RAMDisk 'ramd'  
    A RAM disk based media  
  
#define expMediaType_SecureDigital 'sdig'  
    Secure Digital  
  
#define expMediaType_SmartMedia 'smed'  
    SmartMedia
```

## Capability Flags

|                    |                                                                                                                                                                                                                                                                               |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Describes the capabilities of the expansion card.                                                                                                                                                                                                                             |
| <b>Declared In</b> | ExpansionMgr.h                                                                                                                                                                                                                                                                |
| <b>Constants</b>   | <pre>#define expCapabilityHasStorage (0x00000001)     The card supports reading and (possibly) writing. #define expCapabilityReadOnly (0x00000002)     The card is read only. #define expCapabilitySerial (0x00000004)     The card supports a simple serial interface.</pre> |

## Enumeration Constants

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Control the process of slot enumeration when using <a href="#">ExpSlotEnumerate()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Declared In</b> | ExpansionMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Constants</b>   | <pre>#define expIteratorStart (0L)     Supply this value to <a href="#">ExpSlotEnumerate()</a> to begin     enumeration. #define expIteratorStop (0xffffffffL)     When enumeration reaches the last slot,     <a href="#">ExpSlotEnumerate()</a> sets its slot iterator parameter to this     value. #define expInvalidSlotRefNum (0)     When enumerating slots, if the handheld has no expansion     slots <a href="#">ExpSlotEnumerate()</a> sets its slot reference number     parameter to this value.</pre> |

## Partition Type Flags

|                    |                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The suggested partition type (System ID) of the first partition as given by guidelines in the specification for the media type. |
| <b>Declared In</b> | ExpansionMgr.h                                                                                                                  |
| <b>Constants</b>   | <pre>#define slotDrvRBootablePartition (0x80)     The partition is bootable.</pre>                                              |

## Expansion Manager

### Miscellaneous Expansion Manager Constants

---

```
#define slotDrvNonBootablePartition (0x00)
    The partition is non-bootable.

#define slotDrvPartitionTypeFAT12 (0x01)
    The partition is formatted as FAT12.

#define slotDrvPartitionTypeFAT16Over32MB (0x06)
    The partition is formatted as FAT16 and exceeds 32 MB.

#define slotDrvPartitionTypeFAT16Under32MB (0x04)
    The partition is formatted as FAT16 but is less than 32 MB in
    size.

#define slotDrvPartitionTypeFAT32 (0x0b)
    The partition is formatted as FAT32.
```

**Comments** The partition type is returned in the `partitionType` field of the [CardMetricsType](#) structure by a call to [ExpCardMetrics\(\)](#). The returned value is a combination of one of the `slotDrvPartitionType...` flags and either `slotDrvBootablePartition` or `slotDrvNonBootablePartition`. A value of zero (equivalent to `slotDrvNonBootablePartition`) is returned if the media guidelines don't specify a partition type.

## Miscellaneous Expansion Manager Constants

**Purpose** The Expansion Manager also defines these constants.

**Declared In** `ExpansionMgr.h`

**Constants**

```
#define expCardInfoStringMaxLen (31)
    Maximum length (not including the terminating null
    character) of the strings in the ExpCardInfoType structure.

#define expFtrIDVersion (0)
    To obtain the version of the Expansion Manager, call
    FtrGet\(\), supplying a feature creator of
    sysFileCExpansionMgr and this constant for the feature
    number.

#define expHandledSound (0x02)

#define expHandledVolume (0x01)
```

```
#define expMgrVersionNum ((uint16_t)300)
```

The current version of the Expansion Manager. To obtain the version of the Expansion Manager running on the handheld, call [FtrGet\(\)](#), supplying a feature creator of `sysFileCExpansionMgr` and a feature number of `expFtrIDVersion`.

## Expansion Manager Functions and Macros

### ExpCardInfo Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Obtains information about a card in a given slot.                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Declared In</b> | <code>ExpansionMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Prototype</b>   | <code>status_t ExpCardInfo (uint16_t slotRefNum, ExpCardInfoType *infoP)</code>                                                                                                                                                                                                                                                                                                                                                |
| <b>Parameters</b>  | <p>→ <code>slotRefNum</code><br/>Slot number.</p> <p>← <code>infoP</code><br/>Pointer to <a href="#">ExpCardInfoType</a> structure.</p>                                                                                                                                                                                                                                                                                        |
| <b>Returns</b>     | <p>Returns <code>errNone</code> if the operation completed successfully, or one of the following otherwise:</p> <p><code>expErrCardNotPresent</code><br/>There is no card present in the specified slot.</p> <p><code>expErrInvalidSlotRefNum</code><br/>The specified slot number is invalid.</p> <p><code>expErrSlotDeallocated</code><br/>The specified slot number is within the valid range but has been deallocated.</p> |
| <b>Comments</b>    | This function returns information about a card, including whether the card supports secondary storage or is strictly read-only, by filling in the <code>ExpCardInfoType</code> structure's fields.                                                                                                                                                                                                                             |
| <b>See Also</b>    | <a href="#">ExpCardPresent()</a> , <a href="#">ExpSlotEnumerate()</a>                                                                                                                                                                                                                                                                                                                                                          |

## Expansion Manager

*ExpCardIsFilesystemSupported*

---

### ExpCardIsFilesystemSupported Function

- Purpose** Determine whether a particular file system type is natively supported by the media in the defined slot.
- Declared In** `ExpansionMgr.h`
- Prototype** `Boolean ExpCardIsFilesystemSupported (uint16_t slotRefNum, uint32_t filesystemType)`
- Parameters**
- `slotRefNum`  
Slot reference number.
  - `filesystemType`  
One of the file system types that this library implements. See [“Defined File Systems”](#) in the VFS Manager chapter of *Exploring Palm OS: Memory, Databases, and Files* for a list of file system types.
- Returns** Returns `true` if the specified file system is supported, or `false` if either the file system is not supported or if an error occurred.
- Comments** *Native support* means that formatting the media for another type of file system will probably break the media when used with other devices (such as digital cameras, voice recorders etc.) for which it was designed.
- The VFS Manager uses this function to determine the best file system to be used when formatting media in a slot, and to warn the user before formatting media with an incompatible type.

### ExpCardMediaType Function

- Purpose** Determine the media type of the card in the given slot.
- Declared In** `ExpansionMgr.h`
- Prototype** `status_t ExpCardMediaType (uint16_t slotRefNum, uint32_t *mediaTypeP)`
- Parameters**
- `slotRefNum`  
Slot reference number.
  - ← `mediaTypeP`  
Set to the media type of the card in the given slot. See [“Defined Media Types”](#) on page 262 for a list of possible media types.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`expErrNotOpen`

Block device driver library has not been opened

`expErrInvalidSlotRefNum`

Slot reference number is not valid.

`expErrCardNotPresent`

No card is present in the specified slot.

**Comments** This function sets the `mediaTypeP` to reflect the kind of media that is in the slot. Supported media types are defined in `ExpansionMgr.h`. The media type is used by [VFSVolumeInfo\(\)](#), [VFSRegisterDefaultDirectory\(\)](#), and [VFSUnregisterDefaultDirectory\(\)](#) (see *Exploring Palm OS: Memory, Databases, and Files* for information on the VFS Manager).

**See Also** [ExpSlotMediaType\(\)](#)

## ExpCardMetrics Function

**Purpose** Get all of the information about the physical structure of the card. The returned information is of the type that may be needed by a file system in order to format volumes on the card.

**Declared In** `ExpansionMgr.h`

**Prototype** `status_t ExpCardMetrics (uint16_t slotRefNum, CardMetricsPtr cardMetricsP)`

**Parameters** → `slotRefNum`

Slot reference number.

← `cardMetricsP`

Filled in on return with information on the physical layout of the card. Note that if the `reservedRangesP` field of this structure is filled in, it is the responsibility of the caller to dispose of this memory allocated by this function. Refer to “[CardMetricsType](#)” on page 257 for a description of the fields in this structure.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

## Expansion Manager

*ExpCardPresent*

---

`expErrNotOpen`

Block device driver library has not been opened

`expErrInvalidSlotRefNum`

Slot reference number is not valid.

`expErrCardNotPresent`

No card is present in the specified slot.

**See Also** [ExpCardSectorWrite\(\)](#)

### ExpCardPresent Function

**Purpose** Determines if a card is present in the given slot.

**Declared In** `ExpansionMgr.h`

**Prototype** `status_t ExpCardPresent (uint16_t slotRefNum)`

**Parameters** `→ slotRefNum`

Slot number.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`expErrCardNotPresent`

There is no card present in the specified slot.

`expErrInvalidSlotRefNum`

The specified slot number is invalid.

`expErrSlotDeallocated`

The specified slot number is within the valid range but has been deallocated.

**Comments** The Expansion Manager passes the call through to the appropriate block device driver.

**See Also** [ExpCardInfo\(\)](#), [ExpSlotEnumerate\(\)](#)

## ExpCardSectorRead Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Read contiguous 512-byte sectors from the card in a specified slot.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Declared In</b> | <code>ExpansionMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Prototype</b>   | <pre>status_t ExpCardSectorRead (uint16_t slotRefNum,<br/>                             uint32_t sectorNumber, uint8_t *bufferP,<br/>                             uint32_t *numSectorsP)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Parameters</b>  | <p>→ <i>slotRefNum</i><br/>Slot reference number.</p> <p>→ <i>sectorNumber</i><br/>The sector from which to start reading.</p> <p>← <i>bufferP</i><br/>Buffer that will receive the sectors read from the card. This buffer must be a multiple of 512 bytes long.</p> <p>↔ <i>numSectorsP</i><br/>When calling this function, set this to the number of sectors to read. Upon return, it is set to the number of sectors actually read.</p>                                                                                                                                                                                         |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation completed successfully, or one of the following otherwise:<br><br><code>expErrNotOpen</code><br>Block device driver library has not been opened<br><br><code>expErrInvalidSlotRefNum</code><br>Slot reference number is not valid.<br><br><code>expErrCardNoSectorReadWrite</code><br>The card does not support the block read/write API.<br><br><code>expErrCardBadSector</code><br>One of the requested sectors is bad.<br><br><code>expErrCardProtectedSector</code><br>One of the requested sectors is protected.<br><br><code>expErrCardNotPresent</code><br>No card is present. |
| <b>Comments</b>    | This function is used by the file system library to read data from the media. Information such as the MBR, PBR, Volume Label, and data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

## Expansion Manager

### *ExpCardSectorWrite*

---

itself are all obtained using this function. Applications normally use one of the VFS Manager read functions, which in turn call this one.

**See Also** [ExpCardSectorWrite\(\)](#)

## ExpCardSectorWrite Function

**Purpose** Write contiguous 512-byte sectors to the card in a specified slot.

**Declared In** `ExpansionMgr.h`

**Prototype** `status_t ExpCardSectorWrite (uint16_t slotRefNum,  
uint32_t sectorNumber, uint8_t *bufferP,  
uint32_t *numSectorsP)`

**Parameters**

- *slotRefNum*  
Slot reference number.
- *sectorNumber*  
The sector at which to start writing.
- *bufferP*  
Buffer containing the data being written to the card, or NULL to clear the number of sectors indicated in *numSectorsP*. This buffer must be a multiple of 512 bytes long.
- ↔ *numSectorsP*  
When calling this function, set this to the number of sectors to write. Upon return, it is set to the number of sectors actually written.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`expErrNotOpen`  
Block device driver library has not been opened

`expErrInvalidSlotRefNum`  
Slot reference number is not valid.

`expErrCardNoSectorReadWrite`  
The card does not support the block read/write API.

`expErrCardReadOnly`  
The card is read only

`expErrCardBadSector`  
One of the requested sectors is bad.

`expErrCardProtectedSector`  
One of the requested sectors is protected.

`expErrCardNotPresent`  
No card is present.

**Comments** This function is used by the file system library to write data to the media. Information such as the Volume Label, File Name, File Attributes, as well as the files and directories themselves are all created with this function. Applications normally use one of the VFS Manager write functions, which in turn call this one.

## ExpSlotCustomControl Function

**Purpose** Handle a custom call for a particular slot.

**Declared In** `ExpansionMgr.h`

**Prototype**  

```
status_t ExpSlotCustomControl
    (uint16_t slotRefNum, uint32_t apiCreator,
     uint16_t apiSelector, void *valueP,
     uint16_t *valueLenP)
```

**Parameters**

- `slotRefNum`  
Slot reference number.
- `apiCreator`  
Registered creator code.
- `apiSelector`  
Custom operation to perform. The value of this parameter also determines what kind of data is used with `valueP`.
- ↔ `valueP`  
Buffer containing data specific to the operation specified in `apiSelector`.
- ↔ `valueLenP`  
Size of the `valueP` buffer on entry, size of data written to `valueP` on exit.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`expErrNotOpen`  
Block device driver library has not been opened

## Expansion Manager

### *ExpSlotEnumerate*

---

`expErrUnsupportedOperation`

The selector and/or creator is unsupported or undefined.

`sysErrParamErr`

`valueP` buffer is too small

**Comments** This function is used to make custom calls to the block device driver, calls beyond what is supported by the VFS and Expansion Managers. Depending on the block device driver, custom calls might allow you to do such things as read and write physical blocks on the media, obtain special information for diagnostics, or help with debugging.

Block device drivers identify each call by a registered creator code and a selector.

## ExpSlotEnumerate Function

**Purpose** Iterates through valid slot numbers.

**Declared In** `ExpansionMgr.h`

**Prototype** `status_t ExpSlotEnumerate (uint16_t *slotRefNumP, uint32_t *slotIteratorP)`

**Parameters**  $\leftarrow$  `slotRefNumP`

Reference number of the currently-enumerated slot.

$\rightarrow$  `slotIteratorP`

Pointer to the index of the last entry enumerated. For the first iteration, initialize this parameter to the constant `expIteratorStart`. Upon return this references the next entry in the directory. If this is the last entry, this parameter is set to `expIteratorStop`.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`expErrEnumerationEmpty`

There are no slots left to enumerate. `slotRefNumP` is set to the slot number of the currently enumerated slot or `invalidSlotRefNum` if there are no slots.

**Comments** This function iterates through the device's slots. The first time this function is called, set `*slotIteratorP` to `expIteratorStart` to find the initial slot. Once set this value is changed with each

subsequent call to this function until it reaches the maximum number of slots, at which point `ExpSlotEnumerate()` sets `*slotIteratorP` to `expIteratorStop`.

**Example** The following is an example of how `ExpSlotEnumerate()` should be used:

---

```
UInt16 slotRefNum;
UInt32 slotIterator = expIteratorStart;
while (slotIterator != expIteratorStop) {
    if ((err = ExpSlotEnumerate(&slotRefNum,
                               &slotIterator)) == errNone) {
        // do something with the slotRefNum
    } else {
        // handle error (by breaking out of the
        // loop, most likely
    }
}
```

---

**See Also** [ExpCardInfo\(\)](#), [ExpCardPresent\(\)](#)

## ExpSlotMediaType Function

- Purpose** Determine the media type supported by the given slot.
- Declared In** `ExpansionMgr.h`
- Prototype** `status_t ExpSlotMediaType (uint16_t slotRefNum, uint32_t *mediaTypeP)`
- Parameters**
- `slotRefNum`  
Slot reference number.
  - ← `mediaTypeP`  
Set to the media type supported by the given slot. See “[Defined Media Types](#)” on page 262 for a list of possible media types.
- Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:
- `expErrInvalidSlotRefNum`  
The specified slot number is invalid.
  - `expErrNotOpen`  
The block device driver library has not been opened.

## Expansion Manager

*ExpSlotPowerCheck*

---

**Comments** This function returns the type of media that is supported by the slot, irrespective of whether there is a card in the slot.

**See Also** [ExpCardMediaType\(\)](#)

### ExpSlotPowerCheck Function

**Purpose** Determine if enough power is available to complete an operation before starting it.

**Declared In** `ExpansionMgr.h`

**Prototype** `status_t ExpSlotPowerCheck (uint16_t slotRefNum, uint16_t operationFlags, uint16_t readBlocks, uint16_t writeBlocks)`

**Parameters**

- `slotRefNum`  
Slot reference number.
- `operationFlags`  
Flags for specific operations. The two flags currently supported are:
  - `slotLibPowerFlag_WakeUp`  
Add the power required to bring the slot hardware out of low-power mode.
  - `slotLibPowerFlag_FormatMedia`  
Add the power required to perform a low-level format of the card media.
- `readBlocks`  
Add the power required to read *n* blocks into main memory
- `writeBlocks`  
Add the power required to write *n* blocks onto the card media

**Returns** Returns `errNone` if the necessary power is available, `expErrNotEnoughPower` if the necessary power is *not* available, or one of the following if an error occurred:

- `expErrInvalidSlotRefNum`  
The specified slot number is invalid.
- `expErrNotOpen`  
The block device driver library has not been opened.

`expErrCardNoSectorReadWrite`

The card does not support the read/write API, but `readBlocks` or `writeBlocks` was non-zero.

`expErrCardNotPresent`

No card is present.

### Comments

This function adds up all the power indicated by the parameters and returns `expErrNotEnoughPower` if there isn't enough battery power to perform the operation, `errNone` if there is, or an error code if the operation couldn't be completed successfully.

When formatting a volume, for example, pass `slotLibPowerFlag_FormatMedia` to indicate power is needed for a low level format of the card, and pass a number greater than zero for `writeBlocks` to indicate power is required to install an empty file system.

## **Expansion Manager**

*ExpSlotPowerCheck*

---

# Fatal Alert

---

This chapter describes the Fatal Alert APIs, which provide a simple way to get the user's attention when a serious error occurs. The material in this chapter is organized as follows:

|                                                            |     |
|------------------------------------------------------------|-----|
| <a href="#">Fatal Alert Constants</a> . . . . .            | 277 |
| <a href="#">Fatal Alert Functions and Macros</a> . . . . . | 278 |

The header file `FatalAlert.h` declares the API that this chapter describes.

## Fatal Alert Constants

### Fatal Alert Actions

|                    |                                                                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                                                                                  |
| <b>Declared In</b> | <code>FatalAlert.h</code>                                                                                                                                        |
| <b>Constants</b>   | <pre>#define fatalDoNothing 0xFFFFU  #define fatalEnter68KDebugger 2  #define fatalEnterBothDebugger 3  #define fatalEnterDebugger 1  #define fatalReset 0</pre> |

## Fatal Alert

*Fatal Alert Functions and Macros*

---

# Fatal Alert Functions and Macros

## SysFatalAlert Function

- Purpose** Display a fatal alert until the user taps a button in the alert.
- Declared In** FatalAlert.h
- Prototype** `uint16_t SysFatalAlert (const char *msg)`
- Parameters** → *msg*  
Message to display in the fatal alert dialog.
- Returns** The dialog button that was tapped. The first button is zero.

## SysFatalAlertInit Function

- Purpose**
- Declared In** FatalAlert.h
- Prototype** `void SysFatalAlertInit (void)`
- Parameters** None.
- Returns**

# Feature Manager

---

“Features” provide applications or other components with a means to get and set 32-bit values that persist between invocations of those components. Features are identified by a 32-bit unique creator ID and a 32-bit feature number. The operating system uses the Feature Manager to publish information about the current state of the operating system and the device. Applications are free to use the Feature Manager for similar purposes, or for holding small pieces of data that must persist across application launches. On reset, all features and feature pointers are cleared out and must be re-initialized by the appropriate component.

Feature pointers have traditionally provided a way to allocate chunks of memory larger than the (then very small) dynamic heap, and allow that memory to persist until the device was reset. In Palm OS Cobalt the dynamic heaps are typically much larger and are much more efficient, so reliance on feature pointers is reduced.

This chapter describes the constants and functions that constitute the Feature Manager. It is organized as follows:

|                                                                |     |
|----------------------------------------------------------------|-----|
| <a href="#">Feature Manager Constants</a> . . . . .            | 280 |
| <a href="#">Feature Manager Functions and Macros</a> . . . . . | 280 |

The header file `FeatureMgr.h` declares the API that this chapter describes.

For information on using the APIs documented in this chapter, see [Chapter 2, “Features,”](#) on page 31.

## Feature Manager Constants

### Feature Manager Error Codes

|                    |                                                                                                                                                                                                                                                                                                                        |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error codes returned by the various Feature Manager functions.                                                                                                                                                                                                                                                         |
| <b>Declared In</b> | <code>FeatureMgr.h</code>                                                                                                                                                                                                                                                                                              |
| <b>Constants</b>   | <pre>#define ftrErrInternalErr (ftrErrorClass   5)     An internal error occurred.  #define ftrErrInvalidParam (ftrErrorClass   1)     One of the function parameters is invalid.  #define ftrErrNoSuchFeature (ftrErrorClass   2)     The specified feature number doesn't exist for the specified     creator.</pre> |

## Feature Manager Functions and Macros

### FtrGet Function

|                    |                                                                                                                                                                                                                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Get a feature.                                                                                                                                                                                                                                                                                                |
| <b>Declared In</b> | <code>FeatureMgr.h</code>                                                                                                                                                                                                                                                                                     |
| <b>Prototype</b>   | <pre>status_t FtrGet (uint32_t creator,                 uint32_t featureNum, uint32_t *valueP)</pre>                                                                                                                                                                                                          |
| <b>Parameters</b>  | <p>→ <i>creator</i><br/>Creator ID, which must be registered with PalmSource, Inc. This is usually the same as the creator ID for the application that owns this feature.</p> <p>→ <i>featureNum</i><br/>Feature number of the feature.</p> <p>← <i>valueP</i><br/>Value of the feature is returned here.</p> |
| <b>Returns</b>     | Returns 0 if no error, or <code>ftrErrNoSuchFeature</code> if the specified feature number doesn't exist for the specified creator.                                                                                                                                                                           |

**Comments** The value of the feature is application-dependent.

**See Also** [FtrSet\(\)](#)

## FtrGetByIndex Function

**Purpose** Get a feature by index.

**Declared In** `FeatureMgr.h`

**Prototype** `status_t FtrGetByIndex (uint16_t index,  
Boolean romTable, uint32_t *creatorP,  
uint16_t *numP, uint32_t *valueP)`

**Parameters**

- *index*  
Index of feature.
- *romTable*  
If true, index into ROM table; otherwise, index into RAM table.
- ← *creatorP*  
Feature creator is returned here.
- ← *numP*  
Feature number is returned here.
- ← *valueP*  
Feature value is returned here.

**Returns** Returns `errNone` if no error, or `ftrErrNoSuchFeature` if the index is out of range.

**Comments** This function is intended for system use only. It is used by shell commands. Most applications don't need it.

Until the caller gets back `ftrErrNoSuchFeature`, it should pass indices for each table (ROM, RAM) starting at 0 and incrementing. Note that at system startup, the values in the ROM feature table are copied into the RAM feature table.

### FtrPtrFree Function

- Purpose** Release memory previous allocated with [FtrPtrNew\(\)](#).
- Declared In** `FeatureMgr.h`
- Prototype**

```
status_t FtrPtrFree (uint32_t creator,
                    uint32_t featureNum)
```
- Parameters**
- *creator*  
The creator ID for the feature.
  - *featureNum*  
Feature number of the feature.
- Returns** Returns `errNone` if no error, or `ftrErrNoSuchFeature` if an error occurs.
- Comments** This function unregisters the feature before freeing the memory associated with it.

### FtrPtrGet Function

- Purpose** Retrieve a previously created feature pointer.
- Declared In** `FeatureMgr.h`
- Prototype**

```
status_t FtrPtrGet (uint32_t creator,
                   uint32_t featureNum, size_t *sizeP,
                   void **newPtrP)
```
- Parameters**
- *creator*  
Creator ID, which must be registered with PalmSource, Inc. This is usually the same as the creator ID for the application that owns this feature.
  - *featureNum*  
Feature number of the feature.
  - ← *sizeP*  
Size, in bytes, of the memory chunk allocated to the feature pointer.
  - ← *newPtrP*  
Pointer to the memory chunk is returned here.
- Returns** Returns `errNone` if the operation completed successfully.
- See Also** [FtrPtrNew\(\)](#)

## FtrPtrNew Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Allocate feature memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Declared In</b> | FeatureMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Prototype</b>   | <pre>status_t FtrPtrNew (uint32_t creator,                     uint32_t featureNum, size_t size,                     void **newPtrP)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Parameters</b>  | <p>→ <i>creator</i><br/>Creator ID, which must be registered with PalmSource, Inc. This is usually the same as the creator ID for the application that owns this feature.</p> <p>→ <i>featureNum</i><br/>Feature number of the feature.</p> <p>→ <i>size</i><br/>Size in bytes of the temporary memory to allocate. The maximum chunk size is 64K.</p> <p>← <i>newPtrP</i><br/>Pointer to the memory chunk is returned here.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Returns</b>     | Returns <code>errNone</code> if no error, <code>memErrInvalidParam</code> if the value of <code>size</code> is 0, or <code>memErrNotEnoughSpace</code> if there is not enough space to allocate a chunk of the specified size.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Comments</b>    | <p>This function allocates a chunk of memory and stores a pointer to that chunk in the feature table. The same pointer is returned in <i>newPtrP</i>. The memory chunk remains allocated and locked until the next system reset or until you free the chunk with <a href="#">FtrPtrFree()</a>.</p> <p><code>FtrPtrNew()</code> is useful if you want quick, efficient access to data that persists from one invocation of the application to the next. <code>FtrPtrNew()</code> stores values on the storage heap rather than the dynamic heap, where free space is often extremely limited. The disadvantage to using feature memory is that writing to storage memory is slower than writing to dynamic memory.</p> <p>You can obtain the pointer to the chunk using <a href="#">FtrGet()</a>. To write to the chunk, you <b>must</b> use <a href="#">DmWrite()</a> because the chunk is in the storage heap, not the dynamic heap.</p> |

## Feature Manager

### *FtrPtrResize*

---

For example, if you allocate a memory chunk in this way:

```
FtrPtrNew(appCreator,
          myFtrMemFtr, 32, &ftrMem);
```

You can later access that memory and write to it using the following:

---

```
void* data;
if (!FtrGet(appCreator,
            myFtrMemFtr, (UInt32*)&data))
    DmWrite(data, 0, &someVal, sizeof(someVal));
```

---

**See Also** [FtrPtrGet\(\)](#), [FtrPtrResize\(\)](#)

## FtrPtrResize Function

- Purpose** Resize feature memory.
- Declared In** `FeatureMgr.h`
- Prototype**

```
status_t FtrPtrResize (uint32_t creator,
                      uint32_t featureNum, size_t newSize,
                      void **newPtrP)
```
- Parameters**
- *creator*  
The creator ID for the feature.
  - *featureNum*  
Feature number of the feature.
  - *newSize*  
New size in bytes for the chunk.
  - ← *newPtrP*  
Pointer to the memory chunk is returned here.
- Returns** Returns `errNone` if no error, or `ftrErrNoSuchFeature` if the specified feature number doesn't exist for the specified creator, `memErrInvalidParam` if `newSize` is 0, or `memErrNotEnoughSpace` if there's not enough free space available to allocate a chunk of that size.
- Comments** Use this function to resize a chunk of memory previously allocated by [FtrPtrNew\(\)](#).
- This function may move the chunk to a new location in order to resize it, so it is important to use the pointer returned by this

function when accessing the memory chunk. The pointer in the feature table is automatically updated to be the same as the pointer returned by this function.

If this function fails, the old memory pointer still exists and its data is unchanged.

**See Also** [MemHandleResize\(\)](#)

## FtrSet Function

**Purpose** Set a feature.

**Declared In** `FeatureMgr.h`

**Prototype** `status_t FtrSet (uint32_t creator,  
uint32_t featureNum, uint32_t newValue)`

**Parameters** → *creator*

Creator ID, which must be registered with PalmSource, Inc. This is usually the same as the creator ID for the application that owns this feature.

→ *featureNum*

Feature number for this feature.

→ *newValue*

New value.

**Returns** Returns `errNone` if no error, or `memErrNotEnoughSpace` if the feature table must be resized to add a new feature and no space is available.

**Comments** The value of the feature is application-dependent.

A feature that you define in this manner remains defined until the next system reset or until you explicitly undefine the feature with [FtrUnregister\(\)](#).

**See Also** [FtrGet\(\)](#), [FtrPtrNew\(\)](#)

## FtrUnregister Function

- Purpose** Unregister a feature.
- Declared In** `FeatureMgr.h`
- Prototype** `status_t FtrUnregister (uint32_t creator,  
uint32_t featureNum)`
- Parameters**
- *creator*  
Creator ID for the feature.
  - *featureNum*  
Feature number of the feature.
- Returns** Returns `errNone` if no error, or `ftrErrNoSuchFeature` if the specified feature number doesn't exist for the specified creator.

# Float Manager

---

The Float Manager provides functions for working with single and double-precision floating point values. This chapter provides reference documentation for the Float Manager APIs. It is divided into the following sections:

|                                                              |     |
|--------------------------------------------------------------|-----|
| <a href="#">Float Manager Constants</a> . . . . .            | 287 |
| <a href="#">Float Manager Functions and Macros</a> . . . . . | 288 |

The header file `FloatMgr.h` declares the API that this chapter describes.

For more information on the Float Manager, see [Chapter 12, “Floating Point,”](#) on page 109.

## Float Manager Constants

### Float Manager Error Codes

|                    |                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error codes returned by the various Float Manager functions.                                                                                                                                |
| <b>Declared In</b> | <code>FloatMgr.h</code>                                                                                                                                                                     |
| <b>Constants</b>   | <code>#define flpErrOutOfRange (flpErrorClass   1)</code><br>Returned by <a href="#">FlpBase10Info()</a> if the supplied floating point number is either not a number (NaN) or is infinite. |

### Miscellaneous Float Manager Constants

|                    |                                                                                      |
|--------------------|--------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The Float Manager also defines these constants.                                      |
| <b>Declared In</b> | <code>FloatMgr.h</code>                                                              |
| <b>Constants</b>   | <code>#define flpVersion 0x05000000</code><br>The version of the Float Manager APIs. |

# Float Manager Functions and Macros

## FlpAddDouble Function

- Purpose** Calculate the sum of two double-precision floating point values.
- Declared In** `FloatMgr.h`
- Prototype** `double FlpAddDouble (double addend1,  
double addend2)`
- Parameters**
- *addend1*  
The first double-precision floating point value to be added.
  - *addend2*  
The second double-precision floating point value to be added.
- Returns** Returns the sum of the two supplied values.
- See Also** [FlpAddFloat\(\)](#), [FlpCorrectedAdd\(\)](#), [FlpSubDouble\(\)](#)

## FlpAddFloat Function

- Purpose** Calculate the sum of two single-precision floating point values.
- Declared In** `FloatMgr.h`
- Prototype** `float FlpAddFloat (float addend1, float addend2)`
- Parameters**
- *addend1*  
The first single-precision floating point value to be added.
  - *addend2*  
The second single-precision floating point value to be added.
- Returns** Returns the sum of the two supplied values.
- See Also** [FlpAddDouble\(\)](#), [FlpCorrectedAdd\(\)](#), [FlpSubFloat\(\)](#)

## FlpBase10Info Function

- Purpose** Extract detailed information on the base 10 form of a floating point number: the base 10 mantissa, exponent, and sign.
- Declared In** `FloatMgr.h`
- Prototype**  
`status_t FlpBase10Info (double a,  
          uint32_t *mantissaP, int16_t *exponentP,  
          int16_t *signP)`
- Parameters**
- *a*  
The floating point number.
  - ← *mantissaP*  
The base 10 mantissa.
  - ← *exponentP*  
The base 10 exponent.
  - ← *signP*  
The sign: 1 if the number is negative, 0 otherwise.
- Returns** Returns `errNone` if no error, or `flpErrOutOfRange` if the supplied floating point number is either not a number (NaN) or is infinite.
- Comments** The mantissa is normalized so it contains at least 8 significant digits when printed as an integer value.
- See Also** [FlpGetExponent\(\)](#)

## FlpCompareDoubleEqual Function

- Purpose** Determine whether two double-precision floating point values are equal.
- Declared In** `FloatMgr.h`
- Prototype**  
`Boolean FlpCompareDoubleEqual (double first,  
          double second)`
- Parameters**
- *first*  
The first double-precision floating point value to be compared.

## Float Manager

*FlpCompareDoubleLessThan*

---

→ *second*

The second double-precision floating point value to be compared.

**Returns** Returns `true` if the two double-precision values are equal, `false` otherwise.

**See Also** [FlpCompareDoubleLessThan\(\)](#),  
[FlpCompareDoubleLessThanOrEqual\(\)](#),  
[FlpCompareFloatEqual\(\)](#)

## FlpCompareDoubleLessThan Function

**Purpose** Determine whether one double-precision floating point value is less than another.

**Declared In** `FloatMgr.h`

**Prototype** `Boolean FlpCompareDoubleLessThan (double first,  
double second)`

**Parameters** → *first*

The first double-precision floating point value to be compared.

→ *second*

The second double-precision floating point value to be compared.

**Returns** Returns `true` if the value of *first* is less than the value of *second*. Otherwise, this function returns `false`.

**See Also** [FlpCompareDoubleEqual\(\)](#),  
[FlpCompareDoubleLessThanOrEqual\(\)](#),  
[FlpCompareFloatLessThan\(\)](#)

## FlpCompareDoubleLessThanOrEqual Function

|                    |                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Determine whether one double-precision floating point value is less than or equal to another.                                                                            |
| <b>Declared In</b> | FloatMgr.h                                                                                                                                                               |
| <b>Prototype</b>   | Boolean FlpCompareDoubleLessThanOrEqual<br>(double <i>first</i> , double <i>second</i> )                                                                                 |
| <b>Parameters</b>  | → <i>first</i><br>The first double-precision floating point value to be compared.<br>→ <i>second</i><br>The second double-precision floating point value to be compared. |
| <b>Returns</b>     | Returns true if the value of <i>first</i> is less than or equal to the value of <i>second</i> . Otherwise, this function returns false.                                  |
| <b>See Also</b>    | <a href="#">FlpCompareDoubleEqual()</a> ,<br><a href="#">FlpCompareDoubleLessThan()</a> ,<br><a href="#">FlpCompareFloatLessThanOrEqual()</a>                            |

## FlpCompareFloatEqual Function

|                    |                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Determine whether two single-precision floating point values are equal.                                                                                                  |
| <b>Declared In</b> | FloatMgr.h                                                                                                                                                               |
| <b>Prototype</b>   | Boolean FlpCompareFloatEqual (float <i>first</i> ,<br>float <i>second</i> )                                                                                              |
| <b>Parameters</b>  | → <i>first</i><br>The first single-precision floating point value to be compared.<br>→ <i>second</i><br>The second single-precision floating point value to be compared. |
| <b>Returns</b>     | Returns true if the two single-precision values are equal, false otherwise.                                                                                              |
| <b>See Also</b>    | <a href="#">FlpCompareDoubleEqual()</a> , <a href="#">FlpCompareFloatLessThan()</a> ,<br><a href="#">FlpCompareFloatLessThanOrEqual()</a>                                |

### FlpCompareFloatLessThan Function

- Purpose** Determine whether one single-precision floating point value is less than another.
- Declared In** `FloatMgr.h`
- Prototype** `Boolean FlpCompareFloatLessThan (float first, float second)`
- Parameters**
- *first*  
The first single-precision floating point value to be compared.
  - *second*  
The second single-precision floating point value to be compared.
- Returns** Returns `true` if the value of *first* is less than the value of *second*. Otherwise, this function returns `false`.
- See Also** [FlpCompareDoubleLessThan\(\)](#), [FlpCompareFloatEqual\(\)](#), [FlpCompareFloatLessThanOrEqual\(\)](#)

### FlpCompareFloatLessThanOrEqual Function

- Purpose** Determine whether one single-precision floating point value is less than or equal to another.
- Declared In** `FloatMgr.h`
- Prototype** `Boolean FlpCompareFloatLessThanOrEqual (float first, float second)`
- Parameters**
- *first*  
The first single-precision floating point value to be compared.
  - *second*  
The second single-precision floating point value to be compared.
- Returns** Returns `true` if the value of *first* is less than or equal to the value of *second*. Otherwise, this function returns `false`.
- See Also** [FlpCompareDoubleLessThanOrEqual\(\)](#), [FlpCompareFloatEqual\(\)](#), [FlpCompareFloatLessThan\(\)](#)

## FlpCorrectedAdd Function

- Purpose** Adds two floating point numbers and corrects for least-significant-bit errors when the result should be zero but is instead very close to zero.
- Declared In** `FloatMgr.h`
- Prototype** `double FlpCorrectedAdd (double firstOperand, double secondOperand, int16_t howAccurate)`
- Parameters**
- *firstOperand*  
The first of the two numbers to be added.
  - *secondOperand*  
The second of the two numbers to be added.
  - *howAccurate*  
The smallest difference in exponents that won't force the result to zero. The value returned from `FlpCorrectedAdd ( )` is forced to zero if, when the exponent of the result of the addition is subtracted from the exponent of the smaller of the two operands, the difference exceeds the value specified for *howAccurate*. Supply a value of zero for this parameter to obtain the default level of accuracy (which is equivalent to a *howAccurate* value of 48).
- Returns** Returns the calculated result.
- Comments** Adding or subtracting a large number and a small number produces a result similar in magnitude to the larger number. Adding or subtracting two numbers that are similar in magnitude can, depending on their signs, produce a result with a very small exponent (that is, a negative exponent that is large in magnitude). If the difference between the result's exponent and that of the operands is close to the number of significant bits expressible by the mantissa, it is quite possible that the result should in fact be zero.
- There also exist cases where it may be useful to retain accuracy in the low-order bits of the mantissa. For instance:  $99999999 + 0.00000001 - 99999999$ . However, unless the fractional part is an exact (negative) power of two, it is doubtful that what few bits of mantissa that are available will be enough to properly represent the fractional value. In this example, the  $99999999$  requires 26 bits,

leaving 26 bits for the .00000001; this guarantees inaccuracy after the subtraction.

The problem arises from the difficulty in representing decimal fractions such as 0.1 in binary. After about three successive additions or subtractions, errors begin to appear in the least significant bits of the mantissa. If the value represented by the most significant bits of the mantissa is then subtracted away, the least significant bit error is normalized and becomes the actual result—when in fact the result should be zero.

This problem is only an issue for addition and subtraction.

**See Also** [FlpAddDouble\(\)](#), [FlpAddFloat\(\)](#), [FlpCorrectedSub\(\)](#)

### FlpCorrectedSub Function

**Purpose** Subtracts two floating point numbers and corrects for least-significant-bit errors when the result should be zero but is instead very close to zero.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpCorrectedSub (double firstOperand, double secondOperand, int16_t howAccurate)`

**Parameters**

- *firstOperand*  
The value from which *secondOperand* is to be subtracted.
- *secondOperand*  
The value to subtract from *firstOperand*.
- *howAccurate*  
The smallest difference in exponents that won't force the result to zero. The value returned from `FlpCorrectedSub()` is forced to zero if, when the exponent of the result of the subtraction is subtracted from the exponent of the smaller of the two operands, the difference exceeds the value specified for *howAccurate*. Supply a value of zero for this parameter to obtain the default level of accuracy (which is equivalent to a *howAccurate* value of 48).

**Returns** Returns the calculated result.

**Comments** See the comments for [FlpCorrectedAdd\(\)](#).

**See Also** [FlpSubDouble\(\)](#), [FlpSubFloat\(\)](#)

## FlpDivDouble Function

**Purpose** Divide one double-precision floating point value by another, and return the result.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpDivDouble (double numerator,  
double denominator)`

**Parameters** → *numerator*

The double-precision value to be divided by the denominator.

→ *denominator*

The double-precision value by which the numerator is to be divided.

**Returns** Returns the double-precision result of dividing *numerator* by *denominator*.

**See Also** [FlpDivFloat\(\)](#), [FlpMulDouble\(\)](#)

## FlpDivFloat Function

**Purpose** Divide one single-precision floating point value by another, and return the result.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpDivFloat (float numerator,  
float denominator)`

**Parameters** → *numerator*

The single-precision value to be divided by the denominator.

→ *denominator*

The single-precision value by which the numerator is to be divided.

## Float Manager

*FlpDoubleToFloat*

---

**Returns** Returns the single-precision result of dividing *numerator* by *denominator*.

**See Also** [FlpDivDouble\(\)](#), [FlpMulFloat\(\)](#)

### FlpDoubleToFloat Function

**Purpose** Converts a double-precision floating point value to a float.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpDoubleToFloat (double value)`

**Parameters**  $\rightarrow$  *value*  
A double-precision floating point value.

**Returns** The single-precision floating point representation of the supplied value.

**See Also** [FlpDoubleToInt32\(\)](#), [FlpDoubleToLongDouble\(\)](#), [FlpDoubleToLongLong\(\)](#), [FlpDoubleToUInt32\(\)](#), [FlpDoubleToULongLong\(\)](#), [FlpFloatToDouble\(\)](#)

### FlpDoubleToInt32 Function

**Purpose** Converts a double-precision floating point value to a signed 32-bit integer.

**Declared In** `FloatMgr.h`

**Prototype** `int32_t FlpDoubleToInt32 (double value)`

**Parameters**  $\rightarrow$  *value*  
A double-precision floating point value.

**Returns** The signed 32-bit integer representation of the supplied value.

**See Also** [FlpDoubleToFloat\(\)](#), [FlpDoubleToLongDouble\(\)](#), [FlpDoubleToLongLong\(\)](#), [FlpDoubleToUInt32\(\)](#), [FlpDoubleToULongLong\(\)](#), [FlpInt32ToDouble\(\)](#)

## FlpDoubleToLongDouble Function

|                    |                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Converts a double-precision floating point value to a “long double.”                                                                                                                                                                           |
| <b>Declared In</b> | FloatMgr.h                                                                                                                                                                                                                                     |
| <b>Prototype</b>   | long double FlpDoubleToLongDouble (double <i>value</i> )                                                                                                                                                                                       |
| <b>Parameters</b>  | → <i>value</i><br>A double-precision floating point value.                                                                                                                                                                                     |
| <b>Returns</b>     | The “long double” floating point representation of the supplied value.                                                                                                                                                                         |
| <b>See Also</b>    | <a href="#">FlpDoubleToFloat()</a> , <a href="#">FlpDoubleToInt32()</a> ,<br><a href="#">FlpDoubleToLongLong()</a> , <a href="#">FlpDoubleToUInt32()</a> ,<br><a href="#">FlpDoubleToULongLong()</a> , <a href="#">FlpLongDoubleToDouble()</a> |

## FlpDoubleToLongLong Function

|                    |                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Converts a double-precision floating point value to a “long long.”                                                                                                                                                                             |
| <b>Declared In</b> | FloatMgr.h                                                                                                                                                                                                                                     |
| <b>Prototype</b>   | int64_t FlpDoubleToLongLong (double <i>value</i> )                                                                                                                                                                                             |
| <b>Parameters</b>  | → <i>value</i><br>A double-precision floating point value.                                                                                                                                                                                     |
| <b>Returns</b>     | The signed “long long” integer representation of the supplied value.                                                                                                                                                                           |
| <b>See Also</b>    | <a href="#">FlpDoubleToFloat()</a> , <a href="#">FlpDoubleToInt32()</a> ,<br><a href="#">FlpDoubleToLongDouble()</a> , <a href="#">FlpDoubleToUInt32()</a> ,<br><a href="#">FlpDoubleToULongLong()</a> , <a href="#">FlpLongLongToDouble()</a> |

## FlpDoubleToUInt32 Function

|                    |                                                                                 |
|--------------------|---------------------------------------------------------------------------------|
| <b>Purpose</b>     | Converts a double-precision floating point value to an unsigned 32-bit integer. |
| <b>Declared In</b> | FloatMgr.h                                                                      |
| <b>Prototype</b>   | uint32_t FlpDoubleToUInt32 (double <i>value</i> )                               |
| <b>Parameters</b>  | → <i>value</i><br>A double-precision floating point value.                      |

## Float Manager

*FlpDoubleToULongLong*

---

**Returns** The unsigned 32-bit integer representation of the supplied value.

**See Also** [FlpDoubleToFloat\(\)](#), [FlpDoubleToInt32\(\)](#),  
[FlpDoubleToLongDouble\(\)](#), [FlpDoubleToLongLong\(\)](#),  
[FlpDoubleToULongLong\(\)](#), [FlpUInt32ToDouble\(\)](#)

### FlpDoubleToULongLong Function

**Purpose** Converts a double-precision floating point value to an unsigned “long long.”

**Declared In** `FloatMgr.h`

**Prototype** `uint64_t FlpDoubleToULongLong (double value)`

**Parameters**  $\rightarrow$  *value*

A double-precision floating point value.

**Returns** The unsigned “long long” integer representation of the supplied value.

**See Also** [FlpDoubleToFloat\(\)](#), [FlpDoubleToInt32\(\)](#),  
[FlpDoubleToLongDouble\(\)](#), [FlpDoubleToLongLong\(\)](#),  
[FlpDoubleToUInt32\(\)](#), [FlpULongLongToDouble\(\)](#)

### FlpFloatToDouble Function

**Purpose** Converts a single-precision floating point value to a double.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpFloatToDouble (float value)`

**Parameters**  $\rightarrow$  *value*

A single-precision floating point value.

**Returns** The double-precision floating point representation of the supplied value.

**See Also** [FlpDoubleToFloat\(\)](#), [FlpFloatToInt32\(\)](#),  
[FlpFloatToLongDouble\(\)](#), [FlpFloatToLongLong\(\)](#),  
[FlpFloatToUInt32\(\)](#), [FlpFloatToULongLong\(\)](#)

## FlpFloatToInt32 Function

- Purpose** Converts a single-precision floating point value to a 32-bit signed integer.
- Declared In** `FloatMgr.h`
- Prototype** `int32_t FlpFloatToInt32 (float value)`
- Parameters** `→ value`  
A single-precision floating point value.
- Returns** The 32-bit signed integer representation of the supplied value.
- See Also** [FlpFloatToDouble\(\)](#), [FlpFloatToLongDouble\(\)](#), [FlpFloatToLongLong\(\)](#), [FlpFloatToUInt32\(\)](#), [FlpFloatToULongLong\(\)](#), [FlpInt32ToFloat\(\)](#)

## FlpFloatToLongDouble Function

- Purpose** Converts a single-precision floating point value to a double.
- Declared In** `FloatMgr.h`
- Prototype** `long double FlpFloatToLongDouble (float value)`
- Parameters** `→ value`  
A single-precision floating point value.
- Returns** The double-precision floating point representation of the supplied value.
- See Also** [FlpFloatToDouble\(\)](#), [FlpFloatToInt32\(\)](#), [FlpFloatToLongLong\(\)](#), [FlpFloatToUInt32\(\)](#), [FlpFloatToULongLong\(\)](#), [FlpLongDoubleToFloat\(\)](#)

## FlpFloatToLongLong Function

- Purpose** Converts a single-precision floating point value to a signed “long long” integer.
- Declared In** `FloatMgr.h`
- Prototype** `int64_t FlpFloatToLongLong (float value)`
- Parameters** `→ value`  
A single-precision floating point value.

## Float Manager

*FlpFloatToUInt32*

---

**Returns** The signed long long integer representation of the supplied value.

**See Also** [FlpFloatToDouble\(\)](#), [FlpFloatToInt32\(\)](#),  
[FlpFloatToLongDouble\(\)](#), [FlpFloatToUInt32\(\)](#),  
[FlpFloatToULongLong\(\)](#), [FlpLongLongToFloat\(\)](#)

### FlpFloatToUInt32 Function

**Purpose** Converts a single-precision floating point value to an unsigned 32-bit integer.

**Declared In** `FloatMgr.h`

**Prototype** `uint32_t FlpFloatToUInt32 (float value)`

**Parameters**  $\rightarrow$  *value*  
A single-precision floating point value.

**Returns** The unsigned 32-bit integer representation of the supplied value.

**See Also** [FlpFloatToDouble\(\)](#), [FlpFloatToInt32\(\)](#),  
[FlpFloatToLongDouble\(\)](#), [FlpFloatToLongLong\(\)](#),  
[FlpFloatToULongLong\(\)](#), [FlpUInt32ToFloat\(\)](#)

### FlpFloatToULongLong Function

**Purpose** Converts a single-precision floating point value to an unsigned “long long” integer.

**Declared In** `FloatMgr.h`

**Prototype** `uint64_t FlpFloatToULongLong (float value)`

**Parameters**  $\rightarrow$  *value*  
A single-precision floating point value.

**Returns** The unsigned long long integer representation of the supplied value.

**See Also** [FlpFloatToDouble\(\)](#), [FlpFloatToInt32\(\)](#),  
[FlpFloatToLongDouble\(\)](#), [FlpFloatToLongLong\(\)](#),  
[FlpFloatToUInt32\(\)](#), [FlpULongLongToFloat\(\)](#)

## FlpFToA Function

- Purpose** Convert a floating-point number to a null-terminated ASCII string in exponential format: `[-]x.yyyyyyye[-]zz`
- Declared In** `FloatMgr.h`
- Prototype** `status_t FlpFToA(double value, char *buffer)`
- Parameters**
- *value*  
Floating-point number.
  - ← *buffer*  
Pointer to a buffer that will receive the ASCII string.
- Returns** Returns `errNone` if no error, or `flpErrOutOfRange` if the supplied value is either infinite or not a number. In this case, the buffer is set to the string “INF”, “-INF”, or “NaN” as appropriate.

## FlpGetExponent Macro

- Purpose** Extracts the exponent of a 64-bit floating point value. The returned value has the bias applied, so it ranges from -1023 to +1024.
- Declared In** `FloatMgr.h`
- Prototype** `#define FlpGetExponent (x)`
- Parameters**
- *x*  
The value from which the exponent is to be extracted.
- Returns** Evaluates to the exponent of the specified value.
- See Also** [FlpBase10Info\(\)](#)

## FlpInt32ToDouble Function

- Purpose** Converts a signed 32-bit integer to a double.
- Declared In** `FloatMgr.h`
- Prototype** `double FlpInt32ToDouble (int32_t value)`
- Parameters**
- *value*  
A signed 32-bit integer value.

## Float Manager

*FlpInt32ToFloat*

---

**Returns** The double-precision floating point representation of the supplied value.

**See Also** [FlpDoubleToInt32\(\)](#), [FlpInt32ToFloat\(\)](#)

### FlpInt32ToFloat Function

**Purpose** Converts a signed 32-bit integer to a float.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpInt32ToFloat (int32_t value)`

**Parameters** `→ value`  
A signed 32-bit integer value.

**Returns** The floating point representation of the supplied value.

**See Also** [FlpFloatToInt32\(\)](#), [FlpInt32ToDouble\(\)](#)

### FlpLongDoubleToDouble Function

**Purpose** Converts a long double-precision floating point value to a double.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpLongDoubleToDouble (long double value)`

**Parameters** `→ value`  
A long-double-precision floating point value.

**Returns** The double-precision floating point representation of the supplied value.

**See Also** [FlpDoubleToLongDouble\(\)](#), [FlpLongDoubleToFloat\(\)](#)

### FlpLongDoubleToFloat Function

**Purpose** Converts a long double-precision floating point value to a float.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpLongDoubleToFloat (long double value)`

**Parameters** `→ value`  
A long-double-precision floating point value.

**Returns** The single-precision floating point representation of the supplied value.

**See Also** [FlpFloatToLongDouble\(\)](#), [FlpLongDoubleToDouble\(\)](#)

## **FlpLongLongToDouble Function**

**Purpose** Converts a signed 64-bit integer to a double.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpLongLongToDouble (int64_t value)`

**Parameters** `→ value`  
A signed 64-bit integer value.

**Returns** The double-precision floating point representation of the supplied value.

**See Also** [FlpDoubleToLongLong\(\)](#), [FlpLongLongToFloat\(\)](#)

## **FlpLongLongToFloat Function**

**Purpose** Converts a signed 64-bit integer to a float.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpLongLongToFloat (int64_t value)`

**Parameters** `→ value`  
A signed 64-bit integer value.

**Returns** The floating point representation of the supplied value.

**See Also** [FlpFloatToLongLong\(\)](#), [FlpLongLongToDouble\(\)](#)

### FlpMulDouble Function

- Purpose** Multiply one double-precision floating point value by another, and return the result.
- Declared In** `FloatMgr.h`
- Prototype** `double FlpMulDouble (double multiplier,  
double multiplicand)`
- Parameters**
- *multiplier*  
The first double-precision floating point value to be multiplied.
  - *multiplicand*  
The second double-precision floating point value to be multiplied.
- Returns** The double-precision result of multiplying *multiplier* by *multiplicand*.
- See Also** [FlpDivDouble\(\)](#), [FlpMulFloat\(\)](#)

### FlpMulFloat Function

- Purpose** Multiply one second-precision floating point value by another, and return the result.
- Declared In** `FloatMgr.h`
- Prototype** `float FlpMulFloat (float multiplier,  
float multiplicand)`
- Parameters**
- *multiplier*  
The first single-precision floating point value to be multiplied.
  - *multiplicand*  
The second single-precision floating point value to be multiplied.
- Returns** The single-precision result of multiplying *multiplier* by *multiplicand*.
- See Also** [FlpDivFloat\(\)](#), [FlpMulDouble\(\)](#)

## FlpNegDouble Function

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| <b>Purpose</b>     | Calculate the negative of a double-precision floating point value.       |
| <b>Declared In</b> | <code>FloatMgr.h</code>                                                  |
| <b>Prototype</b>   | <code>double FlpNegDouble (double <i>value</i>)</code>                   |
| <b>Parameters</b>  | <code>→ <i>value</i></code><br>The double-precision value to be negated. |
| <b>Returns</b>     | Returns the negative of the supplied value.                              |
| <b>See Also</b>    | <a href="#"><code>FlpNegFloat()</code></a>                               |

## FlpNegFloat Function

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| <b>Purpose</b>     | Calculate the negative of a single-precision floating point value.       |
| <b>Declared In</b> | <code>FloatMgr.h</code>                                                  |
| <b>Prototype</b>   | <code>float FlpNegFloat (float <i>value</i>)</code>                      |
| <b>Parameters</b>  | <code>→ <i>value</i></code><br>The single-precision value to be negated. |
| <b>Returns</b>     | Returns the negative of the supplied value.                              |
| <b>See Also</b>    | <a href="#"><code>FlpNegDouble()</code></a>                              |

## FlpSubDouble Function

|                    |                                                                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Subtract one double-precision floating point value from another.                                                                                                                                                                              |
| <b>Declared In</b> | <code>FloatMgr.h</code>                                                                                                                                                                                                                       |
| <b>Prototype</b>   | <code>double FlpSubDouble (double <i>minuend</i>,<br/>double <i>subtrahend</i>)</code>                                                                                                                                                        |
| <b>Parameters</b>  | <code>→ <i>minuend</i></code><br>The double-precision floating point value from which the subtrahend is to be subtracted.<br><code>→ <i>subtrahend</i></code><br>The double-precision floating point value to be subtracted from the minuend. |

## Float Manager

*FlpSubFloat*

---

**Returns** Returns the result of subtracting *subtrahend* from *minuend*.

**See Also** [FlpAddDouble\(\)](#), [FlpCorrectedSub\(\)](#), [FlpSubFloat\(\)](#)

### FlpSubFloat Function

**Purpose** Subtract one single-precision floating point value from another.

**Declared In** `FloatMgr.h`

**Prototype** `float FlpSubFloat (float minuend,  
float subtrahend)`

**Parameters** → *minuend*

The single-precision floating point value from which the subtrahend is to be subtracted.

→ *subtrahend*

The single-precision floating point value to be subtracted from the minuend.

**Returns** Returns the result of subtracting *subtrahend* from *minuend*.

**See Also** [FlpAddFloat\(\)](#), [FlpCorrectedSub\(\)](#), [FlpSubDouble\(\)](#)

### FlpUInt32ToDouble Function

**Purpose** Converts an unsigned 32-bit integer to a double.

**Declared In** `FloatMgr.h`

**Prototype** `double FlpUInt32ToDouble (uint32_t value)`

**Parameters** → *value*

An unsigned 32-bit integer value.

**Returns** The double-precision floating point representation of the supplied value.

## FlpUInt32ToFloat Function

- Purpose** Converts an unsigned 32-bit integer to a float.
- Declared In** `FloatMgr.h`
- Prototype** `float FlpUInt32ToFloat (uint32_t value)`
- Parameters** `→ value`  
An unsigned 32-bit integer value.
- Returns** The floating point representation of the supplied value.
- See Also** [FlpFloatToUInt32\(\)](#), [FlpUInt32ToDouble\(\)](#)

## FlpUlongLongToDouble Function

- Purpose** Converts an unsigned 64-bit integer to a double.
- Declared In** `FloatMgr.h`
- Prototype** `double FlpUlongLongToDouble (uint64_t value)`
- Parameters** `→ value`  
An unsigned 64-bit integer value.
- Returns** The double-precision floating point representation of the supplied value.
- See Also** [FlpDoubleToUlongLong\(\)](#), [FlpUlongLongToFloat\(\)](#)

## FlpUlongLongToFloat Function

- Purpose** Converts an unsigned 64-bit integer to a float.
- Declared In** `FloatMgr.h`
- Prototype** `float FlpUlongLongToFloat (uint64_t value)`
- Parameters** `→ value`  
An unsigned 64-bit integer value.
- Returns** The floating point representation of the supplied value.
- See Also** [FlpFloatToUlongLong\(\)](#), [FlpUlongLongToDouble\(\)](#)

## **Float Manager**

*FlpULongLongToFloat*

---

# Host Control

---

This chapter describes the host control API, which you can use to call Emulator-defined functions while your application is running under the Palm OS® Emulator. For example, you can make function calls to start and stop profiling in the Emulator.

This chapter is organized as follows:

|                                                   |     |
|---------------------------------------------------|-----|
| <a href="#">Host Control Structures and Types</a> | 310 |
| <a href="#">Host Control Constants</a>            | 318 |
| <a href="#">Host Control Functions and Macros</a> | 328 |

The header file `HostControl.h` declares the API that this chapter describes.

The functions documented in this chapter are invoked by executing a trap/selector combination that is defined for use by the Emulator and other foreign host environments. Palm OS Emulator catches the calls intended for it that are made to this selector.

---

**IMPORTANT:** These functions are not for use on actual devices; they are only intended to be called from code running in a simulated environment such as Palm OS Emulator.

---

## Host Control Structures and Types

### HostBool Typedef

|                    |                                             |
|--------------------|---------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostBoolType</a> . |
| <b>Declared In</b> | HostControl.h                               |
| <b>Prototype</b>   | <code>typedef HostBoolType HostBool</code>  |

### HostBoolType Typedef

|                    |                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A Boolean value type, used to pass Boolean values to and receive Boolean values from various host control functions. |
| <b>Declared In</b> | HostControl.h                                                                                                        |
| <b>Prototype</b>   | <code>typedef long HostBoolType</code>                                                                               |

### HostClockType Typedef

|                    |                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A platform-independent representation of the standard C library <code>clock_t</code> type. |
| <b>Declared In</b> | HostControl.h                                                                              |
| <b>Prototype</b>   | <code>typedef long HostClockType</code>                                                    |

### HostControlSelectorType Typedef

|                    |                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains a Host Control function selector value.                                                               |
| <b>Declared In</b> | HostControl.h                                                                                                  |
| <b>Prototype</b>   | <code>typedef uint16_t HostControlSelectorType</code>                                                          |
| <b>Comments</b>    | See “ <a href="#">Host Control Function Selectors</a> ” on page 318 for a complete list of function selectors. |

## HostControlTrapNumber Typedef

**Purpose** An alias for [HostControlSelectorType](#).

**Declared In** `HostControl.h`

**Prototype**  

```
typedef HostControlSelectorType  
        HostControlTrapNumber
```

## HostDirEntType Struct

**Purpose** A platform-specific type (usually just a null-terminated file name) that contains a directory name.

**Declared In** `HostControl.h`

**Prototype**  

```
struct HostDirEntType  
{  
    char d_name[HOST_NAME_MAX + 1];  
};  
typedef struct HostDirEntType HostDirEntType
```

**Fields** `d_name`  
The directory name.

**Comments** This type is used by [HostReadDir\(\)](#) to return a directory name.

## HostDIRType Struct

**Purpose** Type that represents an open directory whose contents can be read.

**Declared In** `HostControl.h`

**Prototype**  

```
struct HostDIRType  
{  
    long _field;  
};  
typedef struct HostDIRType HostDIRType
```

**Fields** `_field`

**Comments** Values of this type are returned by [HostOpenDir\(\)](#). The [HostReadDir\(\)](#) and [HostCloseDir\(\)](#) functions take values of this type.

### HostErr Typedef

|                    |                                            |
|--------------------|--------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostErrType</a> . |
| <b>Declared In</b> | HostControl.h                              |
| <b>Prototype</b>   | <code>typedef HostErrType HostErr</code>   |

### HostErrType Typedef

|                    |                                            |
|--------------------|--------------------------------------------|
| <b>Purpose</b>     | A type that contains a host control error. |
| <b>Declared In</b> | HostControl.h                              |
| <b>Prototype</b>   | <code>typedef long HostErrType</code>      |

### HostFILE Typedef

|                    |                                             |
|--------------------|---------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostFILEType</a> . |
| <b>Declared In</b> | HostControl.h                               |
| <b>Prototype</b>   | <code>typedef HostFILEType HostFILE</code>  |

### HostFILEType Struct

|                    |                                                                                                                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Type that represents an open file whose contents can be manipulated.                                                                                                                                                         |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                |
| <b>Prototype</b>   | <pre>struct HostFILEType {     long _field; }; typedef struct HostFILEType HostFILEType</pre>                                                                                                                                |
| <b>Fields</b>      | <code>_field</code>                                                                                                                                                                                                          |
| <b>Comments</b>    | The host control API defines <code>HostFILEType</code> for the standard C library functions that take <code>FILE*</code> parameters. It is returned by <a href="#">HostFOpen()</a> and used by other host control functions. |

## HostID Typedef

|                    |                                           |
|--------------------|-------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostIDType</a> . |
| <b>Declared In</b> | HostControl.h                             |
| <b>Prototype</b>   | <code>typedef HostIDType HostID</code>    |

## HostIDType Typedef

|                    |                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A host identifier.                                                                                                                                             |
| <b>Declared In</b> | HostControl.h                                                                                                                                                  |
| <b>Prototype</b>   | <code>typedef long HostIDType</code>                                                                                                                           |
| <b>Comments</b>    | Retrieve a host's identifier with <a href="#">HostGetHostID()</a> . See the " <a href="#">Host IDs</a> " enum for the set of values that this type can assume. |

## HostPlatform Typedef

|                    |                                                    |
|--------------------|----------------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostPlatformType</a> .    |
| <b>Declared In</b> | HostControl.h                                      |
| <b>Prototype</b>   | <code>typedef HostPlatformType HostPlatform</code> |

## HostPlatformType Typedef

|                    |                                                                                                                                                                               |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A platform identifier.                                                                                                                                                        |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                 |
| <b>Prototype</b>   | <code>typedef long HostPlatformType</code>                                                                                                                                    |
| <b>Comments</b>    | Retrieve a host's platform type with <a href="#">HostGetHostPlatform()</a> . See the " <a href="#">Host Platforms</a> " enum for the set of values that this type can assume. |

### HostSignal Typedef

|                    |                                                |
|--------------------|------------------------------------------------|
| <b>Purpose</b>     | An alias for <a href="#">HostSignalType</a> .  |
| <b>Declared In</b> | HostControl.h                                  |
| <b>Prototype</b>   | <code>typedef HostSignalType HostSignal</code> |

### HostSignalType Typedef

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A host signal, used by functions such as <a href="#">HostSignalSend()</a> .                    |
| <b>Declared In</b> | HostControl.h                                                                                  |
| <b>Prototype</b>   | <code>typedef long HostSignalType</code>                                                       |
| <b>Comments</b>    | See the “ <a href="#">Host Signals</a> ” enum for the set of values that this type can assume. |

### HostSizeType Typedef

|                    |                                                                                    |
|--------------------|------------------------------------------------------------------------------------|
| <b>Purpose</b>     | A platform-independent version of the standard C library <code>size_t</code> type. |
| <b>Declared In</b> | HostControl.h                                                                      |
| <b>Prototype</b>   | <code>typedef long HostSizeType</code>                                             |

### HostStatType Struct

|                    |                       |
|--------------------|-----------------------|
| <b>Purpose</b>     | Contains file status. |
| <b>Declared In</b> | HostControl.h         |

**Prototype**

```
struct HostStatType {
    unsigned long    st_dev_;
    unsigned long    st_ino_;
    unsigned long    st_mode_;
    unsigned long    st_nlink_;
    unsigned long    st_uid_;
    unsigned long    st_gid_;
    unsigned long    st_rdev_;
    HostTimeType     st_atime_;
    HostTimeType     st_mtime_;
    HostTimeType     st_ctime_;
    unsigned long    st_size_;
    unsigned long    st_blksize_;
    unsigned long    st_blocks_;
    unsigned long    st_flags_;
};
typedef struct HostStatType HostStatType
```

**Fields**

**st\_dev\_**  
Drive number of the disk containing the file (the same as st\_rdev\_).

**st\_ino\_**  
Number of the information node for the file (Unix-specific information).

**st\_mode\_**  
Bit mask for file-mode information. The `_S_IFDIR` bit indicates if this is a directory; the `_S_IFREG` bit indicates an ordinary file or device. User read/write bits are set according to the file's permission mode; user execute bits are set according to the filename extension.

**st\_nlink\_**  
Always 1 on non-NTFS file systems.

**st\_uid\_**  
Numeric identifier of the user who owns the file (Unix-specific information).

**st\_gid\_**  
Numeric identifier of the group who owns the file (Unix-specific information).

## Host Control

### *HostTimeType*

---

`st_rdev_`  
Drive number of the disk containing the file (the same as `st_dev_`).

`st_atime_`  
Time of the last access of the file.

`st_mtime_`  
Time of the last modification of the file.

`st_ctime_`  
Time of the creation of the file.

`st_size_`  
Size of the file in bytes.

`st_blksize_`  
Block size for the file.

`st_blocks_`  
Number of blocks.

`st_flags_`  
File flags.

**Comments** Retrieve file status information with [HostStat\(\)](#).

## HostTimeType Typedef

**Purpose** A platform-independent version of the standard C library `time_t` type.

**Declared In** `HostControl.h`

**Prototype** `typedef long HostTimeType`

**Comments** This type is used in the [HostStatType](#) and [HostUTimeType](#) structures and by various time-related host control functions.

## HostTmType Struct

**Purpose** Contains a date and time; used by various host control time functions.

**Declared In** HostControl.h

**Prototype**

```
struct HostTmType
{
    long tm_sec_;
    long tm_min_;
    long tm_hour_;
    long tm_mday_;
    long tm_mon_;
    long tm_year_;
    long tm_wday_;
    long tm_yday_;
    long tm_isdst_;
};
typedef struct HostTmType HostTmType
```

**Fields**

- `tm_sec_`  
Seconds after the minute: range from 0 to 59.
- `tm_min_`  
Minutes after the hour: range from 0 to 59.
- `tm_hour_`  
Hours since midnight: range from 0 to 23.
- `tm_mday_`  
Day of the month: range from 1 to 31.
- `tm_mon_`  
Months since January: range from 0 to 11.
- `tm_year_`  
Years since 1900.
- `tm_wday_`  
Days since Sunday: range from 0 to 6.
- `tm_yday_`  
Days since January 1: range from 0 to 365.
- `tm_isdst_`  
Daylight savings time flag.

## HostUTimeType Struct

|                    |                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains the creation, access, and modification times for a file.                                                                                              |
| <b>Declared In</b> | HostControl.h                                                                                                                                                  |
| <b>Prototype</b>   | <pre>struct HostUTimeType {     HostTimeType crtime_;     HostTimeType actime_;     HostTimeType modtime_; }; typedef struct HostUTimeType HostUTimeType</pre> |
| <b>Fields</b>      | <pre>crtime_  asctime_  modtime_</pre>                                                                                                                         |
| <b>Comments</b>    | This structure is used in conjunction with the <a href="#">HostUTime()</a> function to alter the modification time for a file.                                 |

## Host Control Constants

### Host Control Function Selectors

|                    |                                                                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Host control functions are invoked by executing a trap/selector combination that is defined for use by the Emulator and other foreign host environments. Palm OS Emulator catches the calls intended for it that are made to one of these selectors. |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                        |
| <b>Constants</b>   | <pre>#define hostSelectorAscTime 0x0370 #define hostSelectorBase 0x0100 #define hostSelectorClock 0x0371 #define hostSelectorCloseDir 0x0383 #define hostSelectorCTime 0x0372</pre>                                                                  |

```
#define hostSelectorEnteringApp 0x0C03
#define hostSelectorErrNo 0x0300
#define hostSelectorExgLibAccept 0x0586
#define hostSelectorExgLibClose 0x0581
#define hostSelectorExgLibConnect 0x0585
#define hostSelectorExgLibControl 0x058C
#define hostSelectorExgLibDisconnect 0x0587
#define hostSelectorExgLibGet 0x0589
#define hostSelectorExgLibHandleEvent 0x0584
#define hostSelectorExgLibOpen 0x0580
#define hostSelectorExgLibPut 0x0588
#define hostSelectorExgLibReceive 0x058B
#define hostSelectorExgLibRequest 0x058D
#define hostSelectorExgLibSend 0x058A
#define hostSelectorExgLibSleep 0x0582
#define hostSelectorExgLibWake 0x0583
#define hostSelectorExitedApp 0x0C04
#define hostSelectorExportFile 0x0501
#define hostSelectorFClose 0x0301
#define hostSelectorFEOF 0x0302
#define hostSelectorFError 0x0303
#define hostSelectorFFlush 0x0304
#define hostSelectorFGetC 0x0305
#define hostSelectorFGetPos 0x0306
#define hostSelectorFGetS 0x0307
#define hostSelectorFOpen 0x0308
#define hostSelectorFPrintf 0x0309
#define hostSelectorFPutC 0x030A
```

## Host Control

### *Host Control Function Selectors*

---

```
#define hostSelectorFPutS 0x030B
#define hostSelectorFRead 0x030C
#define hostSelectorFree 0x031A
#define hostSelectorFreopen 0x030F
#define hostSelectorFScanF 0x0310
#define hostSelectorFSeek 0x0311
#define hostSelectorFSetPos 0x0312
#define hostSelectorFTell 0x0313
#define hostSelectorFWrite 0x0314
#define hostSelectorGestalt 0x0104
#define hostSelectorGet68KDebuggerPort 0x0C05
#define hostSelectorGetChar 0x0C40
#define hostSelectorGetDirectory 0x0B02
#define hostSelectorGetEnv 0x0317
#define hostSelectorGetFile 0x0B00
#define hostSelectorGetFileAttr 0x03AF
#define hostSelectorGetFirstApp 0x0C43
#define hostSelectorGetHostID 0x0101
#define hostSelectorGetHostPlatform 0x0102
#define hostSelectorGetHostVersion 0x0100
#define hostSelectorGetPreference 0x0600
#define hostSelectorGMTime 0x0374
#define hostSelectorHostControl68K 0x0C08
#define hostSelectorImportFile 0x0500
#define hostSelectorIsCallingTrap 0x0105
#define hostSelectorIsSelectorImplemented 0x0103
#define hostSelectorLastTrapNumber 0x0CFF
#define hostSelectorLocalTime 0x0375
```

```
#define hostSelectorLogEvent 0x0C07
#define hostSelectorLogFile 0x0700
#define hostSelectorMalloc 0x0318
#define hostSelectorMkDir 0x03AA
#define hostSelectorMkTime 0x0376
#define hostSelectorOpenDir 0x0380
#define hostSelectorPrintf 0x0C41
#define hostSelectorProfileCleanup 0x0204
#define hostSelectorProfileDetailFn 0x0205
#define hostSelectorProfileDump 0x0203
#define hostSelectorProfileInit 0x0200
#define hostSelectorProfileStart 0x0201
#define hostSelectorProfileStop 0x0202
#define hostSelectorPutFile 0x0B01
#define hostSelectorReadDir 0x0381
#define hostSelectorRealloc 0x0319
#define hostSelectorRemove 0x030D
#define hostSelectorRename 0x030E
#define hostSelectorRmdir 0x0394
#define hostSelectorSessionClose 0x0802
#define hostSelectorSessionCreate 0x0800
#define hostSelectorSessionOpen 0x0801
#define hostSelectorSessionQuit 0x0803
#define hostSelectorSetErrorLevel 0x0C06
#define hostSelectorSetFileAttr 0x03B0
#define hostSelectorSetLogFileSize 0x0701
#define hostSelectorSetPreference 0x0601
#define hostSelectorSignalResume 0x0806
```

## Host Control

### Host Control Function Selectors

---

```
#define hostSelectorSignalSend 0x0804
#define hostSelectorSignalWait 0x0805
#define hostSelectorSlotHasCard 0x0A02
#define hostSelectorSlotMax 0x0A00
#define hostSelectorSlotRoot 0x0A01
#define hostSelectorStat 0x03AB
#define hostSelectorStrFTime 0x0377
#define hostSelectorTime 0x0378
#define hostSelectorTmpFile 0x0315
#define hostSelectorTmpNam 0x0316
#define hostSelectorTraceClose 0x0901
#define hostSelectorTraceInit 0x0900
#define hostSelectorTraceOutputB 0x0906
#define hostSelectorTraceOutputT 0x0902
#define hostSelectorTraceOutputTL 0x0903
#define hostSelectorTraceOutputVT 0x0904
#define hostSelectorTraceOutputVTL 0x0905
#define hostSelectorTruncate 0x03A7
#define hostSelectorUTime 0x03AE
#define hostSelectorVFPrintf 0x031B
#define hostSelectorVFScanF 0x031C
#define hostSelectorVPrintf 0x0C42
```

#### Comments

You can use the host function selector constants with the [HostIsSelectorImplemented\(\)](#) function to determine if a certain function is implemented on your debugging host. Each constant is the name of a function, with the Host prefix replaced by hostSelector.

## Host Control Error Codes Enum

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error codes returned by the various host control functions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Declared In</b> | <code>HostControl.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Constants</b>   | <pre>#define hostErrorClass 0x1C00</pre> <p>The value upon which all host control errors are based.</p> <pre>hostErrNone = 0</pre> <p>No error.</p> <pre>hostErrBase = hostErrorClass</pre> <p>The value upon which all host control errors are based.</p> <pre>hostErrUnknownGestaltSelector</pre> <p>The specified Gestalt selector value is not valid.</p> <pre>hostErrDiskError</pre> <p>A disk error occurred. The standard C library error code EIO is mapped to this error constant.</p> <pre>hostErrOutOfMemory</pre> <p>There is not enough memory to complete the request. The standard C library error code ENOMEM is mapped to this error constant.</p> <pre>hostErrMemReadOutOfRange</pre> <p>An out of range error occurred during a memory read.</p> <pre>hostErrMemWriteOutOfRange</pre> <p>An out of range error occurred during a memory write.</p> <pre>hostErrMemInvalidPtr</pre> <p>The pointer is not valid.</p> <pre>hostErrInvalidParameter</pre> <p>A parameter to a function is not valid. The standard C library error codes EBADF, EFAULT, and EINVAL are mapped to this error constant.</p> <pre>hostErrTimeout</pre> <p>A timeout occurred.</p> <pre>hostErrInvalidDeviceType</pre> <p>The specified device type is not valid.</p> <pre>hostErrInvalidRAMSize</pre> <p>The specified RAM size value is not valid.</p> |

## Host Control

### Host Control Error Codes

---

#### `hostErrFileNotFound`

The specified file could not be found. The standard C library error code `ENOENT` is mapped to this error constant.

#### `hostErrRPCCall`

A function that must be called remotely was called by an application. These functions include:

[`HostSessionCreate\(\)`](#), [`HostSessionOpen\(\)`](#),  
[`HostSessionClose\(\)`](#), [`HostSessionQuit\(\)`](#),  
[`HostSignalWait\(\)`](#), and [`HostSignalResume\(\)`](#).

#### `hostErrSessionRunning`

A session is already running and one of the following functions was called: [`HostSessionCreate\(\)`](#), [`HostSessionOpen\(\)`](#), or [`HostSessionQuit\(\)`](#).

#### `hostErrSessionNotRunning`

No session is running and the [`HostSessionClose\(\)`](#) function was called.

#### `hostErrNoSignalWaiters`

The [`HostSignalSend\(\)`](#) function was called, but there are no external scripts waiting for a signal.

#### `hostErrSessionNotPaused`

The [`HostSignalResume\(\)`](#) function was called, but the session has not been paused by a call to [`HostSignalSend\(\)`](#).

#### `hostErrPermissions`

The standard C library error codes `EACCES` and `EPERM` are mapped to this error constant.

#### `hostErrFileNameTooLong`

The standard C library error code `ENAMETOOLONG` is mapped to this error constant.

#### `hostErrNotADirectory`

The standard C library error code `ENOTDIR` is mapped to this error constant.

#### `hostErrTooManyFiles`

The standard C library error codes `EMFILE` and `ENFILE` are mapped to this error constant.

#### `hostErrFileTooBig`

The standard C library error code `EFBIG` is mapped to this error constant.

**hostErrReadOnlyFS**

The standard C library error code EROFS is mapped to this error constant.

**hostErrIsDirectory**

The standard C library error code EISDIR is mapped to this error constant.

**hostErrExists**

The standard C library error code EEXIST is mapped to this error constant.

**hostErrOpNotAvailable**

The standard C library error codes ENOSYS and ENODEV are mapped to this error constant.

**hostErrDirNotEmpty**

The standard C library error code ENOTEMPTY is mapped to this error constant.

**hostErrDiskFull**

The standard C library error code ENOSPC is mapped to this error constant.

**hostErrProfilingNotReady**

The profiling system is not ready.

**hostErrUnknownError**

The standard C library error code values that are not mapped to any of the above error constants are mapped to this error constant.

## Host IDs Enum

|                    |                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The <a href="#">HostGetHostID()</a> function uses a Host ID value to specify the debugging host type. |
| <b>Declared In</b> | HostControl.h                                                                                         |
| <b>Constants</b>   | <b>hostIDPalmOS</b><br>A Palm Powered™ device.                                                        |
|                    | <b>hostIDPalmOSEmulator</b><br>The Palm OS Emulator application.                                      |

## Host Control

### Host Platforms

---

`hostIDPalmOSSimulator`

Returned for both Palm OS Simulator and the Macintosh Palm Simulator application.

## Host Platforms Enum

**Purpose** The [HostGetHostPlatform\(\)](#) function uses one of these values to specify operating system hosting the emulation.

**Declared In** `HostControl.h`

**Constants** `hostPlatformPalmOS`  
The Palm OS platform.

`hostPlatformWindows`  
The Windows operating system platform.

`hostPlatformMacintosh`  
The Mac OS platform.

`hostPlatformUnix`  
The Unix operating system platform.

## Host Signals Enum

**Purpose** This section describes the host signal values, which you can use with [HostSignalSend\(\)](#).

**Declared In** `HostControl.h`

**Constants** `hostSignalReserved`  
System-defined signals start here.

`hostSignalIdle`  
Palm OS Emulator is about to go into an idle state.

`hostSignalQuit`  
Palm OS Emulator is about to quit.

`hostSignalUser = 0x40000000`  
User-defined signals start here.

## File Attributes Enum

|                    |                                                                                                                                                                        |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | File or directory attributes. Supply one of these values to <a href="#">HostGetFileAttr()</a> to retrieve the corresponding setting for a specified file or directory. |
| <b>Declared In</b> | HostControl.h                                                                                                                                                          |
| <b>Constants</b>   | <pre>hostFileAttrReadOnly = 1     The file is read-only.  hostFileAttrHidden = 2     The file is hidden.  hostFileAttrSystem = 4     The file is a system file.</pre>  |

## Error Levels

|                    |                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error levels.                                                                                                                              |
| <b>Declared In</b> | HostControl.h                                                                                                                              |
| <b>Constants</b>   | <pre>#define kErrorLevelFatal 2     Fatal error.  #define kErrorLevelNone 0     No error.  #define kErrorLevelWarning 1     Warning.</pre> |

## Miscellaneous Host Control Constants

|                    |                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The Host Control APIs also include these constants.                                                                                                                                                                                                                                                                                                                  |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                                                                                                                                        |
| <b>Constants</b>   | <pre>#define HOST_NAME_MAX 255     The maximum length, not including the terminating null     character, of a directory name as encapsulated in a     <a href="#">HostDirEntType</a> structure.  #define kPalmOSEmulatorFeatureCreator ('pose')     Creator ID of the Palm OS Emulator feature. Pass this value     along with kPalmOSEmulatorFeatureNumber to</pre> |

## Host Control

### Host Control Functions and Macros

---

[FtrGet\(\)](#) to see if you are running under the Palm OS Emulator. If not, `FtrGet()` returns `ftrErrNoSuchFeature`.

```
#define kPalmOSEmulatorFeatureNumber (0)
    Feature number of the Palm OS Emulator feature. Pass this
    value along with kPalmOSEmulatorFeatureCreator to
FtrGet\(\) to see if you are running under the Palm OS
    Emulator. If not, FtrGet() returns
    ftrErrNoSuchFeature.
```

## Host Control Functions and Macros

### HostAscTime Function

- Purpose** Obtain a character string representation of a [HostTmType](#) structure.
- Declared In** `HostControl.h`
- Prototype** `char *HostAscTime (const HostTmType *time)`
- Parameters** `→ time`  
The time for which a character string representation is desired.
- Returns** The time as a character string.
- See Also** [HostCTime\(\)](#)

### HostClock Function

- Purpose** Obtain an elapsed time value.
- Declared In** `HostControl.h`
- Prototype** `HostClockType HostClock (void)`
- Parameters** None.
- Returns** The elapsed time in terms of the operating system's clock function (usually the number clock ticks that have elapsed since the start of the process), or -1 if the function call was not successful.

## HostCloseDir Function

|                    |                                                                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Close a directory.                                                                                                                                                                |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                     |
| <b>Prototype</b>   | <code>long HostCloseDir (HostDIRType *directory)</code>                                                                                                                           |
| <b>Parameters</b>  | <code>→ directory</code><br>The directory to be closed.                                                                                                                           |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred. |

## HostControl Function

|                    |                                                                                                                                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Host control function dispatcher.                                                                                                                                                                                                                     |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                         |
| <b>Prototype</b>   | <code>uint32_t HostControl<br/>(HostControlTrapNumber selector, ...)</code>                                                                                                                                                                           |
| <b>Parameters</b>  | <code>→ selector</code><br>The function selector of for the function to be called. One of the values listed under “ <a href="#">Host Control Function Selectors</a> ” on page 318.<br><code>→ ...</code><br>Parameters for the function being called. |
| <b>Returns</b>     | Returns the result of the function.                                                                                                                                                                                                                   |

## HostCTime Function

|                    |                                                                                  |
|--------------------|----------------------------------------------------------------------------------|
| <b>Purpose</b>     | Generate a text representation of a <a href="#">HostTimeType</a> structure.      |
| <b>Declared In</b> | HostControl.h                                                                    |
| <b>Prototype</b>   | <code>char *HostCTime (const HostTimeType *timeOfDay)</code>                     |
| <b>Parameters</b>  | <code>→ timeOfDay</code><br>The time for which a text representation is desired. |
| <b>Returns</b>     | A text representation of the specified time.                                     |
| <b>See Also</b>    | <a href="#">HostAscTime()</a>                                                    |

## HostEnteringApp Function

- Purpose** Record an application's name for later use in various logging messages.
- Declared In** `HostControl.h`
- Prototype** `void HostEnteringApp (char *appName)`
- Parameters** → *appName*  
The name of the application that is about to be entered.
- Returns** Returns nothing.
- See Also** [HostSetErrorLevel\(\)](#)

## HostErrNo Function

- Purpose** Returns the value of `errno`, the standard C library variable that reflects the result of many standard C library functions. You can call this function after calling one of the host control functions that wraps the standard C library.
- Declared In** `HostControl.h`
- Prototype** `long HostErrNo (void)`
- Parameters** None.
- Returns** The error number.
- Comments** This function is only applicable to functions that wrap standard C library functions that affect `errno`. It is not applicable to all host control functions.
- See Also** [HostFError\(\)](#)

## HostExgLibAccept Function

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibAccept (uint16_t libRefNum,  
void *exgSocketP)`

## HostExgLibClose Function

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibClose (uint16_t libRefNum)`

## HostExgLibConnect Function

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibConnect (uint16_t libRefNum,  
void *exgSocketP)`

## HostExgLibControl Function

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibControl (uint16_t libRefNum,  
uint16_t op, void *valueP,  
uint16_t *valueLenP)`

## HostExgLibDisconnect Function

**Purpose** Internal function intended for system use only. Applications should

## Host Control

### *HostExgLibGet*

---

not call this function.

**Declared In** HostControl.h

**Prototype** `status_t HostExgLibDisconnect  
(uint16_t libRefNum, void *exgSocketP,  
status_t error)`

### **HostExgLibGet Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** HostControl.h

**Prototype** `status_t HostExgLibGet (uint16_t libRefNum,  
void *exgSocketP)`

### **HostExgLibHandleEvent Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** HostControl.h

**Prototype** `Boolean HostExgLibHandleEvent  
(uint16_t libRefNum, void *eventP)`

### **HostExgLibOpen Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** HostControl.h

**Prototype** `status_t HostExgLibOpen (uint16_t libRefNum)`

### **HostExgLibPut Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** HostControl.h

**Prototype** `status_t HostExgLibPut (uint16_t libRefNum,  
void *exgSocketP)`

### **HostExgLibReceive Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `uint32_t HostExgLibReceive (uint16_t libRefNum,  
void *exgSocketP, void *bufP,  
const uint32_t bufSize, status_t *errP)`

### **HostExgLibRequest Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibRequest (uint16_t libRefNum,  
void *exgSocketP)`

### **HostExgLibSend Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `uint32_t HostExgLibSend (uint16_t libRefNum,  
void *exgSocketP, const void *const bufP,  
const uint32_t bufLen, status_t *errP)`

### **HostExgLibSleep Function**

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

## Host Control

### *HostExgLibWake*

---

**Prototype** `status_t HostExgLibSleep (uint16_t libRefNum)`

### HostExgLibWake Function

**Purpose** Internal function intended for system use only. Applications should not call this function.

**Declared In** `HostControl.h`

**Prototype** `status_t HostExgLibWake (uint16_t libRefNum)`

### HostExitedApp Function

**Purpose**

**Declared In** `HostControl.h`

**Prototype** `void HostExitedApp (char *, uint32_t)`

**Parameters**

`→ uint32_t`

**Returns** Nothing.

### HostExportFile Function

**Purpose** Copies a database from the device to the desktop computer.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostExportFile (const char *fileName,  
const char *dbName)`

**Parameters** `→ fileName`

The filename to use on the desktop computer.

`→ dbName`

The name of the database on the device to be copied.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**See Also** [HostImportFile\(\)](#)

## HostFClose Function

**Purpose** Close a file on the desktop computer.

**Declared In** `HostControl.h`

**Prototype** `long HostFClose (HostFileType *f)`

**Parameters** `→ f`  
The file to close.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**See Also** [HostFOpen\(\)](#)

## HostFEOF Function

**Purpose** Determine the status of a specified file’s end-of-file indicator.

**Declared In** `HostControl.h`

**Prototype** `long HostFEOF (HostFileType *f)`

**Parameters** `→ f`  
The file for which the end-of-file status is being checked.

**Returns** Returns 0 if the specified file’s end-of-file indicator is set, or a nonzero value otherwise.

### HostFError Function

- Purpose** Retrieve the error code from the most recent operation on the specified file.
- Declared In** `HostControl.h`
- Prototype** `long HostFError (HostFileType *f)`
- Parameters** → *f*  
The file for which the error code is needed.
- Returns** The error code from the most recent operation on the specified file. Returns `errNone` if no errors have occurred on the file.
- See Also** [HostErrNo\(\)](#)

### HostFFlush Function

- Purpose** Flush the buffer for the specified file.
- Declared In** `HostControl.h`
- Prototype** `long HostFFlush (HostFileType *f)`
- Parameters** → *f*  
The file to flush.
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.
- See Also** [HostFPrintf\(\)](#), [HostFPutC\(\)](#), [HostFPutS\(\)](#), [HostFWrite\(\)](#)

### HostFGetC Function

- Purpose** Retrieve the character at the current position in the specified file.
- Declared In** `HostControl.h`
- Prototype** `long HostFGetC (HostFileType *f)`
- Parameters** → *f*  
The file from which the character is to be read.

**Returns** The character at the current position, or EOF if the current position is at the end of the file.

**See Also** [HostFGetS\(\)](#), [HostFPutC\(\)](#)

## HostFGetPos Function

**Purpose** Retrieve the current position in the specified file.

**Declared In** `HostControl.h`

**Prototype** `long HostFGetPos (HostFileType *f, long *posP)`

**Parameters**  $\rightarrow f$

The file.

$\leftarrow posP$

The current position in the file.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**See Also** [HostFTell\(\)](#)

## HostFGetS Function

**Purpose** Retrieve a character string from the selected file.

**Declared In** `HostControl.h`

**Prototype** `char *HostFGetS (char *s, long n, HostFileType *f)`

**Parameters**  $\leftrightarrow s$

Pointer to the string buffer that will be filled with characters obtained from the specified file.

$\rightarrow n$

The number of characters to retrieve.

$\rightarrow f$

The file containing the string to be retrieved.

**Returns** A pointer to the character string, or `NULL` if an error occurred.

**See Also** [HostFError\(\)](#), [HostFGetC\(\)](#)

## HostFOpen Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Open a file on the desktop computer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Declared In</b> | <code>HostControl.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Prototype</b>   | <code>HostFileType *HostFOpen (const char *name,<br/>const char *mode)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Parameters</b>  | <p>→ <i>name</i><br/>The name of the file to open.</p> <p>→ <i>mode</i><br/>The mode to use when opening the file. See the Comments section, below, for the format of this string.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Returns</b>     | A filestream pointer or NULL if an error occurred.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Comments</b>    | <p>The <i>mode</i> argument points to a string beginning with one of the following sequences (additional characters may follow these sequences):</p> <p>"r"<br/>Open text file for reading. The stream is positioned at the beginning of the file.</p> <p>"r+"<br/>Open for reading and writing. The stream is positioned at the beginning of the file.</p> <p>"w"<br/>Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.</p> <p>"w+"<br/>Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.</p> <p>"a"<br/>Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file. Subsequent writes to the file will always end up at the then current end of file, irrespective of any intervening <a href="#">HostFSeek()</a> or similar.</p> <p>"a+"<br/>Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file. Subsequent writes to the file will always end up at the then</p> |

current end of file, irrespective of any intervening [HostFSeek\(\)](#) or similar.

The mode string can also include the letter "f" to indicate that only plain files should be opened. It can also include the letter "b", but this is strictly for compatibility with ISO/IEC 9899:1990 ("ISO C89") and has no effect; the "b" is ignored.

**See Also** [HostFClose\(\)](#), [HostFError\(\)](#), [HostFReopen\(\)](#)

## HostFPrintf Function

**Purpose** Write a formatted string to a file.

**Declared In** `HostControl.h`

**Prototype** `long HostFPrintf (HostFILEType *f,  
                  const char *fmt, ...)`

**Parameters**

- *f*  
The file to which the string is written.
- *fmt*  
The format specification.
- ...  
String arguments.

**Returns** The number of characters written to the file.

**See Also** [HostFPutC\(\)](#), [HostFPutS\(\)](#), [HostFScanF\(\)](#)

## HostFPutC Function

**Purpose** Write a character to the specified file.

**Declared In** `HostControl.h`

**Prototype** `long HostFPutC (long c, HostFILEType *f)`

**Parameters**

- *c*  
The character to write.
- *f*  
The file to which the character is written.

## Host Control

### *HostFPutS*

---

**Returns** The number of characters written, or EOF if an error occurred.

**See Also** [HostFError\(\)](#), [HostFGetC\(\)](#), [HostFPutS\(\)](#)

## HostFPutS Function

**Purpose** Write a string to the specified file.

**Declared In** `HostControl.h`

**Prototype** `long HostFPutS (const char *s, HostFileType *f)`

**Parameters** → *s*

The string to write.

→ *f*

The file to which the string is written.

**Returns** The number of characters written, or EOF if an error occurred.

**See Also** [HostFError\(\)](#), [HostFGetS\(\)](#), [HostFPrintf\(\)](#), [HostFPutC\(\)](#)

## HostFRead Function

**Purpose** Read a number of items from a file into a buffer.

**Declared In** `HostControl.h`

**Prototype** `long HostFRead (void *buffer, long size, long count, HostFileType *f)`

**Parameters** ↔ *buffer*

The buffer into which data is read.

→ *size*

The size of each item.

→ *count*

The number of items to read.

→ *f*

The file from which to read.

**Returns** The number of items that were actually read.

**See Also** [HostFScanF\(\)](#), [HostFWrite\(\)](#)

## HostFree Function

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <b>Purpose</b>     | Free memory on the desktop computer.                         |
| <b>Declared In</b> | HostControl.h                                                |
| <b>Prototype</b>   | void HostFree (void *p)                                      |
| <b>Parameters</b>  | → <i>p</i><br>Pointer to the memory block to be freed.       |
| <b>Returns</b>     | Returns nothing.                                             |
| <b>See Also</b>    | <a href="#">HostMalloc()</a> , <a href="#">HostRealloc()</a> |

## HostFReopen Function

|                    |                                                                                                                                                                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Change the file with which a given file stream is associated. HostFReopen closes the file currently associated with the stream, then opens the new file and associates it with the same stream.                                                         |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                           |
| <b>Prototype</b>   | HostFileType *HostFReopen (const char *name,<br>const char *mode, HostFileType *f)                                                                                                                                                                      |
| <b>Parameters</b>  | → <i>name</i><br>The name of the file to open.<br>→ <i>mode</i><br>The mode to use when opening the file. See the Comments section under <a href="#">HostFOpen()</a> for the format of this parameter.<br>→ <i>f</i><br>The file stream to be reopened. |
| <b>Returns</b>     | The file stream pointer, or NULL to indicate an error.                                                                                                                                                                                                  |
| <b>See Also</b>    | <a href="#">HostFClose()</a> , <a href="#">HostFError()</a> , <a href="#">HostFOpen()</a>                                                                                                                                                               |

### HostFScanF Function

|                    |                                                                                                                                                                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Read formatted text from a file.                                                                                                                                                                                                        |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                           |
| <b>Prototype</b>   | <pre>long HostFScanF (HostFILEType *f,<br/>                const char *fmt, ...)</pre>                                                                                                                                                  |
| <b>Parameters</b>  | <p>→ <i>f</i><br/>The file from which to read.</p> <p>→ <i>fmt</i><br/>A format string, as used in standard C-library calls such as <code>scanf</code>.</p> <p>← ...<br/>The list of variables into which scanned input is written.</p> |
| <b>Returns</b>     | The number of items that were read, or a negative value to indicate an error. Returns EOF if end of file was reached while scanning.                                                                                                    |
| <b>See Also</b>    | <a href="#">HostFGetS()</a> , <a href="#">HostFPrintf()</a>                                                                                                                                                                             |

### HostFSeek Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Move the file pointer to a specified position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Prototype</b>   | <pre>long HostFSeek (HostFILEType *f, long offset,<br/>               long origin)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Parameters</b>  | <p>→ <i>f</i><br/>The file for which the current position is to be moved.</p> <p>→ <i>offset</i><br/>The number of bytes to move from the initial position, which is specified in the <i>origin</i> parameter. Note that this is a signed value; <i>offset</i> can either be positive or negative.</p> <p>→ <i>origin</i><br/>The initial position. Either <code>SEEK_SET</code> (<i>offset</i> is relative to the start of the file), <code>SEEK_CUR</code> (<i>offset</i> is relative to the current position), or <code>SEEK_END</code> (<i>offset</i> is relative to the end of the file).</p> |

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**See Also** [HostFGetPos\(\)](#), [HostFSetPos\(\)](#), [HostFTell\(\)](#)

## HostFSetPos Function

**Purpose** Set the file’s position indicator.

**Declared In** `HostControl.h`

**Prototype** `long HostFSetPos (HostFileType *f, long *pos)`

**Parameters** → *f*

The file for which the position indicator is to be set.

→ *pos*

The position within the file.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**See Also** [HostFGetPos\(\)](#), [HostFSeek\(\)](#), [HostFTell\(\)](#)

## HostFTell Function

**Purpose** Retrieve the current position of the specified file.

**Declared In** `HostControl.h`

**Prototype** `long HostFTell (HostFileType *f)`

**Parameters** → *f*

The file.

**Returns** Returns `errNone` if the operation was successful, or -1 if an error occurred.

**See Also** [HostFError\(\)](#), [HostFGetPos\(\)](#), [HostFSeek\(\)](#), [HostFSetPos\(\)](#)

### HostFWrite Function

- Purpose** Write data to a file.
- Declared In** `HostControl.h`
- Prototype** `long HostFWrite (const void *buffer, long size, long count, HostFileType *f)`
- Parameters**
- *buffer*  
The buffer that contains the data to be written.
  - *size*  
The size of each item.
  - *count*  
The number of items to write.
  - *f*  
The file to which the data is written.
- Returns** The number of items actually written.
- See Also** [HostFPutC\(\)](#), [HostFPutS\(\)](#), [HostFRead\(\)](#)

### HostGestalt Function

- Purpose** Currently does nothing except return an “invalid selector” error. In the future, this function will be used for queries about the runtime environment.
- Declared In** `HostControl.h`
- Prototype** `HostErrType HostGestalt (long gestSel, long *response)`
- Parameters**
- *gestSel*  
A selector.
  - ← *response*  
The response. This function currently does nothing with this parameter.
- Returns** Returns `hostErrUnknownGestaltSelector`.

## HostGetChar Function

|                    |                                     |
|--------------------|-------------------------------------|
| <b>Purpose</b>     | Get a character from the host user. |
| <b>Declared In</b> | HostControl.h                       |
| <b>Prototype</b>   | long HostGetChar (void)             |
| <b>Parameters</b>  | None.                               |
| <b>Returns</b>     | The character.                      |

## HostGetDirectory Function

|                    |                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Get a directory, in support of the operating system file chooser dialog box.                                   |
| <b>Declared In</b> | HostControl.h                                                                                                  |
| <b>Prototype</b>   | const char *HostGetDirectory (const char *prompt,<br>const char *defaultDir)                                   |
| <b>Parameters</b>  | → <i>prompt</i><br>Text with which to prompt the user.<br>→ <i>defaultDir</i><br>The default directory to get. |
| <b>Returns</b>     | Returns the directory as a character string.                                                                   |
| <b>See Also</b>    | <a href="#">HostGetFile()</a>                                                                                  |

## HostGetEnv Function

|                    |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Retrieve the value of an environment variable.                                            |
| <b>Declared In</b> | HostControl.h                                                                             |
| <b>Prototype</b>   | char *HostGetEnv (const char *varName)                                                    |
| <b>Parameters</b>  | → <i>varName</i><br>The name of the environment variable that you want to retrieve.       |
| <b>Returns</b>     | The value of the named variable as a string, or NULL if the variable cannot be retrieved. |
| <b>See Also</b>    | <a href="#">HostGetPreference()</a>                                                       |

## HostGetFile Function

- Purpose** Get a file, in support of the operating system file chooser dialog box.
- Declared In** `HostControl.h`
- Prototype**  
`const char *HostGetFile (const char *prompt,  
const char *defaultDir)`
- Parameters**
- *prompt*  
Text with which to prompt the user.
  - *defaultDir*  
The default directory to get.
- Returns** Returns the file as a character string.
- See Also** [HostGetDirectory\(\)](#), [HostGetFileAttr\(\)](#), [HostPutFile\(\)](#)

## HostGetFileAttr Function

- Purpose** Get the attribute settings of a file or directory. This function can tell you whether the file is read-only, hidden, or a system file.
- Declared In** `HostControl.h`
- Prototype**  
`long HostGetFileAttr (const char *fileOrPathName,  
long *attrFlag)`
- Parameters**
- *fileOrPathname*  
The file name or directory path for which you want to get the file attribute setting.
  - *attrFlag*  
The setting to be retrieved. Supply one of the values listed under “[File Attributes](#)” on page 327.
- Returns** The value of the requested file attribute.
- See Also** [HostGetFile\(\)](#), [HostSetFileAttr\(\)](#)

## HostGetFirstApp Function

**Purpose**  
**Declared In** `HostControl.h`  
**Prototype** `uint32_t HostGetFirstApp (void)`  
**Parameters** None.  
**Returns**

## HostGetHostID Function

**Purpose** Retrieve the ID of the debugging host. This is one of the constants described in “[Host IDs](#)” on page 325. Palm OS Emulator always returns the value `hostIDPalmOSEmulator`.  
**Declared In** `HostControl.h`  
**Prototype** `HostIDType HostGetHostID (void)`  
**Parameters** None.  
**Returns** The host ID.  
**See Also** [HostGetHostPlatform\(\)](#)

## HostGetHostPlatform Function

**Purpose** Retrieve the host platform ID, which is one of the values described in “[Host Platforms](#)” on page 326..  
**Declared In** `HostControl.h`  
**Prototype** `HostPlatformType HostGetHostPlatform (void)`  
**Parameters** None.  
**Returns** The platform ID.  
**See Also** [HostGetHostID\(\)](#)

## HostGetHostVersion Function

- Purpose** Retrieve the version number of the debugging host.
- Declared In** `HostControl.h`
- Prototype** `long HostGetHostVersion (void)`
- Parameters** None.
- Returns** The version number.
- Comments** This function returns the version number in the same format that is used by the Palm OS, which means that you can access the version number components using the following macros from the `SystemMgr.h` file:
- `sysGetROMVerMajor(dwROMVer)`
  - `sysGetROMVerMinor(dwROMVer)`
  - `sysGetROMVerFix(dwROMVer)`
  - `sysGetROMVerStage(dwROMVer)`
  - `sysGetROMVerBuild(dwROMVer)`

## HostGetPreference Function

- Purpose** Retrieve a specified preference value.
- Declared In** `HostControl.h`
- Prototype** `HostBoolType HostGetPreference  
(const char *prefName, char *prefValue)`
- Parameters**  $\rightarrow$  `prefName`  
The name of the preference whose value you want to retrieve.
- $\leftarrow$  `prefValue`  
The value of the specified preference, as a string.
- Returns** If the specified preference was successfully retrieved, this function returns `true`, and `*prefValue` is set to the value of the preference. Otherwise, this function returns `false` and `*prefValue` isn't changed.
- Comments** Each preference is identified by name. The preference names can be found in the appropriate Palm OS Emulator preferences file,

depending on your platform. See [Table 29.1](#) for the name and location of the Palm OS Emulator preference file for each supported platform.

**Table 29.1 Palm OS Emulator preferences file names and locations**

| <b>Platform</b> | <b>File name</b>       | <b>File location</b>         |
|-----------------|------------------------|------------------------------|
| Macintosh       | Palm OS Emulator Prefs | The Preferences folder       |
| Windows         | Palm OS Emulator.ini   | The Windows System directory |
| Unix            | .poserrc               | Your home directory          |

**See Also** [HostGetEnv\(\)](#), [HostSetPreference\(\)](#)

## HostGMTime Function

**Purpose** Get a time structure representation of a time value, expressed as Universal Time Coordinated, or UTC (UTC was formerly Greenwich Mean Time, or GMT).

**Declared In** `HostControl.h`

**Prototype** `HostTmType *HostGMTime (const HostTimeType *time)`

**Parameters** `→ time`

The [HostTimeType](#) value.

**Returns** The specified time, as a [HostTmType](#) structure.

**See Also** [HostLocalTime\(\)](#), [HostMkTime\(\)](#)

## HostHostControl68K Function

**Purpose**

**Declared In** HostControl.h

**Prototype** `uint32_t HostHostControl68K  
(HostControlSelectorType selector,  
void *pceEmulState)`

**Parameters** → *selector*

*pceEmulState*

**Returns**

## HostImportFile Function

**Purpose** Copy a database from the desktop computer to the device.

**Declared In** HostControl.h

**Prototype** `HostErrType HostImportFile (const char *fileName)`

**Parameters** → *fileName*  
The file on the desktop computer that contains the database being copied.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**Comments** The database name on the device is the name stored in the file.

**See Also** [HostExportFile\(\)](#)

## HostIsCallingTrap Function

**Purpose** Determine if Palm OS Emulator is currently calling a trap.

**Declared In** HostControl.h

**Prototype** `HostBoolType HostIsCallingTrap (void)`

**Parameters** None.

**Returns** A Boolean value indicating whether or not Palm OS Emulator is currently calling a trap.

## HostIsSelectorImplemented Function

**Purpose** Determine if a specified function selector is implemented on the debugging host.

**Declared In** `HostControl.h`

**Prototype** `HostBoolType HostIsSelectorImplemented  
(long selector)`

**Parameters** → *selector*  
The function selector. This must be one of the constants described in “[Host Control Function Selectors](#)” on page 318.

**Returns** A Boolean value indicating whether or not the specified function selector is implemented on the debugging host.

## HostLocalTime Function

**Purpose** Get a time structure representation of a time value, expressed as local time.

**Declared In** `HostControl.h`

**Prototype** `HostTmType *HostLocalTime  
(const HostTimeType *time)`

**Parameters** → *time*  
The [HostTimeType](#) value.

**Returns** The specified time, as a [HostTmType](#) structure.

**See Also** [HostGMTime\(\)](#)

### HostLogEvent Function

|                    |                            |
|--------------------|----------------------------|
| <b>Purpose</b>     |                            |
| <b>Declared In</b> | HostControl.h              |
| <b>Prototype</b>   | void HostLogEvent (void *) |
| <b>Parameters</b>  |                            |
| <b>Returns</b>     | Nothing.                   |

### HostLogFile Function

|                    |                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Get a reference to the file that the Emulator is using to log information. You can use this reference to add your own information to the log file. |
| <b>Declared In</b> | HostControl.h                                                                                                                                      |
| <b>Prototype</b>   | HostFileType *HostLogFile (void)                                                                                                                   |
| <b>Parameters</b>  | None.                                                                                                                                              |
| <b>Returns</b>     | A pointer to the log file, or NULL if the function was unable to obtain a reference to the log file.                                               |

### HostMalloc Function

|                    |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Allocate a block of memory on the debugging host.                                         |
| <b>Declared In</b> | HostControl.h                                                                             |
| <b>Prototype</b>   | void *HostMalloc (long <i>size</i> )                                                      |
| <b>Parameters</b>  | → <i>size</i><br>The number of bytes to allocate.                                         |
| <b>Returns</b>     | A pointer to the allocated memory block, or NULL if there is not enough memory available. |
| <b>See Also</b>    | <a href="#">HostFree()</a> , <a href="#">HostRealloc()</a>                                |

## HostMkDir Function

- Purpose** Create a directory on the debugging host.
- Declared In** `HostControl.h`
- Prototype** `long HostMkDir (const char *directory)`
- Parameters** `→ directory`  
The directory to create.
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.
- See Also** [HostFOpen\(\)](#), [HostRmdir\(\)](#)

## HostMkTime Function

- Purpose** Convert a time structure representation of a time value to an equivalent encoded local time.
- Declared In** `HostControl.h`
- Prototype** `HostTimeType HostMkTime (HostTmType *tm)`
- Parameters** `→ tm`  
The [HostTmType](#) structure containing the time to be converted.
- Returns** Returns the calendar time equivalent to the encoded time, or returns a value of -1 if the calendar time cannot be represented.
- See Also** [HostGMTime\(\)](#)

## HostOpenDir Function

- Purpose** Open a directory.
- Declared In** `HostControl.h`
- Prototype** `HostDIRType *HostOpenDir (const char *directory)`
- Parameters** `→ directory`  
The directory to open.

## Host Control

### *HostPrintF*

---

**Returns** A directory structure that can be used with other directory operations.

**See Also** [HostCloseDir\(\)](#)

## HostPrintF Function

**Purpose** Send formatted output to stdout.

**Declared In** `HostControl.h`

**Prototype** `long HostPrintF (const char *fmt, ...)`

**Parameters** → *fmt*

The format string. See the man page for the standard C function `printf()` for instructions on creating this argument.

→ ...

Arguments to be sent to stdout, formatted according to *fmt*.

**Returns** Returns the number of characters printed, not including the trailing null character used to end output to strings.

**See Also** [HostVFPrintF\(\)](#), [HostVPrintF\(\)](#)

## HostProfileCleanup Function

**Purpose** Release the memory used for profiling and disable profiling.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostProfileCleanup (void)`

**Parameters** None.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred. Returns `hostErrProfilingNotReady` if called out of sequence. For information on profiling sequence, see [HostProfileInit\(\)](#).

**Comments** This function is available only in the profiling version of the Emulator.

**See Also** [HostProfileDump\(\)](#), [HostProfileStart\(\)](#), [HostProfileStop\(\)](#)

## HostProfileDetailFn Function

**Purpose** Profile the function that contains a specified address.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostProfileDetailFn (void *addr, HostBoolType logDetails)`

**Parameters**

- *addr*  
The address in which you are interested.
- *logDetails*  
If true, profiling is performed at a machine-language instruction level, which means that each opcode is treated as its own function.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**Comments** This function is available only in the profiling version of the Emulator.

**See Also** [HostProfileCleanup\(\)](#), [HostProfileDump\(\)](#), [HostProfileStart\(\)](#), [HostProfileStop\(\)](#)

## HostProfileDump Function

**Purpose** Write the current profiling information to a named file.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostProfileDump (const char *filename)`

**Parameters**

- *filename*  
The name of the file to which the profile information gets written.

## Host Control

### *HostProfileInit*

---

- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred. Returns `hostErrProfilingNotReady` if called out of sequence. For information on profiling sequence, see [HostProfileInit\(\)](#).
- Comments** This function is available only in the profiling version of the Emulator.
- See Also** [HostProfileCleanup\(\)](#), [HostProfileInit\(\)](#), [HostProfileStart\(\)](#), [HostProfileStop\(\)](#)

## HostProfileInit Function

- Purpose** Initialize and enable profiling in the debugging host.
- Declared In** `HostControl.h`
- Prototype** `HostErrType HostProfileInit (long maxCalls, long maxDepth)`
- Parameters**
- *maxCalls*  
The maximum number of calls to profile. This parameter determines the size of the array used to keep track of function calls. A typical value for `maxCalls` is 65536.
  - *maxDepth*  
The maximum profiling depth. This parameter determines the size of the array used to keep track of function call depth. A typical value for `maxDepth` is 200.
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred. Returns `hostErrProfilingNotReady` if called out of sequence.
- Comments** This function is available only in the profiling version of the Emulator.
- The host control profiling functions are intended to be called in sequence:
1. `HostProfileInit()` - All profiling starts with this function, which initializes and enables profiling.
  2. [HostProfileStart\(\)](#) turns profiling on.

3. [HostProfileStop\(\)](#) turns profiling off. After calling `HostProfileStop()`, you can either call `HostProfileStart()` to restart profiling or call `HostProfileDump()`, which disables profiling and writes data to a file.
4. [HostProfileDump\(\)](#) disables profiling and writes data to a file. If you need to do more profiling after calling `HostProfileDump()`, you must call `HostProfileInit()` to re-enable profiling.
5. [HostProfileCleanup\(\)](#) releases the memory used for profiling and disables profiling.

## HostProfileStart Function

|                    |                                                                                                                                                                                                                                                                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Turn profiling on.                                                                                                                                                                                                                                                                                                                        |
| <b>Declared In</b> | <code>HostControl.h</code>                                                                                                                                                                                                                                                                                                                |
| <b>Prototype</b>   | <code>HostErrType HostProfileStart (void)</code>                                                                                                                                                                                                                                                                                          |
| <b>Parameters</b>  | None.                                                                                                                                                                                                                                                                                                                                     |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred. Returns <code>hostErrProfilingNotReady</code> if called out of sequence. For information on profiling sequence, see <a href="#">HostProfileInit()</a> . |
| <b>Comments</b>    | This function is available only in the profiling version of the Emulator.                                                                                                                                                                                                                                                                 |
| <b>See Also</b>    | <a href="#">HostProfileCleanup()</a> , <a href="#">HostProfileDump()</a> , <a href="#">HostProfileInit()</a> , <a href="#">HostProfileStop()</a>                                                                                                                                                                                          |

## HostProfileStop Function

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <b>Purpose</b>     | Turn profiling off.                             |
| <b>Declared In</b> | <code>HostControl.h</code>                      |
| <b>Prototype</b>   | <code>HostErrType HostProfileStop (void)</code> |
| <b>Parameters</b>  | None.                                           |

## Host Control

### HostPutFile

---

- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred. Returns `hostErrProfilingNotReady` if called out of sequence. For information on profiling sequence, see [HostProfileInit\(\)](#).
- Comments** This function is available only in the profiling version of the Emulator.
- See Also** [HostProfileCleanup\(\)](#), [HostProfileDump\(\)](#), [HostProfileInit\(\)](#), [HostProfileStart\(\)](#)

## HostPutFile Function

- Purpose** Write a file, in support of the operating system Save As dialog box.
- Declared In** `HostControl.h`
- Prototype**  
`const char *HostPutFile (const char *prompt,  
                          const char *defaultDir,  
                          const char *defaultName)`
- Parameters**
- *prompt*  
Text with which to prompt the user.
  - *defaultDir*  
The default directory to use.
  - *defaultName*  
The default file name to use.
- Returns** Returns the file name as a character string.
- See Also** [HostGetFile\(\)](#)

## HostReadDir Function

- Purpose** Read a directory.
- Declared In** `HostControl.h`
- Prototype**  
`HostDirEntType *HostReadDir  
                  (HostDIRType *directory)`
- Parameters**
- *directory*  
The directory to read.

**Returns** Returns a character array for the directory.

## HostRealloc Function

**Purpose** Reallocate space for a specified memory block.

**Declared In** `HostControl.h`

**Prototype** `void *HostRealloc (void *p, long size)`

**Parameters** → *p*

A pointer to a memory block to be resized.

→ *size*

The new size for the memory block.

**Returns** A pointer to the reallocated memory block, or NULL if the memory block couldn't be reallocated.

**Comments** Reallocation of the memory block can cause it to be relocated in memory. Thus you must use the pointer returned from this function to access the block after reallocation.

**See Also** [HostFree\(\)](#), [HostMalloc\(\)](#)

## HostRemove Function

**Purpose** Delete a file.

**Declared In** `HostControl.h`

**Prototype** `long HostRemove (const char *name)`

**Parameters** → *name*

The name of the file to be deleted.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in "[Host Control Error Codes](#)" on page 323 if an error occurred.

**See Also** [HostRmdir\(\)](#)

## Host Control

### *HostRename*

---

### HostRename Function

- Purpose** Rename a file.
- Declared In** `HostControl.h`
- Prototype** `long HostRename (const char *oldName,  
const char *newName)`
- Parameters**
- *oldName*  
The name of the file to be renamed.
  - *newName*  
The new name of the file.
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

### HostRmDir Function

- Purpose** Removes a directory.
- Declared In** `HostControl.h`
- Prototype** `long HostRmDir (const char *directory)`
- Parameters**
- *directory*  
The directory to remove.
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.
- See Also** [HostRemove\(\)](#)

## HostSessionClose Function

|                      |                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>       | Close the current emulation session.                                                                                                                                              |
| <b>Declared In</b>   | HostControl.h                                                                                                                                                                     |
| <b>Prototype</b>     | <pre>HostErrType HostSessionClose<br/>    (const char *saveFileName)</pre>                                                                                                        |
| <b>Parameters</b>    | <p>→ <i>saveFileName</i><br/>The name of the file to which the current session is to be saved.</p>                                                                                |
| <b>Returns</b>       | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred. |
| <b>Compatibility</b> | This function is defined for external RPC clients to call; the effect of calling it for Palm OS applications running on the emulated device is undefined.                         |
| <b>See Also</b>      | <a href="#">HostSessionCreate()</a>                                                                                                                                               |

## HostSessionCreate Function

|                    |                                                                                                                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Create a new emulation session.                                                                                                                                                                                                   |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                     |
| <b>Prototype</b>   | <pre>HostErrType HostSessionCreate<br/>    (const char *device, long ramSize,<br/>    const char *romPath)</pre>                                                                                                                  |
| <b>Parameters</b>  | <p>→ <i>device</i><br/>The name of the device to emulate in the session.</p> <p>→ <i>ramSize</i><br/>The amount of emulated RAM in the new session.</p> <p>→ <i>romPath</i><br/>The path to the ROM file for the new session.</p> |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred.                                                 |

**Compatibility** This function is defined for external RPC clients to call; the effect of calling it for Palm OS applications running on the emulated device is undefined.

**See Also** [HostSessionClose\(\)](#)

## HostSessionOpen Function

**Purpose** Open a previously saved emulation session.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostSessionOpen  
(const char *psfFileName)`

**Parameters** `→ psfFileName`  
The name of the file containing the saved session to open.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**Compatibility** This function is defined for external RPC clients to call; the effect of calling it for Palm OS applications running on the emulated device is undefined.

## HostSessionQuit Function

**Purpose** Ask Palm OS Emulator to quit. Return an error if a session is already running.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostSessionQuit (void)`

**Parameters** None.

**Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**Compatibility** This function is defined for external RPC clients to call; the effect of calling it for Palm OS applications running on the emulated device is undefined.

**See Also** [HostSessionCreate\(\)](#)

## HostSetErrorLevel Function

- Purpose** Change the error level, and write a message to the error log.
- Declared In** `HostControl.h`
- Prototype** `void HostSetErrorLevel (uint32_t level,  
const char *msg)`
- Parameters**
- *level*  
The new error level. One of the values listed under “[Error Levels](#)” on page 327..
  - *msg*  
A message to be recorded in the log file (`ErrorLog.txt`).
- Returns** Returns nothing.

## HostSetFileAttr Function

- Purpose** Set the read-only, hidden, or system-file attribute for a file or directory.
- Declared In** `HostControl.h`
- Prototype** `long HostSetFileAttr (const char *fileOrPathName,  
long attrFlag)`
- Parameters**
- *fileOrPathName*  
The file name or directory path for which you want to set the attribute.
  - *long*  
The attribute to be set. Supply one of the values listed under “[File Attributes](#)” on page 327.
- Returns** Returns the file attribute.
- See Also** [HostGetFileAttr\(\)](#)

## HostSetLogFileSize Function

- Purpose** Specify the size of the logging file that Palm OS Emulator is to use.
- Declared In** `HostControl.h`
- Prototype** `void HostSetLogFileSize (long size)`
- Parameters** `→ size`  
The new size for the logging file, in bytes.
- Returns** Returns nothing.
- Comments** By default, Palm OS Emulator saves the last 1 MB of log data to prevent logging files from becoming enormous. You can call this function to change the log file size.

## HostSetPreference Function

- Purpose** Set a specified preference value.
- Declared In** `HostControl.h`
- Prototype** `void HostSetPreference (const char *prefName,  
const char *prefValue)`
- Parameters** `→ prefName`  
The name of the preference whose value is to be set.  
`→ prefValue`  
The new value of the preference, as a string.
- Returns** Returns nothing.
- Comments** Each preference is identified by name. The preference names can be found in the appropriate Palm OS Emulator preferences file, depending on your platform. See [Table 29.1](#) on page 349 for the name and location of the Palm OS Emulator preference file for each supported platform.
- See Also** [HostGetPreference\(\)](#)

## HostSignalResume Function

|                      |                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>       | Restart Palm OS Emulator after it has issued a signal.                                                                                                                            |
| <b>Declared In</b>   | HostControl.h                                                                                                                                                                     |
| <b>Prototype</b>     | HostErrType HostSignalResume (void)                                                                                                                                               |
| <b>Parameters</b>    | None.                                                                                                                                                                             |
| <b>Returns</b>       | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred. |
| <b>Comments</b>      | Palm OS Emulator waits to be restarted after issuing a signal to allow external scripts to perform operations.                                                                    |
| <b>Compatibility</b> | This function is defined for external RPC clients to call and returns an error if you call it from within a Palm OS application.                                                  |
| <b>See Also</b>      | <a href="#">HostSignalSend()</a> , <a href="#">HostSignalWait()</a>                                                                                                               |

## HostSignalSend Function

|                    |                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Send a signal to any scripts with pending <a href="#">HostSignalWait()</a> calls.                                                                                                                                                                                                                                                   |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                                                                                                       |
| <b>Prototype</b>   | HostErrType HostSignalSend<br>(HostSignalType <i>signalNumber</i> )                                                                                                                                                                                                                                                                 |
| <b>Parameters</b>  | → <i>signalNumber</i><br>The signal for which you want to wait. This can be a predefined signal or one that you have defined.                                                                                                                                                                                                       |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation was successful, or one of the error values listed in “ <a href="#">Host Control Error Codes</a> ” on page 323 if an error occurred. Returns <code>hostErrNoSignalWaiters</code> if there aren’t any scripts waiting for a signal.                                                     |
| <b>Comments</b>    | Palm OS Emulator halts and waits to be restarted after sending the signal. This allows external scripts to perform operations. The external script must call the <a href="#">HostSignalResume()</a> function to restart Palm OS Emulator.<br><br>If there are not any scripts waiting for a signal, Palm OS Emulator does not halt. |

## Host Control

### *HostSignalWait*

---

The predefined signals are documented under “[Host Signals](#)” on page 326.

**Compatibility** This function is defined for external RPC clients to call and returns an error if you call it from within a Palm OS application.

## HostSignalWait Function

**Purpose** Wait for a signal from Palm OS Emulator, and return the signalled value.

**Declared In** `HostControl.h`

**Prototype** `HostErrType HostSignalWait (long timeout,  
HostSignalType *signalNumber)`

**Parameters** → *timeout*

The number of milliseconds to wait for the signal before timing out.

← *signalNumber*

The number of the signal that occurred.

**Returns** Returns `errNone` if the operation was successful, `hostErrTimeout` if the amount of time specified in *timeout* passed before a signal was received, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

**Comments** Palm OS Emulator waits to be restarted after issuing a signal to allow external scripts to perform operations.

The predefined signals are documented under “[Host Signals](#)” on page 326.

**Compatibility** This function is defined for external RPC clients to call and returns an error if you call it from within a Palm OS application.

**See Also** [HostSignalResume\(\)](#), [HostSignalSend\(\)](#)

## HostSlotHasCard Function

- Purpose** Determine whether Emulator is emulating a Virtual File System (VFS) card for a specific slot number.
- Declared In** `HostControl.h`
- Prototype** `HostBoolType HostSlotHasCard (long slotNo)`
- Parameters** `→ slotNo`  
The slot number. This number can be in the range from 1 up to and including the number returned by function [HostSlotMax\(\)](#).
- Returns** Returns `true` if the Emulator is emulating a VFS card in the specified slot, `false` otherwise.
- Comments** This function may return `false` if the user has not selected to emulate a VFS card in the given slot, or if Emulator is emulating a different kind of card in that slot.
- This function is provided in support of Expansion Manager emulation.

## HostSlotMax Function

- Purpose** Determine the number of Virtual File System (VFS) cards that Emulator is emulating.
- Declared In** `HostControl.h`
- Prototype** `long HostSlotMax (void)`
- Parameters** None.
- Returns** The number of VFS cards Emulator is emulating.
- Comments** The functions that accept card numbers, [HostSlotHasCard\(\)](#) and [HostSlotRoot\(\)](#), accept numbers from 1 up to and including the number returned by this function.
- This function is provided in support of Expansion Manager emulation.

## HostSlotRoot Function

- Purpose** Get a string representing the root directory of the emulated Virtual File System (VFS) card in a specified slot.
- Declared In** `HostControl.h`
- Prototype** `const char *HostSlotRoot (long slotNo)`
- Parameters** → `slotNo`  
The slot number. This number can be in the range from 1 up to and including the number returned by function [HostSlotMax\(\)](#).
- Returns** A character string in host path format representing the directory used as the root for the given VFS card. This function may return NULL if there is no VFS card mounted in the slot specified by `slotNo`, or if the user has not selected a root directory for that slot.
- Comments** This function is provided in support of Expansion Manager emulation.
- See Also** [HostSlotHasCard\(\)](#)

## HostStat Function

- Purpose** Get status information about a file or directory.
- Declared In** `HostControl.h`
- Prototype** `long HostStat (const char *fileOrDirectory, HostStatType *info)`
- Parameters** → `fileOrDirectory`  
The name of the file or directory for which you want status information
- ← `info`  
A [HostStatType](#) structure that receives the status information
- Returns** Returns `errNone` if the operation was successful, or one of the error values listed in “[Host Control Error Codes](#)” on page 323 if an error occurred.

## HostStrFTime Function

- Purpose** Generate a text representation of the contents of a date/time structure using a specified format.
- Declared In** `HostControl.h`
- Prototype**  
`HostSizeType HostStrFTime (char *buffer,  
HostSizeType maxSize, const char *format,  
const HostTmType *timeP)`
- Parameters**
- ← *buffer*  
The formatted text.
  - *maxSize*  
The maximum number of characters (including the null terminator) to be placed into *buffer*.
  - *format*  
The format definition. See the man page for the standard C library `strftime()` function for documentation on how to construct this string.
  - *timeP*  
The date/time structure whose contents are to be converted to a text string.
- Returns** Returns the number of characters generated if the number is less than the *maxSize* parameter; otherwise, returns zero, and the characters stored in *buffer* are undefined.

## HostTime Function

- Purpose** Get the current calendar time.
- Declared In** `HostControl.h`
- Prototype**  
`HostTimeType HostTime (HostTimeType *time)`
- Parameters**
- ← *time*  
The time structure.
- Returns** Returns the current calendar time if the operation is successful, or -1 if an error occurred.
- See Also** [HostErrNo\(\)](#)

## Host Control

### *HostTmpFile*

---

#### HostTmpFile Function

- Purpose** Identify the temporary file used by the debugging host.
- Declared In** `HostControl.h`
- Prototype** `HostFileType *HostTmpFile (void)`
- Parameters** None.
- Returns** A pointer to the temporary file, or NULL if an error occurred.

#### HostTmpNam Function

- Purpose** Create a unique temporary filename.
- Declared In** `HostControl.h`
- Prototype** `char *HostTmpNam (char *name)`
- Parameters**  $\rightarrow$  *name*  
A pointer to a buffer into which the newly-created temporary filename is written, or NULL.
- Returns** A pointer to an internal static object that the calling program can modify.

#### HostTraceClose Function

- Purpose** Close the connection to the external trace reporting tool.
- Declared In** `HostControl.h`
- Prototype** `void HostTraceClose (void)`
- Parameters** None.
- Returns** Nothing.
- See Also** [HostTraceInit\(\)](#)

## HostTraceInit Function

|                    |                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Initiate a connection to the external trace reporting tool.                                                                                                           |
| <b>Declared In</b> | HostControl.h                                                                                                                                                         |
| <b>Prototype</b>   | void HostTraceInit (void)                                                                                                                                             |
| <b>Parameters</b>  | None.                                                                                                                                                                 |
| <b>Returns</b>     | Nothing.                                                                                                                                                              |
| <b>Comments</b>    | The HostTrace... functions are used in conjunction with an external trace reporting tool. Call these functions to send information to the external tool in real time. |
| <b>See Also</b>    | <a href="#">HostTraceClose()</a> , HostTraceOutput...                                                                                                                 |

## HostTraceOutputB Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Write a buffer of data, in hex dump format, to the external trace reporting tool.                                                                                                                                                                                                                                                                                                                          |
| <b>Declared In</b> | HostControl.h                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Prototype</b>   | void HostTraceOutputB (unsigned short <i>moduleId</i> ,<br>const void * <i>buffer</i> , long <i>len</i> )                                                                                                                                                                                                                                                                                                  |
| <b>Parameters</b>  | <p>→ <i>moduleId</i><br/>The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in CmnErrors.h.</p> <p>→ <i>buffer</i><br/>Pointer to a buffer of raw data to be output.</p> <p>→ <i>len</i><br/>The number of bytes of data in the buffer.</p> |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>See Also</b>    | <a href="#">HostTraceOutputT()</a> , <a href="#">HostTraceOutputTL()</a> ,<br><a href="#">HostTraceOutputVT()</a> , <a href="#">HostTraceOutputVTL()</a>                                                                                                                                                                                                                                                   |

## HostTraceOutputT Function

- Purpose** Write a formatted text string to the external trace reporting tool.
- Declared In** `HostControl.h`
- Prototype** `void HostTraceOutputT (unsigned short moduleId,  
const char *fmt, ...)`
- Parameters**
- *moduleId*  
The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`.
  - *fmt*  
A format string, as used in standard C-library calls such as `printf()`. See the Comments section, below, for more detail.
  - ...  
The list of variables to be formatted for output.
- Returns** Returns nothing.
- Comments** The format string has the following form:

`% flags width type`

[Table 29.2](#) shows the flag types that you can use in the format specification for the tracing output functions.

**Table 29.2 Trace function format specification flags**

| Flag  | Description                                                           |
|-------|-----------------------------------------------------------------------|
| -     | Left-justified output.                                                |
| +     | Always display the sign symbol.                                       |
| space | Display a space when the value is positive, rather than a '+' symbol. |
| #     | Alternate form specifier.                                             |

[Table 29.3](#) shows the output types that you can use in the format specification for the tracing output functions.

**Table 29.3 Trace function format specification types**

| <b>Flag</b>                        | <b>Description</b>                                                             |
|------------------------------------|--------------------------------------------------------------------------------|
| <code>%</code>                     | Display the '%' character.                                                     |
| <code>s</code>                     | Display a null-terminated string value.                                        |
| <code>c</code>                     | Display a character value.                                                     |
| <code>ld</code>                    | Display an <code>int32_t</code> value.                                         |
| <code>lu</code>                    | Display a <code>uint32_t</code> value.                                         |
| <code>lx</code> or <code>lX</code> | Display a <code>uint32_t</code> value in hexadecimal.                          |
| <code>hd</code>                    | Display an <code>int16_t</code> value.                                         |
| <code>hu</code>                    | Display a <code>uint16_t</code> value.                                         |
| <code>hx</code> or <code>hX</code> | Display an <code>int16_t</code> or <code>uint16_t</code> value in hexadecimal. |

**See Also** [HostTraceOutputB\(\)](#), [HostTraceOutputTL\(\)](#),  
[HostTraceOutputVT\(\)](#), [HostTraceOutputVTL\(\)](#)

## HostTraceOutputTL Function

**Purpose** Write a formatted text string, followed by a newline, to the external trace reporting tool. This function performs the same operation as [HostTraceOutputT\(\)](#) but adds the newline character.

**Declared In** `HostControl.h`

**Prototype** `void HostTraceOutputTL (unsigned short moduleId,  
const char *fmt, ...)`

**Parameters**  $\rightarrow$  `moduleId`

The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`.

## Host Control

### *HostTraceOutputVT*

---

→ *fmt*

A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [HostTraceOutputT\(\)](#) for more detail.

→ ...

The list of variables to be formatted for output.

**Returns** Returns nothing.

**See Also** [HostTraceOutputB\(\)](#), [HostTraceOutputVT\(\)](#),  
[HostTraceOutputVTL\(\)](#)

## HostTraceOutputVT Function

**Purpose** Write a formatted text string to the external trace reporting tool.

**Declared In** `HostControl.h`

**Prototype** `void HostTraceOutputVT (unsigned short moduleId,  
const char *fmt, va_list vargs)`

**Parameters** → *moduleId*

The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`.

→ *fmt*

A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [HostTraceOutputT\(\)](#) for more detail.

→ *vargs*

A structure containing the variable argument list. This is the same kind of variable argument list used for standard C-library functions such as `vprintf()`.

**Returns** Returns nothing.

**See Also** [HostTraceOutputB\(\)](#), [HostTraceOutputTL\(\)](#),  
[HostTraceOutputVTL\(\)](#)

## HostTraceOutputVTL Function

- Purpose** Write a formatted text string, followed by a newline, to the external trace reporting tool. This function performs the same operation as [HostTraceOutputVT\(\)](#) but adds the newline character.
- Declared In** `HostControl.h`
- Prototype** `void HostTraceOutputVTL (unsigned short moduleId,  
const char *fmt, va_list vargs)`
- Parameters**
- *moduleId*  
The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`.
  - *fmt*  
A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [HostTraceOutputT\(\)](#) for more detail.
  - *vargs*  
A structure containing the variable argument list. This is the same kind of variable argument list used for standard C-library functions such as `vprintf()`.
- Returns** Returns nothing.
- See Also** [HostTraceOutputB\(\)](#), [HostTraceOutputT\(\)](#), [HostTraceOutputTL\(\)](#)

## HostTruncate Function

- Purpose** Extend or truncate a file to a specified length.
- Declared In** `HostControl.h`
- Prototype** `long HostTruncate (const char *fileName,  
long size)`
- Parameters**
- *fileName*  
The name of the file.
  - *size*  
The size of the file.

## Host Control

*HostUTime*

---

**Returns** Returns 0 if the file was successfully resized, or -1 if an error occurred.

**See Also** [HostErrNo\(\)](#)

### HostUTime Function

**Purpose** Set the modification time for a file.

**Declared In** `HostControl.h`

**Prototype** `long HostUTime (const char *fileName,  
HostUTimeType *modTime)`

**Parameters** → *fileName*  
The name of the file.

→ *modTime*  
The time values to be applied to the file.

**Returns** Returns 0 if the file-modification time was successfully changed, or -1 if an error occurred.

**See Also** [HostErrNo\(\)](#)

### HostVFPRINTF Function

**Purpose** Write formatted output to a file.

**Declared In** `HostControl.h`

**Prototype** `long HostVFPRINTF (HostFILEType *f,  
const char *fmt, va_list args)`

**Parameters** → *f*  
The file to which the formatted output is to be written.

→ *fmt*  
The format string. See the man page for the standard C function `printf()` for instructions on creating this argument.

→ *args*  
Arguments to be written to the file, formatted according to *fmt*.

**Returns** Returns the number of characters written, not including the trailing null character used to end output to strings.

**See Also** [HostPrintf\(\)](#), [HostVFScanF\(\)](#), [HostVPrintF\(\)](#)

## HostVFScanF Function

**Purpose** Read input from a file using a variable argument list of pointers.

**Declared In** `HostControl.h`

**Prototype** `long HostVFScanF (HostFILEType *f,  
const char *fmt, va_list args)`

**Parameters** → *f*

The file from which to read.

→ *fmt*

The format string that specifies how the input data is to be interpreted. See the man page for the standard C function `vfscanf()` for instructions on creating this argument.

→ *args*

Pointers to the variables in which the data is deposited.

**Returns** Returns the number of input items assigned, or EOF if an input failure occurred before any conversion was done. Note that in the event of a matching failure the number of input items assigned can be fewer than provided for.

**See Also** [HostGetChar\(\)](#), [HostVFPrintF\(\)](#)

## HostVPrintF Function

**Purpose** Send formatted output to stdout.

**Declared In** `HostControl.h`

**Prototype** `long HostVPrintF (const char *fmt, va_list args)`

**Parameters** → *fmt*

The format string. See the man page for the standard C function `printf()` for instructions on creating this argument.

## Host Control

*HostVPrintF*

---

→ *args*

Arguments to be sent to stdout, formatted according to *fmt*.

**Returns** Returns the number of characters printed, not including the trailing null character used to end output to strings.

**See Also** [HostPrintf\(\)](#), [HostVFPrintf\(\)](#), [HostVFScanF\(\)](#)  
“[Host Control Function Selectors](#)” on page 318.

# Loader

---

The Program Loader is responsible for loading and unloading executable programs in the calling process. It also provides the means for retrieving information about executable modules and means for patching shared library entries. This chapter provides reference material for the loader as well as the dynamic linker, which links executables with shared libraries. The contents of this chapter are organized into the following sections:

|                                                         |     |
|---------------------------------------------------------|-----|
| <a href="#">Loader Structures and Types</a> . . . . .   | 380 |
| <a href="#">Loader Constants</a> . . . . .              | 382 |
| <a href="#">Loader Launch Codes</a> . . . . .           | 383 |
| <a href="#">Loader Functions and Macros</a> . . . . .   | 384 |
| <a href="#">Application-Defined Functions</a> . . . . . | 395 |

The header file `Loader.h` declares the API that this chapter describes.

See [Chapter 6, “Shared Libraries,”](#) on page 71 for guidance on using, creating, and patching shared libraries.

## Loader Structures and Types

### **SysModuleInfoType Struct**

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Contains module information retrieved with <a href="#">SysGetModuleInfo()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Declared In</b> | Loader.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Prototype</b>   | <pre>typedef struct SysModuleInfoType {     uint32_t revision;     uint32_t entries;     uint32_t dataSize;     uint32_t minArch;     uint32_t minOS;     uint32_t currArch;     uint32_t currOS; } SysModuleInfoType</pre>                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Fields</b>      | <p><b>revision</b><br/>The module revision number specified as a pslib command-line parameter when the module was built.</p> <p><b>entries</b><br/>The total number of entry points exported by this module.</p> <p><b>dataSize</b><br/>The amount of memory, in bytes, needed to hold the module's data segment.</p> <p><b>minArch</b><br/>The minimum required processor architecture.</p> <p><b>minOS</b><br/>The minimum required OS version.</p> <p><b>currArch</b><br/>The processor architecture of the device on which this is running.</p> <p><b>currOS</b><br/>The OS version of the device on which this is running.</p> |

## SysPatchInfoType Struct

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Accompanies a <a href="#">sysPatchLaunchCmdSetInfo</a> launch code and identifies both the shared library being patched and the patch's location within the call chain.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Declared In</b> | Loader.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Prototype</b>   | <pre>typedef struct SysPatchInfoType {     uint32_t refNum;     uint32_t type;     uint32_t creator;     uint16_t rsrcID;     uint16_t reserved;     uint32_t index;     status_t (*sysGetNextPatchP) (uint32_t,         uint32_t, uint32_t, void **); } SysPatchInfoType</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Fields</b>      | <p><b>refNum</b><br/>Reference number identifying the patched shared library.</p> <p><b>type</b><br/>The type of the database that contains the shared library.</p> <p><b>creator</b><br/>The creator ID of the database that contains the shared library.</p> <p><b>rsrcID</b><br/>The resource ID of the shared library resource containing the code for the executable module being patched.</p> <p><b>reserved</b><br/>Reserved for future use.</p> <p><b>index</b><br/>An index that identifies this patch's position within the call chain. The first patch in the call chain has index number zero.</p> <p><b>sysGetNextPatchP</b><br/>Pointer to a function that retrieves the address of the next patching procedure in a patched call chain.</p> |
| <b>Comments</b>    | This structure accompanies a <a href="#">sysPatchLaunchCmdSetInfo</a> launch code. Both are sent to a patch when the shared library that it patches is being loaded. Upon receipt of <code>sysPatchLaunchCmdSetInfo</code> , the patch will likely want to use                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Loader

### Loader Constants

---

the information in this structure to set aside pointers to functions in the next patch in the call chain. This allows the patch, after it has done its work, to invoke the next patch in the chain.

For sample code showing how a patch can record function addresses from in the next patch in the chain, see [Listing 6.3](#) on page 78.

## Loader Constants

### Miscellaneous Loader Constants

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Loader.h declares these constants.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Declared In</b> | Loader.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Constants</b>   | <pre>#define sysDoNotVerifySignature ((uint32_t)0x00000001)</pre> <p>A flag that can be passed to <a href="#">SysLoadModule()</a> so that the program loader doesn't perform verification of the digital signature on the executable module even if the security property of the calling process requires that the digital signature be verified. This behavior is useful for certain type of applications—like web browsers—that have their own ways of verifying the integrity of downloaded programs.</p> <pre>#define sysEntryNumMain ((uint32_t)0xffffffff)</pre> <p>Pass this as the starting entry number parameter of <a href="#">SysGetEntryAddresses()</a> to retrieve only the address of the main entry point.</p> |

## Loader Launch Codes

### **sysPatchLaunchCmdClearInfo**

- Purpose** Informs a patch that a target shared library has been unloaded.
- Declared In** `CmnLaunchCodes.h`
- Prototype**

```
#define sysPatchLaunchCmdClearInfo 0x7ff3
```
- Parameters** The `cmdPBP` parameter for this launch code is a pointer to a [SysPatchInfoType](#) structure.
- Comments** A patch's [PilotMain\(\)](#) function receives this launch code once for each of the shared libraries it patches. Thus, if a given patch patches multiple shared libraries, it will receive this launch code multiple times.
- See Also** [sysPatchLaunchCmdSetInfo](#)

### **sysPatchLaunchCmdSetInfo**

- Purpose** Informs a patch that one of the shared libraries it wants to patch is being loaded.
- Declared In** `CmnLaunchCodes.h`
- Prototype**

```
#define sysPatchLaunchCmdSetInfo 0x7ffb
```
- Parameters** The `cmdPBP` parameter for this launch code is a pointer to a [SysPatchInfoType](#) structure.
- Comments** A patch's [PilotMain\(\)](#) function receives this launch code once for each of the shared libraries it patches. Thus, if a given patch patches multiple shared libraries, it will receive this launch code multiple times.
- This launch code provides the patch with a good opportunity to retrieve and save addresses of functions in the next patch in the call chain.

## Loader

### Loader Functions and Macros

---

For sample code showing how a patch can record function addresses from in the next patch in the chain, see [Listing 6.3](#) on page 78.

**See Also** [SysGetModuleGlobals\(\)](#), [sysPatchLaunchCmdSetInfo](#)

## Loader Functions and Macros

### SysGetEntryAddresses Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Retrieve addresses of one or more entry points exported by a loaded executable module.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Declared In</b> | <code>Loader.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Prototype</b>   | <pre>status_t SysGetEntryAddresses (uint32_t refNum,                                uint32_t startEntryNum, uint32_t numEntries,                                void **addressP)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Parameters</b>  | <p>→ <i>refNum</i><br/>Reference number of the executable module as returned by a call to <a href="#">SysLoadModule()</a>.</p> <p>→ <i>startEntryNum</i><br/>Entry number of the first of <i>numEntries</i> entry point addresses to be retrieved, or <code>sysEntryNumMain</code> to retrieve only the address of the main entry point. Exported entry point index values begin at zero.</p> <p>→ <i>numEntries</i><br/>Total number of entry point addresses to retrieve. If <i>startEntryNum</i> is <code>sysEntryNumMain</code>, this parameter is ignored.</p> <p>← <i>addressP</i><br/>Pointer to a buffer that receives the returned addresses. The size of this buffer should be no less than <i>numEntries</i>*4 bytes.</p> |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation succeeded, or one of the following otherwise:<br><code>sysErrParamErr</code><br><i>refNum</i> doesn't reference a loaded shared library, <i>addressP</i> is NULL, or either <i>startEntryNum</i> or ( <i>startEntryNum</i> +                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

*numEntries - 1*) is outside the range of exported entry points.

**Comments** A shared library can have multiple exported entries, each identified by an entry number assigned when the shared library is built. Entry numbers are assigned in the shared library definition file (SLD), where the names of exported entries are listed in the order of increasing entry numbers starting from zero.

**See Also** [SysGetModuleGlobals\(\)](#)

## SysGetModuleDatabase Function

**Purpose** Retrieve the database ID, an open reference, or both for a database that contains a loaded executable module.

**Declared In** `Loader.h`

**Prototype** `status_t SysGetModuleDatabase (uint32_t refNum,  
DatabaseID *dbIDP, DmOpenRef *openRefP)`

**Parameters** → *refNum*

Reference number of the loaded executable module as returned by a call to [SysLoadModule\(\)](#).

← *dbIDP*

Database ID of the database that contains the executable module. Pass NULL for this parameter if you don't need the database ID.

← *openRefP*

Open reference of the database that contains the executable module. Pass NULL for this parameter if you don't need the reference.

**Returns** Returns `errNone` if the operation succeeded, or one of the following otherwise:

`sysErrParamErr`

*refNum* doesn't identify a loaded executable module.

`sysErrNoFreeResource`

*openRefP* is not NULL and the database that contains the executable module identified by *refNum* could not be opened.

## Loader

### *SysGetModuleGlobals*

---

**Comments**     **WARNING!** Do not call `DmCloseDatabase()` with the `DmOpenRef` you obtain from this function. The referenced database was opened by the system in read-only mode with overlays. It will be closed automatically when the module is unloaded.

---

**See Also**     [SysGetModuleInfo\(\)](#), [SysGetModuleInfoByDatabaseID\(\)](#)

## SysGetModuleGlobals Function

**Purpose**     Retrieve the address of the C global structure or the address of the data segment of a loaded executable module.

**Declared In**     `Loader.h`

**Prototype**     `status_t SysGetModuleGlobals (uint32_t refNum,  
                                  Boolean wantStructure, void **globalsP)`

**Parameters**      $\rightarrow$  *refNum*  
                                  Reference number of an executable module as returned by a call to [SysLoadModule\(\)](#).

$\rightarrow$  *wantStructure*  
                                  Pass `true` to retrieve the base address of the C global structure, `false` to retrieve the address of the executable module's data segment.

$\leftarrow$  *globalsP*  
                                  Pointer to a memory location into which the address of the C global structure or data segment is to be written.

**Returns**     Returns `errNone` if the operation succeeded. Returns one of the following otherwise:

`sysErrNotSupported`  
                  The executable module specified by *refNum* doesn't have a C global structure or doesn't allow it to be retrieved. In this case, the address of the module's data segment will be returned in *\*globalsP*. This error can be returned only when *wantStructure* is `true`.

`sysErrParamErr`  
                  *refNum* doesn't reference a loaded shared library. The memory pointed to by *addressP* is set to `NULL`.

**Comments** This function provides an indirect way for an executable module to access another executable module's global data. Patches can utilize this function to gain access to the globals of the original shared library.

Whether `SysGetModuleGlobals()` is able to return the address of the globals structure depends on whether the module identified by `refNum` defines such a structure and whether it returns the structure's address in response to the [sysLaunchCmdGetGlobals](#) launch code. `SysGetModuleGlobals()` returns `sysErrNotSupported` if `wantStructure` is true and the module doesn't allow the globals structure address to be retrieved.

Note that if you retrieve the address of the executable module's data segment, you must possess sufficient knowledge of the memory map of the module's data segment in order to access data located at particular offsets.

If `*globalsP` is NULL after retrieving the base address of the module's data segment, the executable module has no static data.

**See Also** [SysGetEntryAddresses\(\)](#)

## SysGetModuleInfo Function

**Purpose** Retrieve information about an executable module, given the type and creator of the database containing the executable module.

**Declared In** `Loader.h`

**Prototype**

```
status_t SysGetModuleInfo (uint32_t dbType,  
                           uint32_t dbCreator, uint16_t rsrcID,  
                           SysModuleInfoType *infoP)
```

**Parameters** → `dbType`  
Type of the database that contains the executable module.

→ `dbCreator`  
Creator ID of the database that contains the executable module.

→ `rsrcID`  
ID number of the resources in the database that contain the code, data, and relocation information of the executable module.

## Loader

### *SysGetModuleInfoByDatabaseID*

---

← *infoP*

Pointer to a [SysModuleInfoType](#) structure into which the module information is written.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`sysErrModuleNotFound`

Module not found.

`sysErrModuleInvalid`

Code resource has an invalid or corrupted format.

**Comments** This function is usually called by a client to get information about an executable module before actually loading it. The information returned includes the module's revision number, the total number of exported entries, and the size of the module's data segment.

**See Also** [SysGetModuleDatabase\(\)](#),  
[SysGetModuleInfoByDatabaseID\(\)](#), [SysLoadModule\(\)](#)

## SysGetModuleInfoByDatabaseID Function

**Purpose** Retrieve information about an executable module, given the database ID of the database containing the executable module.

**Declared In** `Loader.h`

**Prototype** `status_t SysGetModuleInfoByDatabaseID  
(DatabaseID dbID, uint16_t rsrcID,  
SysModuleInfoType *infoP)`

**Parameters** → *dbID*  
Database ID of the database that contains the executable module.

→ *rsrcID*  
ID number of the resources in the database that contain the code, data and relocation information of the executable module.

← *infoP*

Pointer to a [SysModuleInfoType](#) structure into which the module information is written.

**Returns** Returns `errNone` if the operation completed successfully, or one of the following otherwise:

`sysErrModuleNotFound`  
Module not found.

`sysErrModuleInvalid`  
Code resource has an invalid or corrupted format.

**Comments** This function is usually called by a client to get information about an executable module before actually loading it. The information returned includes the module's revision number, the total number of exported entries, and the size of the module's data segment.

**See Also** [SysGetModuleDatabase\(\)](#), [SysGetModuleInfo\(\)](#), [SysLoadModuleByDatabaseID\(\)](#)

## SysGetRefNum Function

**Purpose** Retrieve the reference number of the calling module.

**Declared In** `Loader.h`

**Prototype** `uint32_t SysGetRefNum (void)`

**Parameters** None.

**Returns** Reference number of the calling module.

**Comments** This function is often used along with [SysGetModuleDatabase\(\)](#) to retrieve information about the database that contains the calling module. The reference number returned by this function has the same value that was returned by [SysLoadModule\(\)](#) when the module was loaded.

## SysLoadModule Function

**Purpose** Load an executable module and make it ready for execution in the calling process, given the type and creator of the database that contains the executable module and the resource ID of the resource containing the executable module.

## Loader

### *SysLoadModule*

---

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Declared In</b> | Loader.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Prototype</b>   | <code>status_t SysLoadModule (uint32_t dbType,<br/>uint32_t dbCreator, uint16_t rsrcID,<br/>uint32_t flags, uint32_t *refNumP)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Parameters</b>  | <p>→ <i>dbType</i><br/>Type of the database that contains the executable module.</p> <p>→ <i>dbCreator</i><br/>Creator ID of the database that contains the executable module.</p> <p>→ <i>rsrcID</i><br/>ID number of the resources in the database that contain the code, data and relocation information for the executable module.</p> <p>→ <i>flags</i><br/>Bits set or cleared in this 32-bit word indicate specific requirements that must be met while loading the executable module. See the Comments section, below, for more information on this argument.</p> <p>← <i>refNumP</i><br/>If the module is successfully loaded, the location to which this parameter points receives the reference number of the newly-loaded executable module.</p> |
| <b>Returns</b>     | Returns <code>errNone</code> if the operation succeeded, or one of the following otherwise:<br><br><code>sysErrModuleNotFound</code><br>Module not found.<br><br><code>sysErrModuleInvalid</code><br>Code resource has an invalid or corrupted format.<br><br><code>sysErrInvalidSignature</code><br>Module found but has no valid signature.<br><br><code>sysErrNoFreeRAM</code><br>There isn't enough free RAM to load the executable module.<br><br><code>sysErrCPUArchitecture</code><br>The program requires a different CPU architecture to run.                                                                                                                                                                                                       |

`sysErrOSVersion`

The program requires a higher version of the operating system in order to run.

`sysErrRAMModuleNotAllowed`

The program requires a higher version of the operating system in order to run.

`sysErrNoFreeResource`

**Comments**

A single resource database can contain multiple executable modules, each identified by a unique resource ID within that database. Each executable module is composed of several resources of different types, all of which have the same unique resource ID number within the containing database. The resource ID number is specified to `pslib` when the module is built. The same ID number should be specified when the executable is placed into the resource database (PRC file). By default, the resource ID number is zero.

Set the `flags` argument to 0 if the code has no special security requirements. A value of zero instructs the program loader to perform verification of digital signatures on the executable program according to the security property of the calling process. If the digital signature verification fails, the program loader doesn't load the program; `sysErrInvalidSignature` is returned to the caller.

Set the `flags` argument to `sysDoNotVerifySignature` to prevent the program loader from performing any verification of the executable module's digital signature, even if the security property of the calling process requires that the signature be verified. This is useful for certain type of applications—like web browsers—that have their own ways of verifying the integrity of downloaded programs.

---

**NOTE:** If there are multiple versions of the same database, the newest will be loaded.

---

If `SysLoadModule()` is successful, once the loaded module is no longer needed call [SysUnloadModule\(\)](#) to unload the executable module.

**See Also**

[SysGetEntryAddresses\(\)](#), [SysGetModuleInfo\(\)](#), [SysLoadModuleByDatabaseID\(\)](#), [SysUnloadModule\(\)](#)

## Loader

### *SysLoadModuleByDatabaseID*

---

## SysLoadModuleByDatabaseID Function

- Purpose** Load an executable module and make it ready for execution in the calling process, given the database ID of the database that contains the executable module.
- Declared In** `Loader.h`
- Prototype**  
`status_t SysLoadModuleByDatabaseID  
(DatabaseID dbID, uint16_t rsrcID,  
uint32_t flags, uint32_t *refNumP)`
- Parameters**
- *dbID*  
Database ID of database that contains the executable module.
  - *rsrcID*  
ID number of the resources in the database that contain the code, data and relocation information for the executable module.
  - *flags*  
Bits set or cleared in this 32-bit word indicate specific requirements that must be met while loading the executable module. See the Comments section under [SysLoadModule\(\)](#) for more information on this argument.
  - ← *refNumP*  
If the module is successfully loaded, the location to which this parameter points receives the reference number of the newly-loaded executable module.
- Returns** Returns `errNone` if the operation succeeded, or one of the following otherwise:
- `sysErrModuleNotFound`  
Module not found.
  - `sysErrModuleInvalid`  
Code resource has an invalid or corrupted format.
  - `sysErrInvalidSignature`  
Module found but has no valid signature.
  - `sysErrNoFreeRAM`  
There isn't enough free RAM to load the executable module.
  - `sysErrCPUArchitecture`  
The program requires a different CPU architecture to run.

`sysErrOSVersion`

The program requires a higher version of the operating system in order to run.

`sysErrRAMModuleNotAllowed`

The program requires a higher version of the operating system in order to run.

`sysErrNoFreeResource`

**Comments** This function performs the same operation as [SysLoadModule\(\)](#) except that it takes a database ID as input to identify the containing database of the executable module. This function is often used to load plug-ins.

See the `SysLoadModule()` function's Comments section for additional information on the loading of executable modules.

**See Also** [SysGetModuleInfoByDatabaseID\(\)](#), [SysLoadModule\(\)](#), [SysUnloadModule\(\)](#)

## SysRegisterPatch Function

**Purpose** Register a patch that is not packaged in a resource database of type 'apch'.

**Declared In** `Loader.h`

**Prototype** `status_t SysRegisterPatch (uint32_t type, uint32_t creator, uint16_t rsrcID)`

**Parameters** → *type*

Type of the database that contains the patch.

→ *creator*

Creator ID of the database that contains the patch.

→ *rsrcID*

ID number of the resource in the database that contains the patch.

**Returns** Returns `errNone` if the operation succeeded, or one of the following otherwise:

`sysErrModuleNotFound`

Patch not found.

## Loader

### *SysUnloadModule*

---

`sysErrModuleInvalid`

Patch resources have invalid or corrupted format.

`sysErrNoFreeRAM`

There isn't enough free RAM to register the patch.

**See Also** [SysUnregisterPatch\(\)](#)

## SysUnloadModule Function

**Purpose** Unload an executable module and free resources that were allocated when it was loaded.

**Declared In** `Loader.h`

**Prototype** `status_t SysUnloadModule (uint32_t refNum)`

**Parameters** → *refNum*

Reference number of the executable module returned by a former call to [SysLoadModule\(\)](#).

**Returns** Returns `errNone` if the operation succeeded, or `sysErrParamErr` if *refNum* doesn't refer to a loaded shared library.

**See Also** [SysLoadModule\(\)](#), [SysLoadModuleByDatabaseID\(\)](#)

## SysUnregisterPatch Function

**Purpose** Unregister a patch that was registered with [SysRegisterPatch\(\)](#).

**Declared In** `Loader.h`

**Prototype** `status_t SysUnregisterPatch (uint32_t type, uint32_t creator, uint16_t rsrcID)`

**Parameters** → *type*

Type of the database that contains the patch.

→ *creator*

Creator ID of the database that contains the patch.

→ *rsrcID*

ID number of the resource in the database that contains the patch.

**Returns** Returns `errNone` if the operation succeeded, or one of the following otherwise:

`sysErrModuleNotFound`  
Patch not found.

`sysErrNoFreeRAM`  
Out of free RAM while unregistering the patch.

**See Also** [SysRegisterPatch\(\)](#)

## Application-Defined Functions

### SysMainEntryPtrType Function

**Purpose** The prototype of the main entry point of any executable module.

**Declared In** `Loader.h`

**Prototype** `uint32_t (*SysMainEntryPtrType) (uint16_t cmd, MemPtr cmdPBP, uint16_t launchFlags)`

**Parameters**

- `cmd`  
The launch code to which your executable module is to respond. See [Chapter 2, “Application Start and Stop,”](#) on page 21 of *Exploring Palm OS: Programming Basics* for a list of predefined launch codes.
- `cmdPBP`  
A pointer to a structure containing any launch command-specific parameters, or `NULL` if the launch code has none. See the description of each launch code for a description of the parameter structure that accompanies it, if any.
- `launchFlags`  
Flags that indicate whether your application is now the active application, whether it already was the active application, and so on. See “[Launch Flags](#)” on page 105 of *Exploring Palm OS: Programming Basics* for a list of launch flags.

**Returns** Your executable module should return `errNone` if your application processed the launch code successfully or if the launch code is not recognized. Return an appropriate error code for recognized launch codes if there was a problem. When application invokes your

## Loader

*SysMainEntryPtrType*

---

executable module using [SysAppLaunch\(\)](#), the value you return from this function is returned to the caller.

**Comments** See [Chapter 2, “Application Start and Stop,”](#) on page 21 of *Exploring Palm OS: Programming Basics* for a discussion on how applications receive and handle launch codes, with examples.

# Patch

---

This chapter describes APIs that you use to create patches to augment or replace one or more shared library functions. The material in this chapter is organized into the following sections:

|                                                      |     |
|------------------------------------------------------|-----|
| <a href="#">Patch Structures and Types</a> . . . . . | 397 |
| <a href="#">Patch Constants</a> . . . . .            | 399 |

The header file `Patch.h` declares the API that this chapter describes.

For instructions on using these APIs to patch a shared library, see [Chapter 6, “Shared Libraries,”](#) on page 71.

## Patch Structures and Types

### **SysPatchEntryNumType Typedef**

|                    |                                                                                                                                                                        |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Container for a shared library vector table entry number.                                                                                                              |
| <b>Declared In</b> | <code>Patch.h</code>                                                                                                                                                   |
| <b>Prototype</b>   | <code>typedef uint32_t SysPatchEntryNumType</code>                                                                                                                     |
| <b>Comments</b>    | The <a href="#">SysPatchTargetHeaderType</a> structure is followed by one or more values of this type, indicating the shared library functions that are being patched. |

## Patch

*SysPatchTargetHeaderType*

---

### **SysPatchTargetHeaderType Struct**

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Patch header, identifying the shared library being patched and the number of entry points in that shared library that are being patched.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Declared In</b> | <code>Patch.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Prototype</b>   | <pre>typedef struct SysPatchTargetHeaderType {     uint32_t type;     uint32_t creator;     uint16_t rsrcID;     uint16_t flags;     uint32_t numEntries; } SysPatchTargetHeaderType</pre>                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Fields</b>      | <p><code>type</code><br/>Type of the shared library to patch.</p> <p><code>creator</code><br/>Creator ID of the shared library to patch.</p> <p><code>rsrcID</code><br/>Resource ID of the shared library to patch.</p> <p><code>flags</code><br/>Zero—indicating that the patch can occur at any position within the patch call chain—or one of the <a href="#">Patch Flags</a> to request that the patch be placed at the head or the tail of the patch call chain.</p> <p><code>numEntries</code><br/>Total number of entry points being patched.</p>                              |
| <b>Comments</b>    | This structure should be followed immediately by one or more <a href="#">SysPatchEntryNumType</a> values (the number of values is specified in the <code>SysPatchTargetHeaderType</code> structure's <code>numEntries</code> field). Each value is an entry number in the shared library's vector table and indicates a function that is being patched. The order in which these vector table entry numbers occur must match the patch's exports as defined in the patch's Shared Library Definition (SLD) file, and the values must appear in the order of increasing entry numbers. |

## Patch Constants

### Patch Flags

|                    |                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Indicate how the patch should be inserted into the patch chain.                                                                                                                                                                                                                                                                                                           |
| <b>Declared In</b> | <code>Patch.h</code>                                                                                                                                                                                                                                                                                                                                                      |
| <b>Constants</b>   | <pre>#define patchFlagHead ((uint16_t)0x0001)</pre> <p>The patch should be placed at the head of the patch call chain.</p> <pre>#define patchFlagTail ((uint16_t)0x0002)</pre> <p>The patch should be placed at the tail of the patch call chain.</p>                                                                                                                     |
| <b>Comments</b>    | Note that just because you request that a patch be placed at the head or tail of the patch call chain, there is no guarantee that your patch will actually be placed there. The operating system provides no mechanism to arbitrate between two patches that patch the same shared library function and that both need to be at the head or tail of the patch call chain. |

### Miscellaneous Patch Constants

|                    |                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | <code>Patch.h</code> also defines the following constants.                                                                                                                      |
| <b>Declared In</b> | <code>Patch.h</code>                                                                                                                                                            |
| <b>Constants</b>   | <pre>#define patchFlagReservedMask ((uint16_t)0xffffc)</pre> <p>A mask that isolates those <a href="#">SysPatchTargetHeaderType</a> flags that are reserved for system use.</p> |

## **Patch**

*Miscellaneous Patch Constants*

---

# PerfDriver

---

[PerfDriver Structures and Types](#). . . . . 401

[PerfDriver Constants](#). . . . . 402

The header file `PerfDriver.h` declares the API that this chapter describes.

## PerfDriver Structures and Types

### PerfGenCPUClockInfoType Struct

|                    |                                                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     |                                                                                                                                                                                                                                |
| <b>Declared In</b> | <code>PerfDriver.h</code>                                                                                                                                                                                                      |
| <b>Prototype</b>   | <pre>typedef struct PerfGenCPUClockInfoType {     uint32_t minClock;     uint32_t maxClock;     uint32_t defClock;     uint32_t curClock;     uint32_t numClockModes; } PerfGenCPUClockInfoType, *PerfGenCPUClockInfoPtr</pre> |
| <b>Fields</b>      | <p><code>minClock</code></p> <p><code>maxClock</code></p> <p><code>defClock</code></p> <p><code>curClock</code></p> <p><code>numClockModes</code></p>                                                                          |

## PerfRefNumType Typedef

**Purpose**  
**Declared In** PerfDriver.h  
**Prototype** typedef uint32\_t PerfRefNumType

## PerfResultType Struct

**Purpose**  
**Declared In** PerfDriver.h  
**Prototype** typedef struct PerfResultType {  
    uint32\_t clockVal;  
    uint32\_t extraData;  
} PerfResultType, \*PerfResultPtr  
**Fields** clockVal  
  
extraData

## PerfDriver Constants

**Purpose**  
**Declared In** PerfDriver.h  
**Constants** #define kCancelPerfRequest (perfErrorClass | 5)  
  
#define kCPUClockInfoVersion\_0 0  
  
#define kCreatePerfRequest (perfErrorClass | 4)  
  
#define kCurrentCPUClockInfoVersion  
    kCPUClockInfoVersion\_0

```
#define kGetCPUClockInfo (perfErrorClass | 1)

#define kGetCPUClockRateArray (perfErrorClass | 2)

#define kPerfClockValueDelta 0x80000000

#define kPerfClockValueMax 0xFFFFFFFF

#define kPerfRequestAny 0x00000000

#define kSetDefaultCPUClockRate (perfErrorClass |
    3)

#define perfErrBufferTooSmall (perfErrorClass | 4)

#define perfErrDeniedPowerLow (perfErrorClass | 5)

#define perfErrInvalidParams (perfErrorClass | 1)

#define perfErrLimitReached (perfErrorClass | 2)

#define perfErrNone errNone

#define perfErrNotImplemented (perfErrorClass | 3)
```

## PerfDriver

---

# Preferences

---

This chapter documents the APIs you use when getting and setting system-wide and application-specific preferences. The material in this chapter is organized as follows:

|                                                  |     |
|--------------------------------------------------|-----|
| <a href="#">Preferences Structures and Types</a> | 405 |
| <a href="#">Preferences Constants</a>            | 406 |
| <a href="#">Preferences Functions and Macros</a> | 416 |

The header file `Preferences.h` declares the API that this chapter describes. For more information on preferences, see [Chapter 3, “Preferences,”](#) on page 37.

## Preferences Structures and Types

### PrefActivePanelParamsType Struct

|                    |                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Defines the parameter block that accompanies a <a href="#">prefAppLaunchCmdSetActivePanel</a> launch code. This parameter block identifies the active panel. |
| <b>Declared In</b> | <code>Preferences.h</code>                                                                                                                                   |
| <b>Prototype</b>   | <pre>typedef struct {     uint32_t activePanel; } PrefActivePanelParamsType typedef PrefActivePanelParamsType *PrefActivePanelParamsPtr</pre>                |
| <b>Fields</b>      | <p><code>activePanel</code><br/>The active panel.</p>                                                                                                        |

## Preferences Constants

### MeasurementSystemType Typedef

|                    |                                                                                                                                                                               |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The system of measurement that the system is to use.                                                                                                                          |
| <b>Declared In</b> | <code>Preferences.h</code>                                                                                                                                                    |
| <b>Prototype</b>   | <code>typedef Enum8 MeasurementSystemType</code>                                                                                                                              |
| <b>Constants</b>   | <code>unitsEnglish = 0</code><br>The English measurement system (feet, inches, and so on).<br><code>unitsMetric</code><br>The Metric system (meters, centimeters, and so on). |

### SoundLevelTypeV20 Typedef

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| <b>Purpose</b>     | Specifies whether certain sounds are enabled or disabled.                |
| <b>Declared In</b> | <code>Preferences.h</code>                                               |
| <b>Prototype</b>   | <code>typedef Enum8 SoundLevelTypeV20</code>                             |
| <b>Constants</b>   | <code>s1On = 0</code><br>Enabled.<br><code>s1Off = 1</code><br>Disabled. |

### SystemPreferencesChoice Typedef

|                |                                                                                                                                                                                                                        |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b> | A system preference value. You can pass these values to <a href="#">PrefGetPreference()</a> and <a href="#">PrefSetPreference()</a> to retrieve or set a system preference value. There is one constant for each field |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

in the `SystemPreferencesType` structure, which should be considered private.

**Declared In** `Preferences.h`

**Prototype** `typedef Enum8 SystemPreferencesChoice`

**Constants** The following table lists and describes the constants that this enum defines. For each constant, it shows what type of data is returned by [PrefGetPreference\(\)](#) for that constant.

---

| <b>Constant</b>                 | <b>Type</b>                    | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefVersion</code>        | <code>uint16_t</code>          | The preferences version number.                                                                                                                                                                                                                                                                                                                                                                             |
| <code>prefCountry68K</code>     | <code>CountryType</code>       | The country for which the device was built. This preference is intended to be used by applications running under PACE—although such applications should really use the <code>prefLocale</code> value instead. Native ARM applications should call <code>LmGetFormatsLocale()</code> to find out what locale the user has selected in the Formats panel, and <code>LmSetFormatsLocale()</code> to change it. |
| <code>prefDateFormat</code>     | <a href="#">DateFormatType</a> | The short format used to display dates. For example: 95/12/31                                                                                                                                                                                                                                                                                                                                               |
| <code>prefLongDateFormat</code> | <a href="#">DateFormatType</a> | The long format used to display dates. For example: 31 Dec 1995                                                                                                                                                                                                                                                                                                                                             |
| <code>prefWeekStartDay</code>   | <code>int8_t</code>            | The first day of the week (Sunday or Monday). Days of the week are numbered from 0 to 6 starting with Sunday = 0.                                                                                                                                                                                                                                                                                           |

## Preferences

*SystemPreferencesChoice*

---

| Constant                               | Type                              | Description                                                                                                                                                                                                                                                                                     |
|----------------------------------------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefTimeFormat</code>            | <a href="#">TimeFormatType</a>    | The format used to display time values.                                                                                                                                                                                                                                                         |
| <code>prefNumberFormat</code>          | <code>NumberFormatType</code>     | The format used for numbers, with regards to the thousands separator and the decimal point.                                                                                                                                                                                                     |
| <code>prefAutoOffDuration</code>       | <code>uint8_t</code>              | Minutes of user idle time before the device powers off. The default value for this preference is specified by the <code>defaultAutoOffDuration</code> constant. <code>prefAutoOffDuration</code> is replaced by <code>prefAutoOffDurationSecs</code> in version 8 of the preferences structure. |
| <code>prefSysSoundLevelV20</code>      | <a href="#">SoundLevelTypeV20</a> | Specifies whether system sounds are enabled or disabled.                                                                                                                                                                                                                                        |
| <code>prefGameSoundLevelV20</code>     | <a href="#">SoundLevelTypeV20</a> | Specifies whether game sound effects are on or off.                                                                                                                                                                                                                                             |
| <code>prefAlarmSoundLevelV20</code>    | <a href="#">SoundLevelTypeV20</a> | Specifies whether alarm sounds are on or off.                                                                                                                                                                                                                                                   |
| <code>prefHidePrivateRecordsV33</code> | <code>Boolean</code>              | If <code>true</code> , applications should not display database records that have the secret attribute bit set.                                                                                                                                                                                 |
| <code>prefDeviceLocked</code>          | <code>Boolean</code>              | If <code>true</code> , the device is locked. When the device is locked, it remains so until the user enters the password.                                                                                                                                                                       |

| <b>Constant</b>                                  | <b>Type</b>                          | <b>Description</b>                                                                                                                                                                    |
|--------------------------------------------------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefLocalSyncRequire<br/>sPassword</code>  | Boolean                              | If <code>true</code> , the user must enter a password before a HotSync® operation can be performed.                                                                                   |
| <code>prefRemoteSyncRequir<br/>esPassword</code> | Boolean                              | If <code>true</code> , the user must enter a password on the desktop computer before a HotSync operation can be performed.                                                            |
| <code>prefSysBatteryKind</code>                  | Sys<br>Battery<br>Kind               | The type of batteries installed. Use <a href="#">SysBatteryInfo()</a> to retrieve the battery type instead of this preference.                                                        |
| <code>prefAllowEasterEggs</code>                 |                                      |                                                                                                                                                                                       |
| <code>prefMinutesWestOfGMT</code>                | uint32_t                             | The time zone given as minutes <i>east</i> (not west, as the name implies) of Greenwich Mean Time (GMT). For preferences version 9 and higher, use <code>prefTimeZone</code> instead. |
| <code>prefDaylightSavings</code>                 | <a href="#">DaylightSavingsTypes</a> | The type of daylight savings correction. For preferences version 9 and higher, use <code>prefDaylightSavingAdjustment</code> instead.                                                 |
| <code>prefRonamaticChar</code>                   | uint16_t                             | The virtual character generated when the user enters the ronamatic stroke. The ronamatic stroke is made by dragging the pen from the input area to the top of the screen.             |

## Preferences

### *SystemPreferencesChoice*

---

| Constant                     | Type     | Description                                                                                                                |
|------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------|
| prefHard1CharAppCreator      | uint32_t | The creator ID of the application to be launched by the left-most hard key (the Date Book button by default).              |
| prefHard2CharAppCreator      | uint32_t | The creator ID of the application to be launched by the second hard key from the left (the Address button by default).     |
| prefHard3CharAppCreator      | uint32_t | The creator ID of the application to be launched by the second hard key from the right (the To Do List button by default). |
| prefHard4CharAppCreator      | uint32_t | The creator ID of the application to be launched by the right-most hard key (the Memo Pad button by default).              |
| prefCalcCharAppCreator       | uint32_t | The creator ID of the application to be launched by the Calculator silk-screen button.                                     |
| prefHardCradleCharAppCreator | uint32_t | The creator ID of the application to be launched by the hard key on the HotSync cradle.                                    |
| prefLauncherAppCreator       | uint32_t | The creator ID of the application to be launched by the status bar icon.                                                   |
| prefSysPrefFlags             |          |                                                                                                                            |

| <b>Constant</b>                            | <b>Type</b>                     | <b>Description</b>                                                                                                                                                                                                                                         |
|--------------------------------------------|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefHardCradle2CharAppCreator</code> | <code>uint32_t</code>           | The creator ID of the application to be launched by the HotSync button on the modem.                                                                                                                                                                       |
| <code>prefAnimationLevel</code>            | <code>AnimationLevelType</code> | Reserved for future use.                                                                                                                                                                                                                                   |
| <code>prefSysSoundVolume</code>            | <code>uint16_t</code>           | The sound level for system sounds, such as taps and beeps. This is a value from 0 to <code>sndMaxAmp</code> .                                                                                                                                              |
| <code>prefGameSoundVolume</code>           | <code>uint16_t</code>           | The sound level for game sounds. This is a value from 0 to <code>sndMaxAmp</code> .                                                                                                                                                                        |
| <code>prefAlarmSoundVolume</code>          | <code>uint16_t</code>           | The sound level for alarms. This is a value from 0 to <code>sndMaxAmp</code> .                                                                                                                                                                             |
| <code>prefBeamReceive</code>               | <code>Boolean</code>            | If <code>true</code> , the device can receive beams from other devices. If <code>false</code> , the device cannot receive beams but can still send them.<br>This preference is not currently used. Instead, use the <a href="#">ExgControl()</a> function. |
| <code>prefCalibrateDigitizerAtReset</code> | <code>Boolean</code>            | If <code>true</code> , the user must recalibrate the digitizer after a soft reset. The default is <code>false</code> .                                                                                                                                     |
| <code>prefSystemKeyboardID</code>          | <code>uint16_t</code>           | The resource ID of the keyboard panel.                                                                                                                                                                                                                     |
| <code>prefDefSerialPlugIn</code>           | <code>uint32_t</code>           | The creator ID of the default serial plug-in database.                                                                                                                                                                                                     |

## Preferences

### *SystemPreferencesChoice*

---

| Constant                               | Type                                  | Description                                                                                                                                                         |
|----------------------------------------|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefStayOnWhenPluggedIn</code>   | Boolean                               | If <code>true</code> , the device stays powered on when it is in the cradle.                                                                                        |
| <code>prefStayLitWhenPluggedIn</code>  | Boolean                               | If <code>true</code> and <code>prefStayOnWhenPluggedIn</code> is <code>true</code> , the device stays lit when it is in its cradle.                                 |
| <code>prefAntennaCharAppCreator</code> | <code>uint32_t</code>                 | The creator ID of the application to launch when the antenna is raised (used only for devices with built-in antennas).                                              |
| <code>prefMeasurementSystem</code>     | <a href="#">MeasurementSystemType</a> | The system of measurement to use.                                                                                                                                   |
| <code>prefShowPrivateRecords</code>    | <a href="#">privateRecordViewEnum</a> | Specifies whether the private records should be displayed, masked, or completely hidden.                                                                            |
| <code>prefAutoOffDurationSecs</code>   | <code>uint16_t</code>                 | Seconds of user idle time before the device powers off. The default value for this preference is specified by the <code>defaultAutoOffDurationSecs</code> constant. |
| <code>prefTimeZone</code>              | <code>int16_t</code>                  | The time zone given as minutes east of Greenwich Mean Time (GMT). Changing the value of this preference does not update the current time.                           |

| <b>Constant</b>                           | <b>Type</b>           | <b>Description</b>                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefDaylightSavingAdjustment</code> | <code>int16_t</code>  | The number of minutes to add to the current time for daylight savings time. Changing the value of this preference does not update the current time.                                                                                                                                                                                                   |
| <code>prefAutoLockType</code>             | Private structure     | Specifies when the auto-locking feature should take effect. Possibilities are upon power off, at a preset time, or after a certain number of seconds.                                                                                                                                                                                                 |
| <code>prefAutoLockTime</code>             | <code>uint32_t</code> | The time value for the auto-locking feature if the system should lock itself after a delay or at a preset time. Depending on the value of <code>prefAutoLockType</code> , this value is either an absolute date and time given as the number of seconds since January 1, 1904 or a time-out value given as a number of seconds from the current time. |
| <code>prefAutoLockTimeFlag</code>         | Boolean               | If <code>true</code> , <code>prefAutoLockTime</code> is given in minutes. If <code>false</code> , the time is given in hours.                                                                                                                                                                                                                         |

## Preferences

### *SystemPreferencesChoice*

---

| Constant                          | Type                      | Description                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>prefLanguage68K</code>      | <code>LanguageType</code> | The language that the device should use. This preference is intended to be used by applications running under PACE—although such applications should really use the <code>prefLocale</code> value instead. Native ARM applications should call <code>LmGetFormatsLocale()</code> to find out what locale the user has selected in the Formats panel, and <code>LmSetFormatsLocale()</code> to change it. |
| <code>prefFormatsLocale68K</code> | <code>LmLocaleType</code> | The device's current locale, which specifies the country and language. This preference is intended to be used by applications running under PACE. Native ARM applications should call <code>LmGetFormatsLocale()</code> to find out what locale the user has selected in the Formats panel, and <code>LmSetFormatsLocale()</code> to change it.                                                          |
| <code>prefTimeZoneCountry</code>  | <code>CountryType</code>  | The country selected to specify what the time zone is.                                                                                                                                                                                                                                                                                                                                                   |

| Constant                    | Type                          | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| prefAttentionFlags          | <a href="#">AttnFlagsType</a> | The user's preferences for receiving attention signals. The returned value is a bit mask that should be tested (using the & operator) with one of the following values:<br>kAttnFlagsUserWantsLED<br>kAttnFlagsUserWantsSound<br>kAttnFlagsUserWantsVibrate<br>kAttnFlagsUserWantsCustomEffect<br>Note that you can override the values in this preference when you make Attention Manager function calls. |
| prefDefaultAppCreator       | uint32_t                      | Creator ID of the application that is launched after a reset. If 0, the system default application is launched.                                                                                                                                                                                                                                                                                            |
| prefDefaultFepPluginCreator | uint32_t                      | Creator ID of the default FEP plug-in.                                                                                                                                                                                                                                                                                                                                                                     |
| prefColorThemeID            | <a href="#">DmResourceID</a>  | Resource ID of the color theme.                                                                                                                                                                                                                                                                                                                                                                            |
| reservedPrefs1              |                               | Reserved for future use.                                                                                                                                                                                                                                                                                                                                                                                   |
| reservedPrefs2              |                               | Reserved for future use.                                                                                                                                                                                                                                                                                                                                                                                   |

**Comments** Most of the system preferences can be set in the Preferences and Security applications. [Table 3.1](#) on page 39 specifies which system preference is set by each user interface field in these two applications.

# Preferences Launch Codes

## prefAppLaunchCmdSetActivePanel

- Purpose**
- Declared In** Preferences.h
- Prototype**

```
#define prefAppLaunchCmdSetActivePanel  
    (sysAppLaunchCmdCustomBase + 1)
```
- Parameters** The launch code's parameter block pointer references a [PrefActivePanelParamsType](#) structure.
- See Also** [Chapter 6, "Common Launch Codes,"](#) in *Exploring Palm OS: Programming Basics*

# Preferences Functions and Macros

## PrefGetAppPreferences Function

- Purpose** Return a copy of an application's preferences resource.
- Declared In** Preferences.h
- Prototype**

```
int16_t PrefGetAppPreferences (uint32_t creator,  
    uint16_t id, void *prefs, uint32_t *prefsSize,  
    Boolean saved)
```
- Parameters**
- *creator*  
Creator ID of the application that owns the preferences.
  - *id*  
ID number of the preferences resource to retrieve. The IDs 0x8000 through 0xFFFF are reserved for system use.
  - *prefs*  
Pointer to a buffer to hold the preferences.
  - *prefsSize*  
Pointer to the size of the *prefs* buffer passed in. (Note that the pointer and the value to which it points *must* be initialized before you call `PrefGetAppPreferences()`.)

Upon return, contains the number of bytes actually written or the number of bytes needed for the *prefs* structure.

→ *saved*

If `true`, retrieve the preferences from the “saved” preferences database, which is backed up during a HotSync operation. If `false`, retrieve the preferences from the “unsaved” preferences database, which is usually not backed up during a HotSync operation.

**Returns** Returns the version number of the retrieved preferences resource, or returns the constant `noPreferenceFound` if the preferences resource wasn’t found. The returned version number is the same version number that was passed to the [PrefSetAppPreferences\(\)](#) function.

**Comments** Use this function to retrieve the preferences that you previously set with the [PrefSetAppPreferences\(\)](#) function. You typically call this function in your `StartApplication()` function upon a normal launch. The values of the *id* and *saved* parameters should be the same as you specified when calling `PrefSetAppPreferences()`, and the *prefs* parameter should be a structure of the same type as you passed to `PrefSetAppPreferences()`. Most applications store all preferences in a single preferences resource retrieved by a single call to `PrefGetAppPreferences()`, but this is not required. You can use multiple preferences resources if you wish.

To determine the required size for the *prefs* structure, set *prefsSize* to 0 and pass `NULL` for *prefs*. Upon return, the *prefsSize* parameter contains the required size. Never set *prefs* to `NULL` without also setting *prefsSize* to 0. After allocating the required amount of memory to obtain a copy of the application’s preferences resource, then call `PrefGetAppPreferences()` a second time to actually obtain the preference values.

---

**NOTE:** Always compare the value returned in the *prefsSize* parameter with the value you passed in. If the two values differ, you need to resize the *prefs* structure and call this function again.

---

The version number returned by this function allows you to handle the case where a new version of the application is being run for the

## Preferences

### *PrefGetPreference*

---

first time. You can compare the value returned by this function with the current version number to determine if you need to set default values for any preferences created by the current release. For more information, see “[Updating Preferences Upon a New Release](#)” on page 45.

**See Also** [PrefGetPreference\(\)](#), [PrefSetAppPreferences\(\)](#)

## PrefGetPreference Function

**Purpose** Return a system preference.

**Declared In** `Preferences.h`

**Prototype** `uint32_t PrefGetPreference  
(SystemPreferencesChoice choice)`

**Parameters** → *choice*  
A constant that specifies what preference to retrieve. See [SystemPreferencesChoice](#).

**Returns** Returns the system preference or 0 if the preference could not be found. On debug ROMs, a non-fatal error message is also displayed if the specified preference cannot be found.

**See Also** [PrefGetAppPreferences\(\)](#), [PrefSetPreference\(\)](#)

## PrefSetAppPreferences Function

**Purpose** Set an application’s preferences in the specified preferences database.

**Declared In** `Preferences.h`

**Prototype** `void PrefSetAppPreferences (uint32_t creator,  
uint16_t id, int16_t version,  
const void *prefs, uint32_t prefsSize,  
Boolean saved)`

**Parameters** → *creator*  
Creator ID of the application that owns this preference.

→ *id*

ID number of the preference to set. An application can have multiple preferences. The IDs 0x8000 through 0xFFFF are reserved for system use.

→ *version*

Version number of the application's preferences.

→ *prefs*

Pointer to a buffer that holds the current value of the preferences structure. Pass NULL if you want to delete the preferences.

→ *prefsSize*

Size of the buffer passed. Pass 0 if you want to delete the preferences structure.

→ *saved*

If `true`, saves the preferences in the "saved" preferences database. If not, saves the preferences in the "unsaved" preferences database.

**Returns** Returns nothing.

**Comments** You typically call this function in your `StopApplication()` function to save the current state of the application.

The "saved" preferences database is backed up when a user performs the HotSync operation. The "unsaved" preferences database is not backed up by default. (The user can use a third-party tool to set the backup bit in the "unsaved" preferences database, which would cause it to be backed up.) Both the "saved" and the "unsaved" preferences reside in the storage heap and thus persist across soft resets. The only way that preferences are lost is if a hard reset is performed. "[Which Preferences Database to Use](#)" on page 43 describes how to choose between the "saved" and "unsaved" preferences databases.

The version number that you pass as the *version* parameter is returned when the preferences are retrieved by [PrefGetAppPreferences\(\)](#). You can use this version number to determine if a new release of the application is being run for the first time. For more information, see "[Updating Preferences Upon a New Release](#)" on page 45.

**See Also** [PrefGetAppPreferences\(\)](#), [PrefSetPreference\(\)](#)

## Preferences

### *PrefSetPreference*

---

## PrefSetPreference Function

- Purpose** Set a system preference.
- Declared In** `Preferences.h`
- Prototype**  
`void PrefSetPreference  
    (SystemPreferencesChoice choice,  
    uint32_t value)`
- Parameters**
- *choice*  
A [SystemPreferencesChoice](#) constant specifying the preference to be set.
  - *value*  
Value to assign to the preference.
- Returns** Returns nothing. If the specified preference cannot be found, displays a non-fatal error message on debug ROMs. On release ROMs, this function fails silently.
- See Also** [PrefGetPreference\(\)](#), [PrefSetAppPreferences\(\)](#)

# Sync Manager

---

The Sync Manager provides functions that allow a sync application—with the user’s permission—to gain access to secure databases that have the sync bypass rule set.

This chapter is organized as follows:

|                                                             |     |
|-------------------------------------------------------------|-----|
| <a href="#">Sync Manager Constants</a> . . . . .            | 421 |
| <a href="#">Sync Manager Functions and Macros</a> . . . . . | 423 |

The header file `SyncMgr.h` declares the API that this chapter describes.

## Sync Manager Constants

### Sync Manager Error Codes

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Error codes returned by the various Sync Manager functions.                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Declared In</b> | <code>SyncMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Constants</b>   | <pre>#define syncMgrErrAccessDenied (syncMgrErrorClass   0x04)     Access was denied. Either the user refused to authorize     registration of the sync client, or an unregistered sync client     is attempting to get or release access  #define syncMgrErrMaxSessionsActive (syncMgrErrorClass   0x06)  #define syncMgrErrMemAllocFailure (syncMgrErrorClass   0x03)  #define syncMgrErrOperationNotSupported (syncMgrErrorClass   0x01)</pre> |

## Sync Manager

### Sync Manager Security Policies

---

```
#define syncMgrErrSystemErr (syncMgrErrorClass |  
    0x02)  
    A system error occurred.  
#define syncMgrErrUserRefusedSyncApp  
    (syncMgrErrorClass | 0x05)
```

## Sync Manager Security Policies

|                    |                                                                                                                                                                                                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Security policies enforced by the Sync Manager.                                                                                                                                                                                                                                                                                       |
| <b>Declared In</b> | SyncMgr.h                                                                                                                                                                                                                                                                                                                             |
| <b>Constants</b>   | <pre>#define kSyncPolicyRestrictAppRegistration 'sync'<br/>    Limits the set of applications that are allowed to register to<br/>    those specified in in the security policy.<br/><br/>#define kSyncPolicyNonUiAuthentication 'noui'<br/>    Allows specified applications to register non-intrusively<br/>    (without UI).</pre> |

## Miscellaneous Sync Manager Constants

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | The Sync Manager also defines these constants.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Declared In</b> | SyncMgr.h                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Constants</b>   | <pre>#define kSyncAppDescriptionMaxLen (95 + 1)<br/>    The maximum length in bytes—including the null<br/>    terminator—of the product description supplied to<br/>    <a href="#">SyncAddSynchronizer()</a>.<br/><br/>#define kSyncMaxActiveSessions (32)<br/><br/>#define kSyncProductNameMaxLen (43 + 1)<br/>    The maximum length in bytes—including the null<br/>    terminator—of the product display name supplied to<br/>    <a href="#">SyncAddSynchronizer()</a>.</pre> |

## Sync Manager Functions and Macros

### SyncAddSynchronizer Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Ask the user for permission to register a sync application with the Sync Manager and, if permission is granted, register that application.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Declared In</b> | <code>SyncMgr.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Prototype</b>   | <pre>status_t SyncAddSynchronizer     (const char *displayNameP,      const char *descriptionP)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Parameters</b>  | <p>→ <i>displayNameP</i><br/>Pointer to the display name of the product. This string is displayed to the user. This string should be no more than <code>kSyncProductNameMaxLen</code> bytes long, including the null terminator.</p> <p>→ <i>descriptionP</i><br/>Pointer to a brief description of the product, the vendor, or both that is to be displayed to the user. Supply <code>NULL</code> for this parameter if no description is needed. This string should be no more than <code>kSyncAppDescriptionMaxLen</code> bytes long, including the null terminator.</p> |
| <b>Returns</b>     | Returns <code>errNone</code> if the sync application was successfully registered, or <code>syncMgrErrAccessDenied</code> if there was an error.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Comments</b>    | <p>A sync application can synchronize or back up secure databases only if it is registered with the Sync Manager.</p> <p>The sync client should call this function once after it is installed on the handheld. This function prompts the user to approve the sync client by entering the system password. If the user is unable or unwilling to approve the sync client, this function returns <code>syncMgrErrAccessDenied</code>.</p>                                                                                                                                     |

## Sync Manager

*SyncSessionGetAccess*

---

**Comments** **IMPORTANT:** When called from the main application thread, this function may block. While blocked, the application will not receive events and won't redraw its windows. As well, deferred sublaunches and notifications won't execute while the main application thread is blocked.

---

**See Also** [SyncSessionGetAccess\(\)](#), [SyncSessionReleaseAccess\(\)](#)

### SyncSessionGetAccess Function

**Purpose** Get access to secure databases for a sync session.

**Declared In** `SyncMgr.h`

**Prototype** `status_t SyncSessionGetAccess (void)`

**Parameters** None.

**Returns** Returns `errNone` if the Sync Manager was able to add the sync client's token to the global bypass rule, or `syncMgrErrAccessDenied` if not.

**Comments** The sync client must be registered prior to this call.  
The sync client must call this function once for every sync session. After the sync client has accessed the desired secure databases, it should call [SyncSessionReleaseAccess\(\)](#).

**See Also** [SyncAddSynchronizer\(\)](#)

### SyncSessionReleaseAccess Function

**Purpose** Signal that the sync client no longer needs access to any secure databases.

**Declared In** `SyncMgr.h`

**Prototype** `status_t SyncSessionReleaseAccess (void)`

**Parameters** None.

**Returns** Returns `errNone` as long as the sync client is registered. Otherwise, returns `syncMgrErrAccessDenied`.

- Comments** The sync client must call this function once for every sync session. Prior to accessing any secure databases it should call [SyncSessionGetAccess\(\)](#). Once it is done accessing secure databases, it should call `SyncSessionReleaseAccess()`.
- See Also** [SyncAddSynchronizer\(\)](#)

## **Sync Manager**

*SyncSessionReleaseAccess*

---

# System Manager

---

The System Manager APIs cover a wide range of topics, including features (the system feature constants are defined here), processor types, power management, ROM version and serial numbers, display brightness and contrast and system time intervals (the System Manager defines a number of macros useful for manipulating system time “ticks”). The reference material in this chapter is organized as follows:

|                                                               |     |
|---------------------------------------------------------------|-----|
| <a href="#">System Manager Constants</a> . . . . .            | 427 |
| <a href="#">System Manager Functions and Macros</a> . . . . . | 436 |

The header file `SystemMgr.h` declares the API that this chapter describes.

Additional information on the material covered in this chapter can be found in the conceptual chapters in the first part of this book. In particular, see [Chapter 2, “Features,”](#) [Chapter 9, “Power Management,”](#) and [Chapter 10, “The ROM Serial Number.”](#)

## System Manager Constants

### Power Manager Error Codes

|                    |                                                                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | These error codes can be returned by various functions to indicate a lack of power.                                                       |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                                                  |
| <b>Constants</b>   | <pre>#define pwrErrBacklight (pwrErrorClass   1)  #define pwrErrBeam (pwrErrorClass   3)  #define pwrErrGeneric (pwrErrorClass   4)</pre> |

## System Manager

### System Features

---

```
#define pwrErrNone (pwrErrorClass | 0)
```

```
#define pwrErrRadio (pwrErrorClass | 2)
```

## System Features

- Purpose** Feature constants are defined by various parts of Palm OS to identify how the system works.
- Declared In** `SystemMgr.h`
- Constants**
- ```
#define sysFtrDefaultBoldFont 13
```
- The FontID of the default font used for bold text as specified in the ROM's locale module.
- ```
#define sysFtrDefaultFont 12
```
- The FontID of the default font for standard text, as defined in the ROM's locale module.
- ```
#define sysFtrNumAccessorTrapPresent 25
```
- ```
#define sysFtrNumBacklight 3
```
- A nonzero value indicates that the device has a backlight.
- ```
#define sysFtrNumCharEncodingFlags68K 16
```
- One or more of the character encoding feature attributes (declared in `TextMgr.h`), which specify the attributes of the character encoding used on the device. For example, these constants specify if the device uses only single-byte characters or has double-byte characters as well.
- ```
#define sysFtrNumCountry68K 5
```
- One of the country constants (defined in `PalmLocale.h`), identifying the default country as specified in the ROM's locale module.
- ```
#define sysFtrNumDefaultCompression 23
```
- The default compression algorithm used for wireless networking. The Palm Web Clipping Application checks this feature. Note that the Palm Web Clipping Application is obsolete.
- ```
#define sysFtrNumDisableSortDuringSyncThreshold 33
```
- If this feature is not set, database sorting is always disabled during a HotSync operation on Palm OS Cobalt version 6.1

(or later). If this feature is set, and the value is zero, database sorting is enabled during a HotSync operation. If this feature is set and the value is greater than zero, sorting during HotSync is enabled up until a database with a number of rows greater than or equal to the value of this feature is encountered, at which point sorting is disabled. Sorting is then disabled from that point onwards.

This feature can be set by licensees to optimize HotSync performance on their Palm Powered devices.

```
#define sysFtrNumDisplayDepth 7
```

The device's default display depth. Supported depths are 1, 2, 4, 8, and 16 bits per pixel.

```
#define sysFtrNumDisplayUpdateMode 30
```

Indicates how the display is updated. If this feature has a value of 0, or it is not defined, drawing occurs directly to the screen (which allows the screen to flicker). If it is set to 1, the display uses double buffering (page flipping) to eliminate flicker when drawing.

---

**NOTE:** Additional values may be defined in the future for other methods of eliminating flicker. Regardless, a value of 0 always means that drawing in update-based windows causes flickering.

---

```
#define sysFtrNumDmAutoBackup 31
```

Indicates that the device has the Automatic Database Backup and Restore feature, which preserves device data even without a backup battery.

```
#define sysFtrNumEnableSortAfterSyncThreshold 34
```

If this feature is not set, databases are not sorted when closed during a HotSync operation on Palm OS Cobalt version 6.1 (or later). If this feature is set, and the value of the feature is zero, during a HotSync operation any database that has not already been sorted is sorted when the database is closed. If this feature is set, and the value of the feature is greater than zero, during a HotSync operation any database that has not been sorted and that has a number of rows that equals or exceeds the value of this feature is sorted when the database is closed.

This feature can be set by licensees to optimize HotSync performance on their Palm Powered devices.

## System Manager

### System Features

---

```
#define sysFtrNumEncoding68K 11
    One of the character encoding constants identifying the
    character encoding used on the device.

#define sysFtrNumEncryption 4
    One or more flags indicating which encryption schemes are
    present. The only currently supported encryption is DES,
    which can be tested for by ANDing the returned value with
    sysFtrNumEncryptionMaskDES.

#define sysFtrNumFastBoot 29
    Enable fast minimal boot. When non-zero, various parts of
    the boot process are be shortened or completely skipped.
    This mode is only for development, to reduce the amount of
    waiting required each time a new build is tested.

#define sysFtrNumFiveWayNavVersion 32

#define sysFtrNumHwrMiscFlags 8
    One or more flags indicating hardware capabilities.
    This feature is not applicable on devices with ARM
    processors.

#define sysFtrNumHwrMiscFlagsExt 9
    One or more flags indicating additional hardware
    capabilities.
    This feature is not applicable on devices with ARM
    processors.

#define sysFtrNumInputAreaFlags 26
    Indicates the device-specific capabilities of the input area. See
    "Input Area Flags Constants" on page 75 for the flags that
    make up this feature.

#define sysFtrNumLanguage68K 6
    One of the language constants (defined in PalmLocale.h)
    specifying the default language in the ROM's locale module.

#define sysFtrNumNotifyMgrVersion 17
    The version number of the Notification Manager API.

#define sysFtrNumOEMCompanyID 20
    The 4-character company ID of the HAL manufacturer. The
    company ID is unique for each Palm OS licensee.
```

```
#define sysFtrNumOEMDeviceID 21
    The 4-character ID of the device on which Palm OS is
    running. There is roughly one device ID per model of device.

#define sysFtrNumOEMHALID 22
    The 4-character ID of the HAL on which Palm OS is running.
    Each HAL is specific to a device model and Palm OS version.

#define sysFtrNumOEMROMVersion 18
    A ROM system version provided by the Palm OS licensee.
    Used to identify patches provided by licensees.

#define sysFtrNumProcessorID 2
    The processor type and the processor revision.

#define sysFtrNumProductID sysFtrNumProcessorID
    The processor type and the processor revision.

#define sysFtrNumResetType 28

#define sysFtrNumROMBuildType 19

#define sysFtrNumROMVersion 1
    The ROM version. You can use the sysMakeROMVersion\(\)
    macro to create a value to test against.

#define sysFtrNumSkipCalibration 35

#define sysFtrNumTextMgrFlags 10

#define sysFtrNumUIHardwareFlags 27

#define sysFtrNumVendor 15
    Not used.

#define sysFtrNumWinVersion 24
    The version of the Window Manager.
```

**Comments**

You can obtain the values of these features with the [FtrGet\(\)](#) function:

---

```
err = FtrGet(sysFileCSystem, constant, &value)
```

---

## System Manager

### Processor Types

---

where *constant* is one of the values listed above. If the feature is defined, value is set by the `FtrGet ( )` function. If the feature is not defined, value is not set, and `FtrGet ( )` returns `sysErrNoSuchFeature`. Unless otherwise specified, you should consider these features to be read-only.

## Processor Types

**Purpose** Palm OS device processor types. Obtain a device's processor type by getting the value of the `sysFtrNumProcessorID` feature. Use the the bit mask `sysFtrNumProcessorMask` to extract the processor type from the values returned for these features.

**Declared In** `SystemMgr.h`

**Constants**

```
#define sysFtrNumProcessor328 0x00010000
    Motorola 68328 (Dragonball).

#define sysFtrNumProcessorARM710A 0x00170000
    ARM710A.

#define sysFtrNumProcessorARM720T 0x00100000
    ARM 720T.

#define sysFtrNumProcessorARM7TDMI 0x00110000
    ARM7TDMI.

#define sysFtrNumProcessorARM920T 0x00120000
    ARM920T.

#define sysFtrNumProcessorARM922T 0x00130000
    ARM922T.

#define sysFtrNumProcessorARM925 0x00140000
    ARM925.

#define sysFtrNumProcessorEZ 0x00020000
    Motorola 68EZ328 (Dragonball EZ).

#define sysFtrNumProcessorStrongARM 0x00150000
    Strong ARM.

#define sysFtrNumProcessorSuperVZ 0x00040000
    Motorola 68SZ328 (Dragonball SuperVZ).

#define sysFtrNumProcessorVZ 0x00030000
    Motorola 68VZ328 (Dragonball VZ)
```

```
#define sysFtrNumProcessorx86 0x01000000
    The Palm OS ARM Simulator, which runs on Windows.
#define sysFtrNumProcessorXscale 0x00160000
    X-scale.
```

## Processor Masks

- Purpose** These mask values allow you to extract the processor type from a `sysFtrNumProcessorID` feature value, and to easily determine if that processor is an ARM processor or a 68K-family processor.
- Declared In** `SystemMgr.h`
- Constants**
- ```
#define sysFtrNumProcessor68KIfZero 0xFFF00000
    The processor is a 68K-family processor if, after ANDing this
    mask with the sysFtrNumProcessorID feature value, the
    result is zero.

#define sysFtrNumProcessorARMIfNotZero 0x00F00000
    The processor is an ARM processor if, after ANDing this
    mask with the sysFtrNumProcessorID feature value, the
    result is not zero.

#define sysFtrNumProcessorMask 0xFFFF0000
    AND this mask with the sysFtrNumProcessorID feature
    value to obtain a processor type that can be compared with
    the values listed under "Processor Types" on page 432.
```

## Build Stages

- Purpose** Build stage values used when constructing a ROM version number or when checking the build stage of a device's ROM.
- Declared In** `SystemMgr.h`
- Constants**
- ```
#define sysROMStageAlpha (1)
    An alpha ROM.

#define sysROMStageBeta (2)
    A beta ROM.

#define sysROMStageDevelopment (0)
    A ROM that is still in development (pre-release).
```

## System Manager

### ROM Tokens

---

```
#define sysROMStageRelease (3)
    A release ROM.
```

**Comments** Use [sysMakeROMVersion\(\)](#) to construct a ROM version number. Use [sysGetROMVerBuild\(\)](#) to extract the build stage from a device's ROM version number.

## ROM Tokens

**Purpose** ROM token identifiers that, when supplied to [SysGetROMToken\(\)](#), cause the corresponding ROM token value to be retrieved.

**Declared In** SystemMgr.h

**Constants**

```
#define sysROMTokenSnum 'snum'
```

 Used to retrieve the ROM serial number, expressed as a text string with no null terminator.

**Comments** The serial number is shown to the user in the Application Launcher, along with a checksum digit you can use to validate input when your users read the ID from their device and type it in or tell it to someone else. [Chapter 10, "The ROM Serial Number,"](#) on page 105 shows how to retrieve the ROM serial number and calculate its associated checksum.

## Device Manufacturers

**Purpose** Identifies the manufacturer, HAL ID, and device ID of a Palm Powered™ device.

**Declared In** SystemMgr.h

**Constants**

```
#define sysOEMCompanyIDHandspring 'hspr'
    Handspring.

#define sysOEMCompanyIDPalmDevices 'palm'
    Palm, Inc.

#define sysOEMCompanyIDPalmPlatform 'psys'
    PalmSource, Inc.

#define sysOEMCompanyIDQualcomm 'qcom'
    Qualcomm.
```

```
#define sysOEMCompanyIDSymbol 'smb1'  
    Symbol.  
  
#define sysOEMCompanyIDTRG 'trgp'  
    TRG.  
  
#define sysOEMCompanyIDUnspecified 0x00000000  
    The company is unspecified.  
  
#define sysOEMDeviceIDUnspecified 0x00000000  
    The device is unspecified.  
  
#define sysOEMHALIDUnspecified 0x00000000  
    The HAL (hardware layer) is unspecified.
```

**Comments** These values are assigned by PalmSource's Platform Engineering group. Note that these values differ from those found in some devices which use ROM tokens and run versions of Palm OS prior to 3.5.

## Miscellaneous System Manager Constants

**Purpose** The System Manager also defines these constants.

**Declared In** SystemMgr.h

**Constants**

```
#define sysFtrNumEncryptionMaskDES 0x00000001  
    AND this value with the value of the  
    sysFtrNumEncryption feature; if the result is nonzero, the  
    encryption scheme is DES.  
  
#define sysEntryNumMain ((uint32_t)0xffffffff)  
    The main entry point in an executable.  
  
#define sysFileDescStdIn 0  
    The "stdin" file descriptor.  
  
#define sysFtrCreator sysFileCSystem  
    Creator ID for those features defined by the System Manager.  
    Supply sysFtrCreator or sysFileCSystem to  
    FtrGet\(\) when obtaining the value of those features listed  
    under "System Features" on page 428.
```

# System Manager Functions and Macros

## SysBatteryInfo Function

**Purpose** Retrieve settings for the batteries. Set the *set* parameter to *false* to retrieve battery settings. (Applications should *not* change any of the settings).

---

**WARNING!** Use this function only to *retrieve* settings!

---

**Declared In** `SystemMgr.h`

**Prototype**

```
uint16_t SysBatteryInfo (Boolean set,
    uint16_t *warnThresholdPercentP,
    uint16_t *criticalThresholdPercentP,
    uint16_t *shutdownThresholdPercentP,
    uint32_t *maxMilliSecsP,
    SysBatteryKind *kindP, Boolean *pluggedInP,
    uint8_t *percentP)
```

**Parameters**

- *set*  
If *false*, parameters with non-NULL pointers are retrieved. Never set this parameter to *true*.
- ↔ *warnThresholdPercentP*  
Pointer to battery voltage warning threshold in volts\*100, or NULL.
- ↔ *criticalThresholdPercentP*  
Pointer to the battery voltage critical threshold in volts\*100, or NULL.
- ↔ *shutdownThresholdPercentP*  
Pointer to the battery voltage threshold at which the device will shut down, in volts\*100, or NULL.
- ↔ *maxMilliSecsP*  
Pointer to the battery timeout, or NULL.
- ↔ *kindP*  
Pointer to the battery type, or NULL. For a complete set of battery types, see “[SysBatteryKindTag](#)” on page 166.
- ↔ *pluggedInP*  
Pointer to pluggedIn return value, or NULL.

↔ *percentP*

Percentage of power remaining in the battery.

**Returns** Returns the current battery voltage in volts\*100.

**Comments** Call this function to make sure an upcoming activity won't be interrupted by a low battery warning.

*warnThresholdP* and *maxTicksP* are the battery-warning voltage threshold and time out. If the battery voltage falls below the threshold, or the timeout expires, a `lowBatteryChr` key event is put on the queue. Normally, applications call [SysHandleEvent\(\)](#) which calls `SysBatteryDialog` in response to this event.

*criticalThresholdP* is the battery voltage threshold. If battery voltage falls below this level, the system turns itself off without warning and doesn't turn on until battery voltage is above it again.

## **sysFtrNumProcessorIs68K Macro**

**Purpose** Determines whether or not the underlying processor is part of the 68K family.

**Declared In** `SystemMgr.h`

**Prototype** `#define sysFtrNumProcessorIs68K (x)`

**Parameters** → *x*

Processor type obtained from a call to [FtrGet\(\)](#).

**Returns** Returns `true` if the underlying processor is a 68K, `false` otherwise.

**Example** `UInt32 processorType;`

```
FtrGet(sysFileCSystem, sysFtrNumProcessorID, &processorType);
if (sysFtrNumProcessorIs68K(processorType)){
    // processor is 68K
} else {
    // processor is not 68K
}
```

---

### **sysFtrNumProcessorIsARM Macro**

- Purpose** Determines whether or not the underlying processor is part of the ARM family.
- Declared In** `SystemMgr.h`
- Prototype** `#define sysFtrNumProcessorIsARM (x)`
- Parameters**  $\rightarrow x$   
Processor type obtained from a call to [FtrGet\(\)](#).
- Returns** Returns `true` if the underlying processor is an ARM core, `false` otherwise.

---

**Example**

```
UInt32 processorType;

FtrGet(sysFileCSystem, sysFtrNumProcessorID, &processorType);
if (sysFtrNumProcessorIsARM(processorType)){
    // processor is ARM
} else {
    // processor is not ARM
}
```

---

### **SysGetROMToken Function**

- Purpose** Return from the ROM a value specified by token.
- Declared In** `SystemMgr.h`
- Prototype** `status_t SysGetROMToken (uint32_t token, uint8_t **dataP, uint16_t *sizeP)`
- Parameters**  $\rightarrow token$   
The value to retrieve, as specified by one of the tokens listed under “[ROM Tokens](#)” on page 434.
- $\leftarrow dataP$   
Pointer to a buffer that holds the requested value when the function returns.
- $\leftarrow sizeP$   
The number of bytes in the `dataP` buffer.
- Returns** Returns `errNone` if the requested token was successfully retrieved, or an error code if an error occurred. If this function returns an error,

or if the returned pointer to the buffer is NULL or if the first byte of the text buffer is 0xFF, then no serial number is available.

## **sysGetROMVerBuild Macro**

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Extract the build stage from a ROM version number.                                                                                                                                              |
| <b>Declared In</b> | SystemMgr.h                                                                                                                                                                                     |
| <b>Prototype</b>   | <pre>#define sysGetROMVerBuild (dwROMVer)</pre>                                                                                                                                                 |
| <b>Parameters</b>  | → <i>dwROMVer</i><br>The ROM version number.                                                                                                                                                    |
| <b>Returns</b>     | Returns the build stage. See “ <a href="#">Build Stages</a> ” on page 433 for the predefined set of build stage values.                                                                         |
| <b>Comments</b>    | Obtain the ROM version number by calling <a href="#">FtrGet()</a> with a creator ID of <code>sysFtrCreator</code> and a feature number of <code>sysFtrNumROMVersion</code> .                    |
| <b>See Also</b>    | <a href="#">sysGetROMVerFix()</a> , <a href="#">sysGetROMVerMajor()</a> ,<br><a href="#">sysGetROMVerMinor()</a> , <a href="#">sysGetROMVerStage()</a> ,<br><a href="#">sysMakeROMVersion()</a> |

## **sysGetROMVerFix Macro**

|                    |                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Extract the fix number from a ROM version number.                                                                                                                                                 |
| <b>Declared In</b> | SystemMgr.h                                                                                                                                                                                       |
| <b>Prototype</b>   | <pre>#define sysGetROMVerFix (dwROMVer)</pre>                                                                                                                                                     |
| <b>Parameters</b>  | → <i>dwROMVer</i><br>The ROM version number.                                                                                                                                                      |
| <b>Returns</b>     | Returns the fix number.                                                                                                                                                                           |
| <b>Comments</b>    | Obtain the ROM version number by calling <a href="#">FtrGet()</a> with a creator ID of <code>sysFtrCreator</code> and a feature number of <code>sysFtrNumROMVersion</code> .                      |
| <b>See Also</b>    | <a href="#">sysGetROMVerBuild()</a> , <a href="#">sysGetROMVerMajor()</a> ,<br><a href="#">sysGetROMVerMinor()</a> , <a href="#">sysGetROMVerStage()</a> ,<br><a href="#">sysMakeROMVersion()</a> |

### **sysGetROMVerMajor Macro**

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Extract the major version number from a ROM version number.                                                                                                                                     |
| <b>Declared In</b> | SystemMgr.h                                                                                                                                                                                     |
| <b>Prototype</b>   | <pre>#define sysGetROMVerMajor (dwROMVer)</pre>                                                                                                                                                 |
| <b>Parameters</b>  | → <i>dwROMVer</i><br>The ROM version number.                                                                                                                                                    |
| <b>Returns</b>     | Returns the major version number.                                                                                                                                                               |
| <b>Comments</b>    | Obtain the ROM version number by calling <a href="#">FtrGet()</a> with a creator ID of <code>sysFtrCreator</code> and a feature number of <code>sysFtrNumROMVersion</code> .                    |
| <b>See Also</b>    | <a href="#">sysGetROMVerBuild()</a> , <a href="#">sysGetROMVerFix()</a> ,<br><a href="#">sysGetROMVerMinor()</a> , <a href="#">sysGetROMVerStage()</a> ,<br><a href="#">sysMakeROMVersion()</a> |

### **sysGetROMVerMinor Macro**

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Extract the minor version number from a ROM version number.                                                                                                                                     |
| <b>Declared In</b> | SystemMgr.h                                                                                                                                                                                     |
| <b>Prototype</b>   | <pre>#define sysGetROMVerMinor (dwROMVer)</pre>                                                                                                                                                 |
| <b>Parameters</b>  | → <i>dwROMVer</i><br>The ROM version number.                                                                                                                                                    |
| <b>Returns</b>     | Returns the minor version number.                                                                                                                                                               |
| <b>Comments</b>    | Obtain the ROM version number by calling <a href="#">FtrGet()</a> with a creator ID of <code>sysFtrCreator</code> and a feature number of <code>sysFtrNumROMVersion</code> .                    |
| <b>See Also</b>    | <a href="#">sysGetROMVerBuild()</a> , <a href="#">sysGetROMVerFix()</a> ,<br><a href="#">sysGetROMVerMajor()</a> , <a href="#">sysGetROMVerStage()</a> ,<br><a href="#">sysMakeROMVersion()</a> |

## **sysGetROMVerStage Macro**

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Extract the build stage from a ROM version number.                                                                                                                                              |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                                                                                                        |
| <b>Prototype</b>   | <code>#define sysGetROMVerStage (dwROMVer)</code>                                                                                                                                               |
| <b>Parameters</b>  | <code>→ dwROMVer</code><br>The ROM version number.                                                                                                                                              |
| <b>Returns</b>     | Returns the build stage.                                                                                                                                                                        |
| <b>Comments</b>    | Obtain the ROM version number by calling <a href="#">FtrGet()</a> with a creator ID of <code>sysFtrCreator</code> and a feature number of <code>sysFtrNumROMVersion</code> .                    |
| <b>See Also</b>    | <a href="#">sysGetROMVerBuild()</a> , <a href="#">sysGetROMVerFix()</a> ,<br><a href="#">sysGetROMVerMajor()</a> , <a href="#">sysGetROMVerMinor()</a> ,<br><a href="#">sysMakeROMVersion()</a> |

## **SysHandleEvent Function**

|                    |                                                                                                                                                                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Handle defaults for system events such as hard and soft key presses.                                                                                                                                                                                    |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                                                                                                                                                                |
| <b>Prototype</b>   | <code>Boolean SysHandleEvent (EventPtr eventP)</code>                                                                                                                                                                                                   |
| <b>Parameters</b>  | <code>→ eventP</code><br>Pointer to an event.                                                                                                                                                                                                           |
| <b>Returns</b>     | Returns <code>true</code> if the system handled the event.                                                                                                                                                                                              |
| <b>Comments</b>    | Applications should call this routine immediately after calling <a href="#">EvtGetEvent()</a> unless they want to override the default system behavior. However, overriding the default system behavior is almost never appropriate for an application. |
| <b>See Also</b>    | <code>KeyRates()</code>                                                                                                                                                                                                                                 |

## SysLCDBrightness Function

- Purpose** Get or set the LCD's brightness level.
- Declared In** `SystemMgr.h`
- Prototype** `uint8_t SysLCDBrightness (Boolean set, uint8_t newBrightnessLevel)`
- Parameters**
- *set*  
Pass true to set the brightness level, false to retrieve it.
  - *newBrightnessLevel*  
The desired new brightness level. This parameter is ignored if *set* is false.
- Returns** If *set* is true, the previous brightness level. If *set* is false, the current brightness level.
- See Also** [SysLCDContrast\(\)](#)

## SysLCDContrast Function

- Purpose** Get or set the LCD's contrast level.
- Declared In** `SystemMgr.h`
- Prototype** `uint8_t SysLCDContrast (Boolean set, uint8_t newContrastLevel)`
- Parameters**
- *set*  
Pass true to set the contrast level, false to retrieve it.
  - *newContrastLevel*  
The desired new contrast level. This parameter is ignored if *set* is false.
- Returns** If *set* is true, the previous contrast level. If *set* is false, the current contrast level.
- See Also** [SysLCDBrightness\(\)](#)

## sysMakeROMVersion Macro

- Purpose** Build a ROM version value from the major, minor, fix, stage, and build numbers.
- Declared In** `SystemMgr.h`
- Prototype** `#define sysMakeROMVersion (major, minor, fix, stage, buildNum)`
- Parameters**
- *major*  
The major version number.
  - *minor*  
The minor version number.
  - *fix*  
The fix number.
  - *stage*  
The build stage. See “[Build Stages](#)” on page 433 for the set of predefined build stage values.
  - *buildNum*  
The build number.
- Returns** This macro produces a `uint32_t` that contains the ROM version value.
- See Also** [sysGetROMVerBuild\(\)](#), [sysGetROMVerFix\(\)](#), [sysGetROMVerMajor\(\)](#), [sysGetROMVerMinor\(\)](#), [sysGetROMVerStage\(\)](#)

## SysRequestSleep Function

- Purpose** Request that the system be put to sleep.
- Declared In** `SystemMgr.h`
- Prototype** `void SysRequestSleep (void)`
- Parameters** None.
- Returns** Nothing.

## System Manager

### *SysSetAutoOffTime*

---

- Comments** Unlike [SysSleep\(\)](#), this function sends out a sleep request notification that allows any executable registered for the notification to prevent the system from going to sleep.
- See Also** [SysSleep\(\)](#)

## SysSetAutoOffTime Function

- Purpose** Set the time out value in seconds for auto-power-off.
- Declared In** `SystemMgr.h`
- Prototype** `uint16_t SysSetAutoOffTime (uint16_t seconds)`
- Parameters** `→ seconds`  
Time out in seconds, or 0 for no time out.
- Returns** Returns the previous time out value, in seconds.

## SysSleep Function

- Purpose** Put the system into lowest power mode by shutting down all peripherals, the CPU, and the system clock.
- Declared In** `SystemMgr.h`
- Prototype** `void SysSleep (void)`
- Parameters** None.
- Returns** Nothing.
- See Also** [SysRequestSleep\(\)](#)

## SysTaskDelay Function

- Purpose** Put the processor into doze mode for the specified number of milliseconds.
- Declared In** `SystemMgr.h`
- Prototype** `status_t SysTaskDelay (int32_t delayInMilliSecs)`
- Parameters** `→ delayInMilliSecs`  
Amount of time to wait, in milliseconds.

**Returns** Returns `errNone` if no error occurred.

**See Also** `EvtGetEvent()`

## **SysTicksPerSecond Macro**

**Purpose** Return the number of ticks per second. This routine allows applications to be tolerant of changes to the ticks per second rate in the system.

**Declared In** `SystemMgr.h`

**Prototype** `#define SysTicksPerSecond ()`

**Parameters** None.

**Returns** Evaluates to the number of ticks per second.

**Comments** Applications should not be written to measure time in system ticks. Instead, use the various `SysTimeIn...` and `SysTimeTo...` macros to work with time values in system-independent units.

## **SysTimeInCentiSecs Macro**

**Purpose** Create a system time value from a value in centiseconds (1/100 second).

**Declared In** `SystemMgr.h`

**Prototype** `#define SysTimeInCentiSecs (centiSecs)`

**Parameters** `→ centiSecs`

The time, in hundredths of a second.

**Returns** Evaluates to the corresponding system time value.

**See Also** [`SysTimeInMicroSecs\(\)`](#), [`SysTimeInMilliSecs\(\)`](#),  
[`SysTimeInMins\(\)`](#), [`SysTimeInSecs\(\)`](#)

### **SysTimeInMicroSecs Macro**

|                    |                                                                                                                                                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Create a system time value from a value in microseconds.                                                                                                                                     |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                                                                                                     |
| <b>Prototype</b>   | <code>#define SysTimeInMicroSecs (<i>microSecs</i>)</code>                                                                                                                                   |
| <b>Parameters</b>  | <code>→ <i>microSecs</i></code><br>The time, in microseconds.                                                                                                                                |
| <b>Returns</b>     | Evaluates to the corresponding system time value.                                                                                                                                            |
| <b>See Also</b>    | <a href="#">SysTimeInCentiSecs()</a> , <a href="#">SysTimeInMilliSecs()</a> ,<br><a href="#">SysTimeInMins()</a> , <a href="#">SysTimeInSecs()</a> ,<br><a href="#">SysTimeToMicroSecs()</a> |

### **SysTimeInMilliSecs Macro**

|                    |                                                                                                                                                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Create a system time value from a value in milliseconds.                                                                                                                                     |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                                                                                                     |
| <b>Prototype</b>   | <code>#define SysTimeInMilliSecs (<i>milliSecs</i>)</code>                                                                                                                                   |
| <b>Parameters</b>  | <code>→ <i>milliSecs</i></code><br>The time, in milliseconds.                                                                                                                                |
| <b>Returns</b>     | Evaluates to the corresponding system time value.                                                                                                                                            |
| <b>See Also</b>    | <a href="#">SysTimeInCentiSecs()</a> , <a href="#">SysTimeInMicroSecs()</a> ,<br><a href="#">SysTimeInMins()</a> , <a href="#">SysTimeInSecs()</a> ,<br><a href="#">SysTimeToMilliSecs()</a> |

### **SysTimeInMins Macro**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <b>Purpose</b>     | Create a system time value from a value in minutes. |
| <b>Declared In</b> | <code>SystemMgr.h</code>                            |
| <b>Prototype</b>   | <code>#define SysTimeInMins (<i>mins</i>)</code>    |
| <b>Parameters</b>  | <code>→ <i>mins</i></code><br>The time, in minutes. |

**Returns** Evaluates to the corresponding system time value.

**See Also** [SysTimeInCentiSecs\(\)](#), [SysTimeInMicroSecs\(\)](#),  
[SysTimeInMilliSecs\(\)](#), [SysTimeInSecs\(\)](#)

## **SysTimeInSecs Macro**

**Purpose** Create a system time value from a value in seconds.

**Declared In** `SystemMgr.h`

**Prototype** `#define SysTimeInSecs (secs)`

**Parameters** `→ secs`  
The time, in seconds.

**Returns** Evaluates to the corresponding system time value.

**See Also** [SysTimeInCentiSecs\(\)](#), [SysTimeInMicroSecs\(\)](#),  
[SysTimeInMilliSecs\(\)](#), [SysTimeInMins\(\)](#),  
[SysTimeToSecs\(\)](#)

## **SysTimeToMicroSecs Macro**

**Purpose** Convert a system time value to microseconds.

**Declared In** `SystemMgr.h`

**Prototype** `#define SysTimeToMicroSecs (sysTime)`

**Parameters** `→ sysTime`  
The system time value to be converted.

**Returns** Evaluates to the corresponding time value in microseconds.

**See Also** [SysTimeInMicroSecs\(\)](#), [SysTimeToMilliSecs\(\)](#),  
[SysTimeToSecs\(\)](#)

### **SysTimeToMilliSecs Macro**

|                    |                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Convert a system time value to milliseconds.                                                                     |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                         |
| <b>Prototype</b>   | <code>#define SysTimeToMilliSecs (sysTime)</code>                                                                |
| <b>Parameters</b>  | <code>→ sysTime</code><br>The system time value to be converted.                                                 |
| <b>Returns</b>     | Evaluates to the corresponding time value in milliseconds.                                                       |
| <b>See Also</b>    | <a href="#">SysTimeInMilliSecs()</a> , <a href="#">SysTimeToMicroSecs()</a> ,<br><a href="#">SysTimeToSecs()</a> |

### **SysTimeToSecs Macro**

|                    |                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Convert a system time value to seconds.                                                                          |
| <b>Declared In</b> | <code>SystemMgr.h</code>                                                                                         |
| <b>Prototype</b>   | <code>#define SysTimeToSecs (sysTime)</code>                                                                     |
| <b>Parameters</b>  | <code>→ sysTime</code><br>The system time value to be converted.                                                 |
| <b>Returns</b>     | Evaluates to the corresponding time value in seconds.                                                            |
| <b>See Also</b>    | <a href="#">SysTimeInSecs()</a> , <a href="#">SysTimeToMicroSecs()</a> ,<br><a href="#">SysTimeToMilliSecs()</a> |

### **SysTurnDeviceOn Function**

|                    |                                          |
|--------------------|------------------------------------------|
| <b>Purpose</b>     | Does nothing.                            |
| <b>Declared In</b> | <code>SystemMgr.h</code>                 |
| <b>Prototype</b>   | <code>void SysTurnDeviceOn (void)</code> |
| <b>Parameters</b>  | None.                                    |
| <b>Returns</b>     | Nothing.                                 |

## SysUIBusy Function

- Purpose** This function originally let you get or set the system UI busy count. In Palm OS Cobalt, however, this function is provided only for compatibility: it effectively does nothing.
- Declared In** `SystemMgr.h`
- Prototype** `uint16_t SysUIBusy (Boolean set, Boolean value)`
- Parameters**
- *set*  
true to alter the UI busy count, false to retrieve the system UI busy count. In Palm OS Cobalt this parameter is ignored.
  - *value*  
true to increment the UI busy count, false to decrement it. This parameter is ignored if *set* is false. In Palm OS Cobalt this parameter is ignored.
- Returns** Returns the current UI busy count. The user interface is not busy if the value returned here is 0. In Palm OS Cobalt this function always returns zero.



# SysThread

---

This chapter provides reference documentation for the threading APIs provided by Palm OS. These APIs allow you to work with threads and thread groups, semaphores, condition variables, and critical sections. As well, the threading APIs include a set of atomic operations that some applications may find useful.

The contents of this chapter is divided into the following sections:

|                                                |     |
|------------------------------------------------|-----|
| <a href="#">SysThread Structures and Types</a> | 451 |
| <a href="#">SysThread Constants</a>            | 454 |
| <a href="#">SysThread Functions and Macros</a> | 458 |
| <a href="#">Application-Defined Functions</a>  | 483 |

The header file `SysThread.h` declares the API that this chapter describes.

This chapter provides reference information on the individual types, constants, and functions that you use when writing multi-threaded applications. For background information as well as tips on using these APIs, see [Chapter 8, “Threading,”](#) on page 85.

## SysThread Structures and Types

### SysConditionVariableType Typedef

|                    |                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Defines a condition variable upon which threads will block until the variable becomes open.                                                                  |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                     |
| <b>Prototype</b>   | <code>typedef void *SysConditionVariableType</code>                                                                                                          |
| <b>Comments</b>    | Variables of this type should be initialized to <code>sysConditionVariableInitializer</code> . This marks the condition variable as closed: to open it, call |

## **SysThread**

*SysCriticalSectionType*

---

[SysConditionVariableOpen\(\)](#). To block on the condition variable, use [SysConditionVariableWait\(\)](#).

### **SysCriticalSectionType Typedef**

|                    |                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Defines a mutex that is used to control access to a critical section.                                                                                                                                                                                                                                                                                   |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                                                                                                |
| <b>Prototype</b>   | <code>typedef void *SysCriticalSectionType</code>                                                                                                                                                                                                                                                                                                       |
| <b>Comments</b>    | Use <code>SysCriticalSectionType</code> variables in conjunction with <a href="#">SysCriticalSectionEnter()</a> and <a href="#">SysCriticalSectionExit()</a> to prevent more than one thread from executing the code in the critical section at the same time.<br><br>Initialize variables of this type to <code>sysCriticalSectionInitializer</code> . |

### **SysThreadExitCallbackID Typedef**

|                    |                                                                                                                                                                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | ID of a thread exit callback function.                                                                                                                                                                                                                           |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                         |
| <b>Prototype</b>   | <code>typedef uint32_t SysThreadExitCallbackID</code>                                                                                                                                                                                                            |
| <b>Comments</b>    | This identifier can be returned to you when you install the exit callback function with <a href="#">SysThreadInstallExitCallback()</a> . You use it to identify the callback function to be removed when calling <a href="#">SysThreadRemoveExitCallback()</a> . |

### **SysThreadGroupHandle Typedef**

|                    |                                                                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Handle to a thread group.                                                                                                                                                                                                                                           |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                            |
| <b>Prototype</b>   | <code>typedef SysThreadGroupType *SysThreadGroupHandle</code>                                                                                                                                                                                                       |
| <b>Comments</b>    | Thread groups are identified by a <code>SysThreadGroupHandle</code> . You obtain a thread group handle when you create the group with <a href="#">SysThreadGroupCreate()</a> . You then supply this handle when adding threads to the group (at creation time, with |

[SysThreadCreate\(\)](#)), or when waiting on the group's threads or destroying the group ([SysThreadGroupWait\(\)](#) and [SysThreadGroupDestroy\(\)](#), respectively).

## **SysThreadGroupTag Struct**

**Purpose** Identifier for an internal structure that represents a thread group.

**Declared In** `SysThread.h`

**Prototype** `struct SysThreadGroupTag`

## **SysThreadGroupType Struct**

**Purpose** Type definition for an internal structure that represents a thread group.

**Declared In** `SysThread.h`

**Prototype** `typedef struct SysThreadGroupTag  
SysThreadGroupType`

## **SysTSDSlotID Typedef**

**Purpose** Thread-specific data (TSD) slot identifier.

**Declared In** `SysThread.h`

**Prototype** `typedef uint32_t SysTSDSlotID`

**Comments** You receive a `SysTSDSlotID` when you allocate a new thread-specific data slot with [SysTSDAllocate\(\)](#).

## SysThread Constants

### Thread Priorities

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Common thread priority values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Constants</b>   | <pre>#define sysThreadPriorityBestApp 80     The highest priority for the application process and any     thread created by it.  #define sysThreadPriorityBestSystem 5     The highest priority reserved for system programs.  #define sysThreadPriorityBestUser 30     The highest priority that should be used by user programs.  #define sysThreadPriorityDisplay 60     Priority that can be used when drawing to the screen.  #define sysThreadPriorityHigh     sysThreadPriorityRealTime     An alias for sysThreadPriorityRealTime.  #define sysThreadPriorityLow     sysThreadPriorityLowered     An alias for sysThreadPriorityLowered.  #define sysThreadPriorityLowered 100     Priority to be used by background threads or threads that are     "CPU hogs."  #define sysThreadPriorityNormal 80     Default priority for event handling.  #define sysThreadPriorityRaised 70     Increased priority (over sysThreadPriorityNormal) for     user operations.  #define sysThreadPriorityRealTime 40     Priority to be used by time-critical code, such as that doing     audio recording and playback.  #define sysThreadPriorityTransaction 65     Priority to be used for UI transactions.</pre> |

```
#define sysThreadPriorityUrgentDisplay 50
    Priority that can be used during event collection and
    dispatching.
```

**Comments** [SysThreadCreate\(\)](#) and [EvtCreateBackgroundThread\(\)](#) do not guarantee that the requested priority will be satisfied. A return value of `errNone` does not guarantee that the thread has been created at requested priority. Depending upon the context in which the function was called, the actual thread priority may be lower than what was requested. In operating system processes, the highest priority is `sysThreadPriorityBestSystem`. For an application, the highest is `sysThreadPriorityBestApp`. Accordingly, if an application calls `SysThreadCreate()` and specifies a priority of 79, the thread will be created successfully but the actual priority of the new thread will be 80.

## Miscellaneous System Thread Constants

**Purpose** The header file `SysThread.h` also declares these constants.

**Declared In** `SysThread.h`

**Constants**

```
#define sysConditionVariableInitializer NULL
    Initial value for a condition variable of type
    SysConditionVariableType.
```

```
#define sysCriticalSectionInitializer NULL
    Initial value for a critical section mutex of type
    SysCriticalSectionType.
```

```
#define sysThreadNoGroup NULL
    A thread group handle value that represents “no thread
    group.” This value is returned from
    SysThreadGroupCreate\(\) if the thread group couldn’t be
    created. Pass this value to SysThreadCreate\(\) if you
    don’t want the thread you are creating to be part of a thread
    group. Note that SysThreadCreateEZ\(\) only creates
    threads that are not part of a thread group: it calls
    SysThreadCreate() and passes a value of
    sysThreadNoGroup for the group parameter.
```

```
#define sysThreadStackBasic 4*1024
    The size of a stack, in bytes, required by a typical non-UI
    thread.
```

## **SysThread**

### *Predefined TSD Slot Names*

---

```
#define sysThreadStackUI 8*1024
```

The size of a stack, in bytes, required by a typical UI thread. Threads created with [SysThreadCreateEZ\(\)](#) have a stack of this size.

## **Predefined TSD Slot Names Enum**

- Purpose** Defines a set of TSD slot names that have special meaning to the operating system.
- Declared In** `SysThread.h`
- Constants** `sysTSDAnonymous = 0`  
Anonymous slot. Supply this value for the slot name when allocating a new TSD slot (using [SysTSDAllocate\(\)](#)) if you don't need to update the slot's destructor function pointer and thus don't need to give the slot a particular name.

## **Predefined Semaphore Counts Enum**

- Purpose** Defines useful semaphore count values.
- Declared In** `SysThread.h`
- Constants** `sysSemaphoreMaxCount = 0xffff`  
The greatest value that can be supplied either for the initial or maximum semaphore counts when creating a semaphore with either [SysSemaphoreCreate\(\)](#) or [SysSemaphoreCreateEZ\(\)](#).

## **timeoutFlags\_t Typedef**

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Variables of this type contain flags that indicate how a given timeout value should be interpreted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Prototype</b>   | <code>typedef uint16_t timeoutFlags_t</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Constants</b>   | <p><code>P_WAIT_FOREVER = 0x0000</code><br/>The specified timeout value should be ignored; the thread or semaphore should wait for an indefinite period of time. This flag should never be used with <a href="#">SysThreadDelay()</a>.</p> <p><code>P_POLL = 0x0300</code><br/>The timeout value should be ignored, and the function should return immediately.</p> <p><code>P_RELATIVE_TIMEOUT = 0x0100</code><br/>The timeout value is a number of nanoseconds that should be added to the amount of time the thread or semaphore is already scheduled to wait.</p> <p><code>P_ABSOLUTE_TIMEOUT = 0x0200</code><br/>The timeout value specifies the absolute number of nanoseconds that the thread or semaphore should wait.</p> |
| <b>Comments</b>    | These flags are used in conjunction with the <a href="#">SysSemaphoreWait()</a> , <a href="#">SysSemaphoreWaitCount()</a> , and <a href="#">SysThreadDelay()</a> functions. These values are contained in a variable of type <a href="#">timeoutFlags_t</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## SysThread Functions and Macros

### SysAtomicAdd32 Function

- Purpose** Atomically adds a 32-bit quantity to a location in memory and returns the value of that memory location as it was prior to the addition.
- Declared In** `SysThread.h`
- Prototype**  
`int32_t SysAtomicAdd32  
(int32_t volatile *ioOperandP,  
int32_t iAddend)`
- Parameters**  
 $\leftrightarrow$  `ioOperandP`  
Pointer to a 32-bit quantity to which `iAddend` is to be added.  
 $\rightarrow$  `iAddend`  
Value to be added to `*ioOperandP`.
- Returns** Returns the value in `*ioOperandP` before `iAddend` was added to it.
- See Also** [SysAtomicAnd32\(\)](#), [SysAtomicCompareAndSwap32\(\)](#), [SysAtomicOr32\(\)](#)

### SysAtomicAnd32 Function

- Purpose** Atomically ANDs a 32-bit quantity into a location in memory and returns the value of that memory location as it was prior to the AND operation.
- Declared In** `SysThread.h`
- Prototype**  
`uint32_t SysAtomicAnd32  
(uint32_t volatile *ioOperandP,  
uint32_t iValue)`
- Parameters**  
 $\leftrightarrow$  `ioOperandP`  
Pointer to a 32-bit quantity to which `iValue` is to be ANDed. Upon return the indicated memory location contains the results of the AND operation.  
 $\rightarrow$  `iValue`  
Value to be ANDed with `*ioOperandP`.

**Returns** Returns the value in *\*ioOperandP* before *iValue* was ANDed with it.

**See Also** [SysAtomicAdd32\(\)](#), [SysAtomicCompareAndSwap32\(\)](#), [SysAtomicOr32\(\)](#)

## SysAtomicCompareAndSwap32 Function

**Purpose** In a single atomic operation, compares the contents of a location in memory with a supplied 32-bit value, and, if they are the same, changes the location in memory to a second supplied 32-bit value.

**Declared In** SysThread.h

**Prototype**

```
uint32_t SysAtomicCompareAndSwap32
(uint32_t volatile *ioOperandP,
 uint32_t iOldValue, uint32_t iNewValue)
```

**Parameters**

- ↔ *ioOperandP*  
Pointer to a 32-bit quantity against which *iOldValue* is to be compared. Upon return the indicated memory location is set to *iNewValue* if *\*ioOperandP* proved equal to *iOldValue*.
- *iOldValue*  
Value against which *\*ioOperandP* is to be compared.
- *iNewValue*  
Value to be stored in *\*ioOperandP* if *\*ioOperandP* is equal to *iOldValue*.

**Returns** Returns 0 if the swap was performed, or 1 if the swap was not performed (thus indicating that the location pointed to by *ioOperandP* wasn't equal to *iOldvalue*).

**See Also** [SysAtomicAdd32\(\)](#), [SysAtomicAnd32\(\)](#), [SysAtomicOr32\(\)](#)

## **SysAtomicOr32 Function**

- Purpose** Atomically ORs a 32-bit quantity into a location in memory and returns the value of that memory location as it was prior to the OR operation.
- Declared In** `SysThread.h`
- Prototype**  
`uint32_t SysAtomicOr32  
(uint32_t volatile *ioOperandP,  
uint32_t iValue)`
- Parameters**  
`↔ ioOperandP`  
Pointer to a 32-bit quantity to which *iValue* is to be ORed. Upon return the indicated memory location contains the results of the OR operation.  
`→ iValue`  
Value to be ORed with *\*ioOperandP*.
- Returns** Returns the value in *\*ioOperandP* before *iValue* was ORed with it.
- See Also** [SysAtomicAdd32\(\)](#), [SysAtomicAnd32\(\)](#),  
[SysAtomicCompareAndSwap32\(\)](#)

## **SysConditionVariableBroadcast Function**

- Purpose** Cause all threads waiting on a specified condition variable to continue.
- Declared In** `SysThread.h`
- Prototype**  
`void SysConditionVariableBroadcast  
(SysConditionVariableType *iCV)`
- Parameters**  
`→ iCV`  
Pointer to the condition variable upon which waiting threads are to be released.
- Returns** Nothing.
- Comments** All threads waiting on the specified condition variable are released. Upon completion of this function, the condition variable remains closed.
- See Also** [SysConditionVariableOpen\(\)](#),  
[SysConditionVariableWait\(\)](#)

## SysConditionVariableClose Function

|                    |                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Transition a condition variable from opened to closed.                                     |
| <b>Declared In</b> | SysThread.h                                                                                |
| <b>Prototype</b>   | <pre>void SysConditionVariableClose<br/>    (SysConditionVariableType *iCV)</pre>          |
| <b>Parameters</b>  | → <i>iCV</i><br>Pointer to the condition variable that is to be closed.                    |
| <b>Returns</b>     | Nothing.                                                                                   |
| <b>Comments</b>    | If the condition variable is not open when this function is called, it is left as-is.      |
| <b>See Also</b>    | <a href="#">SysConditionVariableOpen()</a> ,<br><a href="#">SysConditionVariableWait()</a> |

## SysConditionVariableOpen Function

|                    |                                                                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Transition a condition variable from closed to opened.                                                                                                                            |
| <b>Declared In</b> | SysThread.h                                                                                                                                                                       |
| <b>Prototype</b>   | <pre>void SysConditionVariableOpen<br/>    (SysConditionVariableType *iCV)</pre>                                                                                                  |
| <b>Parameters</b>  | → <i>iCV</i><br>Pointer to the condition variable that is to be opened.                                                                                                           |
| <b>Returns</b>     | Nothing.                                                                                                                                                                          |
| <b>Comments</b>    | All threads that are blocked on the specified condition variable are released. If the specified condition variable is already open when this function is called, it does nothing. |
| <b>See Also</b>    | <a href="#">SysConditionVariableBroadcast()</a> ,<br><a href="#">SysConditionVariableClose()</a>                                                                                  |

## **SysConditionVariableWait Function**

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Wait for a specified condition variable to become open.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Prototype</b>   | <pre>void SysConditionVariableWait     (SysConditionVariableType *iCV,      SysCriticalSectionType *iOptionalCS)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Parameters</b>  | <p>→ <i>iCV</i><br/>Pointer to the condition variable upon which to wait.</p> <p>→ <i>iOptionalCS</i><br/>Pointer to the mutex for a critical section if this function is called from within the critical section and the critical section is to be temporarily exited while this thread waits on the condition variable. Pass <code>NULL</code> if the call doesn't come from within a critical section, or if it does but other threads shouldn't be granted access to the critical section while this thread waits.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Comments</b>    | <p>If the critical section is not open when this function is called, the calling thread is put in the <code>WAIT</code> state and placed in a queue associated with the specified condition variable. Once the condition variable is opened, all threads waiting on the variable are released. Note that you can also release all waiting threads by calling <a href="#">SysConditionVariableBroadcast()</a>, which, unlike <a href="#">SysConditionVariableOpen()</a>, leaves the condition variable in a closed state.</p> <p>If the call to this function occurs within a critical section, you may want to supply the critical section mutex as the <i>iOptionalCS</i> parameter. This causes the critical section to be exited in the event that the thread is put into the <code>WAIT</code> state. Once the thread is released the specified critical section is immediately re-entered. This has the effect of allowing other threads to enter the critical section while this thread is waiting on the condition variable. If the call to this function does not occur within a critical section, or if it does but other threads shouldn't be able to enter the critical section while this thread is waiting, pass <code>NULL</code> for the <i>iOptionalCS</i> parameter.</p> |
| <b>See Also</b>    | <a href="#">SysConditionVariableBroadcast()</a> ,<br><a href="#">SysConditionVariableOpen()</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## SysCriticalSectionEnter Function

|                    |                                                                                                                                                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Acquire a critical section—a block of code that can only be executed by one thread at a time.                                                                                                                                                                                       |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                            |
| <b>Prototype</b>   | <code>void SysCriticalSectionEnter<br/>(SysCriticalSectionType *iCS)</code>                                                                                                                                                                                                         |
| <b>Parameters</b>  | → <i>iCS</i><br>Pointer to a <a href="#">SysCriticalSectionType</a> variable that acts as the mutex for the code block. This variable should be either globally or statically defined and should be initialized to <code>sysCriticalSectionInitializer</code> .                     |
| <b>Returns</b>     | Nothing.                                                                                                                                                                                                                                                                            |
| <b>Comments</b>    | Call this function at the beginning of the critical section to ensure that yours is the only process that is executing the code in the critical section—that code that is bounded by the <code>SysCriticalSectionEnter()</code> and <a href="#">SysCriticalSectionExit()</a> calls. |
| <b>See Also</b>    | <a href="#">SysCriticalSectionExit()</a>                                                                                                                                                                                                                                            |

## SysCriticalSectionExit Function

|                    |                                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Release a critical section.                                                                                             |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                |
| <b>Prototype</b>   | <code>void SysCriticalSectionExit<br/>(SysCriticalSectionType *iCS)</code>                                              |
| <b>Parameters</b>  | → <i>iCS</i><br>Pointer to a <a href="#">SysCriticalSectionType</a> variable that acts as the mutex for the code block. |
| <b>Returns</b>     | Nothing.                                                                                                                |
| <b>Comments</b>    | Call this function at the end of the critical section, thereby allowing another process to enter the critical section.  |
| <b>See Also</b>    | <a href="#">SysCriticalSectionEnter()</a>                                                                               |

## SysThread

### *SysCurrentThread*

---

#### **SysCurrentThread Function**

- Purpose** Return the identity of the current thread.
- Declared In** `SysThread.h`
- Prototype** `SysHandle SysCurrentThread (void)`
- Parameters** None.
- Returns** Returns the thread ID of the current thread.

#### **SysGetRunTime Function**

- Purpose** Get the length of time since last reset, in nanoseconds.
- Declared In** `SysThread.h`
- Prototype** `nsecs_t SysGetRunTime (void)`
- Parameters** None.
- Returns** Returns the amount of time, in nanoseconds, since the last reset.

#### **SysSemaphoreCreate Function**

- Purpose** Create a new counting semaphore with the specified initial count and maximum count values.
- Declared In** `SysThread.h`
- Prototype** `status_t SysSemaphoreCreate  
(uint32_t initialCount, uint32_t maxCount,  
uint32_t flags, SysHandle *outSemaphore)`
- Parameters**
- *initialCount*  
The initial count of the newly-created semaphore. This value can range from zero to `sysSemaphoreMaxCount`. Typically, this represents the number of objects available in a resource pool.
  - *maxCount*  
The maximum count of the newly-created semaphore. This value can range from one to `sysSemaphoreMaxCount`. The maximum count typically represents the maximum number of objects that can be in the resource pool.

→ *flags*

Flags that specify how the semaphore is to be created. This parameter is currently ignored; applications should pass zero for *flags*.

← *outSemaphore*

ID of the newly-created semaphore.

**Returns** Returns `errNone` if the semaphore was successfully created, or one of the following otherwise:

`sysErrOutOfRange`

*initialCount* or *maxCount* exceeds `sysSemaphoreMaxCount`.

`sysErrParamErr`

*initialCount* exceeds *maxCount*.

`sysErrNoFreeResource`

All possible semaphores are already in use.

**Comments** Counting semaphores are used to keep track of the availability of a resource within a pool of limited size.

**See Also** [SysSemaphoreCreateEZ\(\)](#), [SysSemaphoreDestroy\(\)](#)

## SysSemaphoreCreateEZ Function

**Purpose** Creates a new counting semaphore with a maximum count value of `sysSemaphoreMaxCount` (65535).

**Declared In** `SysThread.h`

**Prototype** `status_t SysSemaphoreCreateEZ  
(uint32_t initialCount,  
SysHandle *outSemaphore)`

**Parameters** → *initialCount*

The initial count of the newly-created semaphore. This value can range from zero to `sysSemaphoreMaxCount`.

← *outSemaphore*

ID of the newly-created semaphore.

**Returns** Returns `errNone` if the semaphore was successfully created, or one of the following otherwise:

## SysThread

### *SysSemaphoreDestroy*

---

`sysErrOutOfRange`

*initialCount* exceeds `sysSemaphoreMaxCount`.

`sysErrNoFreeResource`

All possible semaphores are already in use.

**Comments** This function simply calls [SysSemaphoreCreate\(\)](#) and passes a value of zero for the *flags* parameter and a value of `sysSemaphoreMaxCount` for the *maxCount* parameter.

**See Also** [SysSemaphoreCreate\(\)](#), [SysSemaphoreDestroy\(\)](#)

## SysSemaphoreDestroy Function

**Purpose** Destroy a counting semaphore.

**Declared In** `SysThread.h`

**Prototype** `status_t SysSemaphoreDestroy  
(SysHandle semaphore)`

**Parameters** → *semaphore*  
ID of the semaphore to be destroyed.

**Returns** Always returns `errNone`.

**Comments** Any threads that are blocked on this semaphore are released and their return values are set to `kalErrObjectDestroyed`.

**See Also** [SysSemaphoreCreate\(\)](#), [SysSemaphoreCreateEZ\(\)](#)

## SysSemaphoreSignal Function

**Purpose** Release a single resource back to the semaphore. One of the threads waiting on the semaphore (assuming that there are any) is released.

**Declared In** `SysThread.h`

**Prototype** `status_t SysSemaphoreSignal (SysHandle semaphore)`

**Parameters** → *semaphore*  
ID of the semaphore to which resources are to be released.

**Returns** Returns `errNone` if the maximum semaphore count wasn't exceeded, or `kalErrLimitReached` if the supplied count would cause the semaphore's count to exceed the maximum value specified when the semaphore was created.

**Comments** This function is equivalent to calling [SysSemaphoreSignalCount\(\)](#) with a count of 1. See the comments under that function for more on how this function operates.

**See Also** [SysSemaphoreSignalCount\(\)](#), [SysSemaphoreWait\(\)](#)

## SysSemaphoreSignalCount Function

**Purpose** Release a specified number of resources back to the semaphore. As a result, as many threads as possible that are waiting on the semaphore are released.

**Declared In** `SysThread.h`

**Prototype** `status_t SysSemaphoreSignalCount  
(SysHandle semaphore, uint32_t count)`

**Parameters**  
→ *semaphore*  
ID of the semaphore to which resources are to be released.  
→ *count*  
The amount to be added to the semaphore's count.

**Returns** Returns `errNone` if the maximum semaphore count wasn't exceeded, or `kalErrLimitReached` if the supplied count would cause the semaphore's count to exceed the maximum value specified when the semaphore was created.

**Comments** Use this function to return one or more resources to the specified semaphore. If there are threads waiting at the semaphore, as many threads as possible (given the new available resource count) are released either to the RUN or READY states. After this the semaphore's new resource count becomes its old count, plus *count*, minus the resources consumed by the newly-released threads.

Note that it is not an error if the semaphore count exceeds the initial semaphore count that was specified when the semaphore was created. If a semaphore is going to be used for exclusive control, create the semaphore with the maximum count equal to the initial semaphore count.

**See Also** [SysSemaphoreSignal\(\)](#), [SysSemaphoreWaitCount\(\)](#)

## **SysSemaphoreWait Function**

- Purpose** Obtains a single resource from a specified semaphore.
- Declared In** `SysThread.h`
- Prototype** `status_t SysSemaphoreWait (SysHandle semaphore, timeoutFlags_t iTimeoutFlags, nsecs_t iTimeout)`
- Parameters**
- *semaphore*  
ID of the semaphore from which resources are to be acquired.
  - *iTimeoutFlags*  
A value specifying how the timeout value should be interpreted. See [timeoutFlags\\_t](#).
  - *iTimeout*  
Timeout value, in nanoseconds.
- Returns** Returns `errNone` if the specified number of resources were successfully obtained, or one of the following otherwise:
- `sysErrTimeout`  
There aren't *count* resources available, and either the timeout expired before enough resources were made available, or the timeout flags were set to `P_POLL`, indicating that this function shouldn't wait for additional resources to become available.
- Comments** This function is equivalent to calling [SysSemaphoreWaitCount\(\)](#) with a count of 1. See the comments under that function for more on how this function operates.
- See Also** [SysSemaphoreSignal\(\)](#), [SysSemaphoreWaitCount\(\)](#)

## SysSemaphoreWaitCount Function

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>     | Obtains one or more resources from a specified semaphore.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Declared In</b> | <code>SysThread.h</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Prototype</b>   | <pre>status_t SysSemaphoreWaitCount         (SysHandle semaphore,          timeoutFlags_t iTimeoutFlags,          nsecs_t iTimeout, uint32_t count)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Parameters</b>  | <ul style="list-style-type: none"><li>→ <i>semaphore</i><br/>ID of the semaphore from which resources are to be acquired.</li><li>→ <i>iTimeoutFlags</i><br/>A value specifying how the timeout value should be interpreted. See <a href="#">timeoutFlags_t</a>.</li><li>→ <i>iTimeout</i><br/>Timeout value, in nanoseconds.</li><li>→ <i>count</i><br/>The number of resources to be acquired.</li></ul>                                                                                                                                                                                        |
| <b>Returns</b>     | Returns <code>errNone</code> if the specified number of resources were successfully obtained, or one of the following otherwise:<br><br><code>sysErrParamErr</code><br><i>count</i> is zero, or <i>count</i> exceeds the maximum count specified when the resource was created.<br><br><code>sysErrTimeout</code><br>There aren't <i>count</i> resources available, and either the timeout expired before enough resources were made available, or the timeout flags were set to <code>P_POLL</code> , indicating that this function shouldn't wait for additional resources to become available. |
| <b>Comments</b>    | If the available count for the specified semaphore is greater than or equal to the number of requested resources, the available count is decremented by the number of requested resources and the function returns with <code>errNone</code> . If the number of available resources is less than the number of resources being requested, the calling thread is put in the <code>WAIT</code> state and placed in a queue associated with the semaphore. Until the request is fulfilled, the semaphore's count is not changed.                                                                     |
| <b>See Also</b>    | <a href="#">SysSemaphoreSignalCount()</a> , <a href="#">SysSemaphoreWait()</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## SysThread

### *SysThreadChangePriority*

---

## SysThreadChangePriority Function

- Purpose** Change the current priority of a thread.
- Declared In** `SysThread.h`
- Prototype** `status_t SysThreadChangePriority  
(SysHandle thread, uint8_t priority)`
- Parameters**
- *thread*  
The thread's identifier.
  - *priority*  
The thread's new priority. Useful thread priority values are defined under "[Thread Priorities](#)" on page 454.
- Returns** Returns `errNone` if the thread's priority was changed, or `sysErrParamErr` if either the specified priority is out of the range of allowable thread priorities, or *thread* doesn't identify a known thread.
- Comments** If this function is successful, the thread's new priority remains in effect until it is changed again.
- This function may result in the re-ordering of the thread queues.
- If `SysThreadChangePriority()` is called for a thread waiting on the ready queue (including running threads) or another priority-based queue, the thread will be moved to the end of that part of the queue designated for the new priority. If the priority specified is the same as the current priority, the thread is still moved behind other threads of the same priority. Thus, a thread can relinquish its execution privileges by calling `SysThreadChangePriority()` on itself and specifying its current priority.
- See Also** [SysThreadCreate\(\)](#)

## SysThreadCreate Function

- Purpose** Create a new thread.
- Declared In** `SysThread.h`
- Prototype**
- ```
status_t SysThreadCreate
    (SysThreadGroupHandle group, const char *name,
     uint8_t priority, uint32_t stackSize,
     SysThreadEnterFunc *func, void *argument,
     SysHandle *outThread)
```
- Parameters**
- *group*  
The thread group of which this thread is to be a part (obtained from [SysThreadGroupCreate\(\)](#)), or `sysThreadNoGroup` if the thread isn't to be part of a thread group.
  - *name*  
The name of the thread. The first four characters are used as the thread's ID. If a thread name isn't provided, the thread's ID is '----'.
  - *priority*  
The requested thread priority. Valid thread priorities range from 1 to 255, with lower values having higher priority. Priority level 0 is reserved and cannot be used—applications typically have a thread priority no higher than 30. Common thread priority values are defined under “[Thread Priorities](#)” on page 454.
  - *stackSize*  
The size of the thread's stack, in bytes. For a UI thread this is typically 8 KB.
  - *func*  
Pointer to the function that should initially be executed when the thread is started. See [SysThreadEnterFunc\(\)](#) for the prototype of this function.
  - *argument*  
Pointer to the argument block to be passed to the thread entry function identified by *func*, or NULL if the function takes no arguments.
  - ← *outThread*  
The thread identifier. This value is owned by the newly-created thread and will be freed when the thread exits.

## SysThread

### *SysThreadCreateEZ*

---

**Returns** Returns `errNone` if the thread was successfully created, or one of the following otherwise:

`sysErrParamErr`

The *func* parameter is `NULL`, or the *stackSize* parameter is zero.

`sysErrNoFreeResource`

The system has no free threads.

**Comments** This function creates the specified thread. The new thread is put in the DORMANT state.

---

**NOTE:** `SysThreadCreate()` does not guarantee that the requested priority will be satisfied. A return value of `errNone` does not guarantee that the thread has been created at requested priority. Depending upon the context in which the function was called, the actual thread priority may be lower than what was requested.

---

**See Also** [EvtCreateBackgroundThread\(\)](#), [SysThreadCreateEZ\(\)](#), [SysThreadGroupCreate\(\)](#), [SysThreadInstallExitCallback\(\)](#), [SysThreadStart\(\)](#)

## SysThreadCreateEZ Function

**Purpose** Create a new thread, using default values for the thread group, priority, and stack size.

**Declared In** `SysThread.h`

**Prototype** `status_t SysThreadCreateEZ (const char *name, SysThreadEnterFunc *func, void *argument, SysHandle *outThread)`

**Parameters** → *name*

The name of the thread. The first four characters are used as the thread's ID. If a thread name isn't provided, the thread's ID is '----'.

→ *func*

Pointer to the function that should initially be executed when the thread is started. See [SysThreadEnterFunc\(\)](#) for the prototype of this function.

→ *argument*

Pointer to the argument block to be passed to the thread entry function identified by *func*, or NULL if the function takes no arguments.

← *outThread*

The thread identifier. This value is owned by the newly-created thread and will be freed when the thread exits.

**Returns** Returns `errNone` if the thread was successfully created, or one of the following otherwise:

`sysErrParamErr`

The *func* parameter is NULL, or the *stackSize* parameter is zero.

`sysErrNoFreeResource`

The system has no free threads.

**Comments** This function simply calls [SysThreadCreate\(\)](#) with the *group* parameter set to `sysThreadNoGroup`, the *priority* parameter set to `sysThreadPriorityNormal`, and the *stackSize* parameter set to `sysThreadStackUI`.

**See Also** [SysThreadInstallExitCallback\(\)](#), [SysThreadStart\(\)](#)

## SysThreadDelay Function

**Purpose** Halts execution of the thread calling this function for a specified period of time.

**Declared In** `SysThread.h`

**Prototype** `status_t SysThreadDelay (nsecs_t timeout,  
timeoutFlags_t flags)`

**Parameters** → *timeout*

Timeout value, in nanoseconds.

→ *flags*

A value specifying how the timeout value should be interpreted. See [timeoutFlags\\_t](#).

**Returns** Returns `errNone` if the function executed successfully, or `sysErrParamErr` if *flags* was set to `P_WAIT_FOREVER`.

**Comments** This system call temporarily halts the execution of the calling thread and makes it enter the time elapse wait state (this is one type of WAIT state). The thread halts execution for the amount of time defined by *timeout*. If *timeout* is 0 or if *flags* is P\_POLL, the function does a yield, moving the invoking thread to the end of the priority level in the ready queue.

The time count continues even if the calling thread later enters the SUSPEND state (placing it in the WAIT-SUSPEND state).

This function returns after the calling thread has been delayed by the specified amount of time.

**See Also** [SysThreadChangePriority\(\)](#), [SysThreadGroupWait\(\)](#), [SysThreadSuspend\(\)](#)

## **SysThreadExit Function**

**Purpose** Causes the issuing thread to exit.

**Declared In** `SysThread.h`

**Prototype** `void SysThreadExit (void)`

**Parameters** None.

**Returns** Nothing.

**Comments** A thread can exit either by calling this function or by returning from its initial entry function (specified by the *func* parameter when creating the thread with [SysThreadCreate\(\)](#)).

This function changes the state of the specified thread to the DORMANT state. Because the thread still exists and is in the DORMANT state, it can again be started (at which point the initial entry function is executed, just as it was when the thread was originally started) by calling [SysThreadStart\(\)](#).

When a thread calls this function to exit, that thread does not automatically release all of the resources (stack, memory blocks, semaphores, thread-specific data slots, and so on) which it had obtained prior to the function call. Your code is responsible for releasing all resources beforehand or, as in the case of the stack, after this function returns.

Note that if an error is detected during the execution of this function, the error is not returned to the thread which called this function.

**See Also** [SysThreadSuspend\(\)](#)

## SysThreadGroupCreate Function

- Purpose** Create a new thread group.
- Declared In** `SysThread.h`
- Prototype** `SysThreadGroupHandle SysThreadGroupCreate (void)`
- Parameters** None.
- Returns** Returns a handle to the new thread group, or `sysThreadNoGroup` if the a new thread group couldn't be created.
- Comments** Thread groups are a convenience provided by the operating system that allows you to wait for one or more threads to exit. Thread groups are useful for unloading libraries that have spawned their own threads. Note that destroying a thread group implicitly waits for all threads in that group to exit.
- Threads must be added to a thread group at the time that they are created. Specify the thread group handle returned from `SysThreadGroupCreate()` to [SysThreadCreate\(\)](#) in order to have the newly-created thread added to the thread group.
- See Also** [SysThreadCreate\(\)](#), [SysThreadGroupDestroy\(\)](#), [SysThreadGroupWait\(\)](#)

## SysThreadGroupDestroy Function

- Purpose** Destroy a thread group.
- Declared In** `SysThread.h`
- Prototype** `status_t SysThreadGroupDestroy  
(SysThreadGroupHandle group)`
- Parameters**  $\rightarrow$  *group*  
Handle to the thread group to be destroyed.
- Returns** Always returns `errNone`.

## SysThread

### *SysThreadGroupWait*

---

**Comments** This function waits until all of the group's threads have exited before destroying the thread group.

**See Also** [SysThreadCreate\(\)](#), [SysThreadGroupCreate\(\)](#), [SysThreadGroupWait\(\)](#)

## SysThreadGroupWait Function

**Purpose** Wait until all of the threads in the specified thread group have exited.

**Declared In** `SysThread.h`

**Prototype** `status_t SysThreadGroupWait  
(SysThreadGroupHandle group)`

**Parameters** → *group*  
Handle to the thread group upon which to wait.

**Returns** Always returns `errNone`.

**See Also** [SysThreadCreate\(\)](#), [SysThreadGroupCreate\(\)](#), [SysThreadGroupDestroy\(\)](#)

## SysThreadInstallExitCallback Function

**Purpose** Installs a function that will be executed when the current thread exits.

**Declared In** `SysThread.h`

**Prototype** `status_t SysThreadInstallExitCallback  
(SysThreadExitCallbackFunc *iExitCallbackP,  
void *iCallbackArg,  
SysThreadExitCallbackID *oThreadExitCallbackId  
)`

**Parameters** → *iExitCallbackP*  
Pointer to the function to be executed when the thread exits. Your callback function should have the prototype defined by [SysThreadExitCallbackFunc\(\)](#).

→ *iCallbackArg*  
Pointer to the argument block needed by the exit callback function, or NULL if the callback requires no arguments.

← *oThreadExitCallbackId*

Pointer to a location where the ID of the exit callback function is stored, or NULL if you don't need this ID. You use this value in the event that you need to remove the exit callback function with [SysThreadRemoveExitCallback\(\)](#).

**Returns** Returns `errNone` if the exit callback function was installed successfully, or `sysErrNoFreeRAM` if there wasn't enough memory to allocate the thread callback structure.

**Comments** A thread's exit callback functions are executed when the thread exits. You can install multiple exit callback functions for a given thread: when the thread exits, they are executed in the reverse order in which they were installed. That is, the last exit callback function installed will be the first executed.

The exit callback functions are stored in a thread-specific data (TSD) slot for the current thread.

**See Also** [SysThreadExit\(\)](#), [SysThreadRemoveExitCallback\(\)](#)

## **SysThreadRemoveExitCallback Function**

**Purpose** Removes a thread exit callback function originally installed with [SysThreadInstallExitCallback\(\)](#).

**Declared In** `SysThread.h`

**Prototype** `status_t SysThreadRemoveExitCallback  
(SysThreadExitCallbackID iThreadCallbackId)`

**Parameters** → *iThreadCallbackId*  
ID of the exit callback function returned from [SysThreadInstallExitCallback\(\)](#).

**Returns** Returns `errNone` if the specified exit callback function was successfully removed, or `sysErrParamErr` if the supplied callback ID doesn't reference an exit callback function in the current thread's TSD (thread-specific data).

**See Also** [SysThreadExit\(\)](#), [SysThreadInstallExitCallback\(\)](#)

## **SysThreadResume Function**

- Purpose** Resumes execution of a suspended thread.
- Declared In** `SysThread.h`
- Prototype** `status_t SysThreadResume (SysHandle thread)`
- Parameters** `→ thread`  
The ID of the thread that is to resume execution.
- Returns** Returns `errNone` if the operation completed successfully (but note that because [SysThreadSuspend\(\)](#) can be called multiple times on a given thread, a return value of `errNone` from `SysThreadResume()` doesn't necessarily mean that the thread is now executing). If `thread` doesn't reference a suspended thread, this function returns `kalErrObjectInvalid`.
- Comments** This function releases the SUSPEND state of the specified thread. Specifically, it causes the SUSPEND state to be released and the execution of the specific prior call to [SysThreadSuspend\(\)](#).
- If the specified thread is in WAIT-SUSPEND state, a call to `SysThreadResume()` only releases the SUSPEND state; the thread will then be in the WAIT state.
- A thread cannot specify itself to this function. If a thread attempts to do so, `kalErrObjectInvalid` is returned.
- `SysThreadResume()` only counters a single suspend request. Accordingly, if `SysThreadSuspend()` has been called more than once for the thread, that thread will remain suspended even after `SysThreadResume()` returns.
- After resuming, the thread in the ready queue remains in the same position it was prior to suspension.
- See Also** [SysThreadStart\(\)](#), [SysThreadSuspend\(\)](#)

## SysThreadStart Function

<b>Purpose</b>	Start a thread created with <a href="#">SysThreadCreate()</a> .
<b>Declared In</b>	<code>SysThread.h</code>
<b>Prototype</b>	<code>status_t SysThreadStart (SysHandle thread)</code>
<b>Parameters</b>	<code>→ thread</code> Handle to the thread to be started. This is the value returned through the <code>outThread</code> parameter of <a href="#">SysThreadCreate()</a> .
<b>Returns</b>	Returns <code>errNone</code> if the thread was started, or <code>kalErrObjectInvalid</code> if the specified thread is not in the DORMANT state.
<b>Comments</b>	<p>This function changes the state of the specified thread from DORMANT to RUN/READY.</p> <p>The thread priority on starting the thread is that which was specified when the thread was created. That priority may have been changed by any calls to <a href="#">SysThreadChangePriority()</a> prior to this function call.</p> <p>If this function is called for a thread that is not in the DORMANT state, this function does nothing and returns <code>kalErrObjectInvalid</code>. If the thread is suspended, call <a href="#">SysThreadResume()</a> instead.</p>
<b>See Also</b>	<a href="#">SysThreadCreate()</a> , <a href="#">SysThreadResume()</a>

## SysThreadSuspend Function

<b>Purpose</b>	Suspend execution of a thread.
<b>Declared In</b>	<code>SysThread.h</code>
<b>Prototype</b>	<code>status_t SysThreadSuspend (SysHandle thread)</code>
<b>Parameters</b>	<code>→ thread</code> The thread ID of the thread to suspend.
<b>Returns</b>	Returns <code>errNone</code> if the thread was suspended, or one of the following otherwise:

## SysThread

### *SysThreadSuspend*

---

`kalErrObjectInvalid`

The specified thread is dormant, or the specified thread is the current thread.

`kalErrLimitReached`

This function has been called more than 255 times for the given thread.

#### **Comments**

A thread cannot suspend itself by calling this function.

The specified thread is placed into the SUSPEND state. To cause this thread to resume, call [SysThreadResume\(\)](#).

If the thread specified is already in the WAIT state, it is put in the combined WAIT-SUSPEND state. If wait conditions for the thread are later fulfilled, it will enter the SUSPEND state. If you supply a thread that is in the combined WAIT-SUSPEND state to [SysThreadResume\(\)](#), the thread returns to the WAIT state. The upshot of all of this is that the WAIT and SUSPEND states are independent and either one prevents execution.

If [SysThreadSuspend\(\)](#) is called more than once for a given thread, that thread is put in multiple SUSPEND states. This is called suspend request nesting. In this case, you must call [SysThreadResume\(\)](#) the same number of times that [SysThreadSuspend\(\)](#) was called in order to return the thread to its original state before the suspension. Note that the maximum number of times suspend requests may be nested is 255.

A thread which is suspended in addition to waiting for resources (such as waiting for a semaphore) can be allocated resources (such as semaphore counts) based on the same conditions as threads which are not suspended. Even when suspended, the allocation of resources is not delayed in any way. Conditions concerning resource allocation and release of the wait state remain unchanged. In other words, the SUSPEND state is completely independent of other processing and thread states.

If you need to delay the allocation of resources to a thread which is suspended, the use [SysThreadChangePriority\(\)](#) in conjunction with [SysThreadSuspend\(\)](#) and [SysThreadResume\(\)](#).

#### **See Also**

[SysThreadDelay\(\)](#), [SysThreadExit\(\)](#), [SysThreadResume\(\)](#)

## SysTSDAllocate Function

- Purpose** Allocates a new thread-specific data (TSD) slot.
- Declared In** `SysThread.h`
- Prototype**  
`status_t SysTSDAllocate (SysTSDSlotID *oTSDSlot,  
SysTSDDestructorFunc *iDestructor,  
uint32_t iName)`
- Parameters**
- ← *oTSDSlot*  
TSD slot identifier, set upon successful allocation of the slot.
  - *iDestructor*  
Pointer to a destructor function that is called to clean up any data associated with the slot when the thread exits. This function is optional; pass NULL if you don't want to use a destructor. The destructor function should have the prototype defined by [SysTSDDestructorFunc\(\)](#).
  - *iName*  
TSD slot name (note that this is a 32-bit value). Pass `sysTSDAnonymous` if you don't care about the slot name (see the Comments section, below, for more on why you might care about the TSD slot name).
- Returns** Returns `errNone` if the operation completed successfully, or `kaErrLimitReached` if the thread already has the maximum number of TSD slots allocated for it.
- Comments** Calling `SysTSDAllocate()` more than once with the same slot name simply updates the destructor function pointer; the same TSD slot identifier is returned each time. If you don't need to update the TSD slot's destructor function pointer, you can pass `sysTSDAnonymous` for the TSD slot name. This simply allocates the next available slot, sets its destructor pointer to the supplied value, and returns the slot's identifier.
- See Also** [SysTSDFree\(\)](#), [SysTSDGet\(\)](#), [SysTSDSet\(\)](#)

## SysTSDFree Function

- Purpose** Deallocates a previously created thread-specific data (TSD) slot.
- Declared In** `SysThread.h`
- Prototype** `status_t SysTSDFree (SysTSDSlotID tsdslot)`
- Parameters** `→ tsdslot`  
TSD slot identifier obtained from [SysTSDAllocate\(\)](#).
- Returns** Returns `errNone` if the TSD slot was successfully freed, or `sysErrParamErr` if *tsdslot* doesn't reference a valid slot.
- Comments** Applications should not normally make use of this function. A thread's TSD slots are deallocated automatically when the thread exits, so applications generally don't need to free the slots explicitly.
- See Also** [SysThreadExit\(\)](#), [SysTSDAllocate\(\)](#), [SysTSDDestructorFunc\(\)](#)

## SysTSDGet Function

- Purpose** Get the contents of a specified thread-specific data (TSD) slot.
- Declared In** `SysThread.h`
- Prototype** `void *SysTSDGet (SysTSDSlotID tsdslot)`
- Parameters** `→ tsdslot`  
TSD slot identifier obtained from [SysTSDAllocate\(\)](#).
- Returns** Returns the slot contents, or 0 if *tsdslot* doesn't reference a valid TSD slot.
- See Also** [SysTSDAllocate\(\)](#), [SysTSDSet\(\)](#)

## SysTSDSet Function

- Purpose** Set the contents of a specified thread-specific data (TSD) slot.
- Declared In** `SysThread.h`
- Prototype** `void SysTSDSet (SysTSDSlotID tsdslot,  
void *iValue)`
- Parameters** `→ tsdslot`  
TSD slot identifier obtained from [SysTSDAllocate\(\)](#).

→ *iValue*

A 32-bit value to be placed into the specified TSD slot.

**Returns** Nothing.

**Comments** This function does nothing if *tsdslot* doesn't reference a valid TSD slot.

**See Also** [SysTSDAllocate\(\)](#), [SysTSDGet\(\)](#)

## Application-Defined Functions

### **SysThreadEnterFunc Function**

**Purpose** Prototype of a thread entry function.

**Declared In** `SysThread.h`

**Prototype** `void (SysThreadEnterFunc) (void *param)`

**Parameters** → *param*

The parameter pointer originally supplied to the [SysThreadCreate\(\)](#) or [SysThreadCreateEZ\(\)](#) function

**Returns** Nothing.

**See Also** [SysThreadExitCallbackFunc\(\)](#)

### **SysThreadExitCallbackFunc Function**

**Purpose** Prototype of a thread exit callback function.

**Declared In** `SysThread.h`

**Prototype** `void (SysThreadExitCallbackFunc) (void *param)`

**Parameters** → *param*

The parameter pointer originally supplied to the [SysThreadInstallExitCallback\(\)](#) function

**Returns** Nothing.

**See Also** [SysThreadEnterFunc\(\)](#)

## SysThread

*SysTSDDestructorFunc*

---

### SysTSDDestructorFunc Function

- Purpose** Prototype of a thread-specific data (TSD) slot destructor function.
- Declared In** `SysThread.h`
- Prototype** `void (SysTSDDestructorFunc) (void *param)`
- Parameters**  $\rightarrow$  *param*  
The contents of the TSD slot. This is the value supplied in the most recent call to [SysTSDSet\(\)](#) for the slot.
- Returns** Nothing.
- Comments** The slot's destructor function—specified when the TSD slot was allocated with [SysTSDAllocate\(\)](#)—is called to clean up any data associated with the slot when the thread exits. This function is only called for a slot if the slot contents are non-NULL. Slots are initialized to NULL when they are allocated, so if you never call [SysTSDSet\(\)](#) for a slot, when the thread exits the slot destructor isn't called for that slot.
- See Also** [SysTSDAllocate\(\)](#)

# System Utilities

---

The utility APIs declared here are of general use by applications. These functions are wide-ranging, and include functions that return the handheld's battery type, to functions that sort arrays using either quicksort or insertion sort algorithms, to functions that retrieve specific resource types.

This chapter is organized as follows:

<a href="#">System Utilities Structures and Types</a>	486
<a href="#">System Utilities Constants</a>	487
<a href="#">System Utilities Functions and Macros</a>	488

The header file `SysUtils.h` declares the API that this chapter describes.

## System Utilities Structures and Types

### **CmpFuncPtr Typedef**

- Purpose** Pointer to a comparison function that is used when sorting arrays with either [SysInsertionSort\(\)](#) or [SysQSort\(\)](#).
- Declared In** `SysUtils.h`
- Prototype** `typedef _comparF *CmpFuncPtr`

### **SearchFuncPtr Typedef**

- Purpose** Pointer to a search function, used with [SysBinarySearch\(\)](#).
- Declared In** `SysUtils.h`
- Prototype** `typedef _searchF *SearchFuncPtr`

### **SysBatteryKind Typedef**

- Purpose** Contains a value indicating the type of battery used by the handheld.
- Declared In** `SysUtils.h`
- Prototype** `typedef Enum8 SysBatteryKind`
- Comments** See [SysBatteryKindTag](#) (in [Chapter 17, "Common Battery Types,"](#) on page 165) for the set of values that this type can assume.

### **SysDBListItemType Struct**

- Purpose** Describes a single database or panel. The [SysCreateDataBaseList\(\)](#) and [SysCreatePanelList\(\)](#) functions each create and return an array of `SysDBListItemType` structures.

<b>Declared In</b>	SysUtils.h														
<b>Prototype</b>	<pre>typedef struct {     char name[dmDBNameLength];     uint32_t creator;     uint32_t type;     uint16_t version;     uint16_t padding;     LocalID dbID;     BitmapPtr iconP; } SysDBListItemType</pre>														
<b>Fields</b>	<table><tr><td>name</td><td>The name of the database or panel.</td></tr><tr><td>creator</td><td>The database or panel's creator ID.</td></tr><tr><td>type</td><td>The database or panel's type.</td></tr><tr><td>version</td><td>The version of the database or panel.</td></tr><tr><td>padding</td><td>Padding bytes.</td></tr><tr><td>dbID</td><td>The database or panel's local ID.</td></tr><tr><td>iconP</td><td>The database or panel's icon.</td></tr></table>	name	The name of the database or panel.	creator	The database or panel's creator ID.	type	The database or panel's type.	version	The version of the database or panel.	padding	Padding bytes.	dbID	The database or panel's local ID.	iconP	The database or panel's icon.
name	The name of the database or panel.														
creator	The database or panel's creator ID.														
type	The database or panel's type.														
version	The version of the database or panel.														
padding	Padding bytes.														
dbID	The database or panel's local ID.														
iconP	The database or panel's icon.														

## System Utilities Constants

### Miscellaneous System Utilities Constants

<b>Purpose</b>	The System Utilities header file defines the following constants.
<b>Declared In</b>	SysUtils.h
<b>Constants</b>	<pre>#define sysRandomMax 0x7FFF</pre> <p>The upper limit of random number generation. The <a href="#">SysRandom()</a> function generates pseudo-random integers where <math>0 \leq \text{randomInteger} \leq \text{sysRandomMax}</math>.</p>

# System Utilities Functions and Macros

## Abs Macro

<b>Purpose</b>	Calculate the absolute value of a numeric argument.
<b>Declared In</b>	<code>SysUtils.h</code>
<b>Prototype</b>	<code>#define Abs (a)</code>
<b>Parameters</b>	→ <i>a</i> A numeric value.
<b>Returns</b>	Returns the argument if it is greater than or equal to zero, or the argument multiplied by -1 if it is less than zero.

## SysAreWeRunningUI Function

<b>Purpose</b>	Determine whether or not the calling thread is running a UI context.
<b>Declared In</b>	<code>SysUtils.h</code>
<b>Prototype</b>	<code>Boolean SysAreWeRunningUI (void)</code>
<b>Parameters</b>	None.
<b>Returns</b>	Returns <code>true</code> if this thread is running a UI context. This is true for the main UI thread and any thread that is currently inside a UI context from <a href="#">WinStartThreadUI()</a> .
<b>See Also</b>	<a href="#">SysAreWeUIThread()</a>

## SysAreWeUIThread Function

<b>Purpose</b>	Determine whether or not the calling thread is the UI thread.
<b>Declared In</b>	<code>SysUtils.h</code>
<b>Prototype</b>	<code>Boolean SysAreWeUIThread (void)</code>
<b>Parameters</b>	None.
<b>Returns</b>	Returns <code>true</code> if the calling thread is the UI thread, <code>false</code> if not.

## SysBatteryInfoV40 Function

**Purpose** Retrieve settings for the batteries. Set `set` to `false` to retrieve battery settings.

---

**WARNING!** Use this function only to *retrieve* settings! Applications should *not* change any of the battery settings.

---

**Declared In** SysUtils.h

**Prototype** `uint16_t SysBatteryInfoV40 (Boolean set, uint16_t *warnThresholdP, uint16_t *criticalThresholdP, uint32_t *maxTimeP, SysBatteryKind *kindP, Boolean *pluggedIn, uint8_t *percentP)`

**Parameters**

- `set`  
If `false`, parameters with non-NULL pointers are retrieved. Never set this parameter to `true`.
- ↔ `warnThresholdP`  
Pointer to battery voltage warning threshold in volts\*100, or NULL.
- ↔ `criticalThresholdP`  
Pointer to the battery voltage critical threshold in volts\*100, or NULL.
- ↔ `maxTimeP`  
Pointer to the battery timeout, or NULL.
- ↔ `kindP`  
Pointer to the battery type, or NULL. For the complete set of battery types, see “[SysBatteryKindTag](#)” on page 166.
- ↔ `pluggedIn`  
Pointer to `pluggedIn` return value, or NULL.
- ↔ `percentP`  
Percentage of power remaining in the battery.

**Returns** Returns the current battery voltage in volts\*100.

**Comments** Call this function to make sure an upcoming activity won't be interrupted by a low battery warning.

`warnThresholdP` and `maxTimeP` are the battery-warning voltage threshold and time out. If the battery voltage falls below the

## System Utilities

### *SysBinarySearch*

---

threshold, or the timeout expires, a `lowBatteryChr` key event is put on the queue. Normally, applications call [SysHandleEvent\(\)](#) which calls `SysBatteryDialog()` in response to this event.

*criticalThresholdP* is the battery voltage threshold. If battery voltage falls below this level, the system turns itself off without warning and doesn't turn on until battery voltage is above it again.

## SysBinarySearch Function

<b>Purpose</b>	Search elements in a sorted array for the specified data according to the specified comparison function.
<b>Declared In</b>	<code>SysUtils.h</code>
<b>Prototype</b>	<pre>Boolean SysBinarySearch (void const *baseP,     const uint16_t numElements,     const int16_t width, SearchFuncPtr searchF,     void const *searchData, const int32_t other,     int32_t *position, const Boolean findFirst)</pre>
<b>Parameters</b>	<ul style="list-style-type: none"><li>→ <i>baseP</i> Base pointer to an array of elements.</li><li>→ <i>numElements</i> Number of elements to search. Must be greater than 0.</li><li>→ <i>width</i> Width of each array element.</li><li>→ <i>searchF</i> Search function.</li><li>→ <i>searchData</i> Data to search for. This data is passed to the <i>searchF</i> function.</li><li>→ <i>other</i> Data to be passed as the third parameter (the <i>other</i> parameter) to the comparison function.</li><li>→ <i>position</i> Pointer to the position result.</li></ul>

→ *findFirst*

If set to `true`, the first matching element is returned. Use this parameter if the array contains duplicate entries to ensure that the first such entry will be the one returned.

**Returns** Returns `true` if an exact match was found. In this case, *position* points to the element number where the data was found.

Returns `false` if an exact match was not found. If `false` is returned, *position* points to the element number where the data should be inserted if it was to be added to the array in sorted order.

**Comments** The array must be sorted in ascending order prior to the search. Use [SysInsertionSort\(\)](#) or [SysQSort\(\)](#) to sort the array.

The search starts at element 0 and ends at element (*numOfElements* - 1).

The search function's (*searchF*) prototype is:

---

```
int16_t _searchF (void const *searchData,  
void const *arrayData, int32_t other);
```

---

The first parameter is the data for which to search, the second parameter is a pointer to an element in the array, and the third parameter is any other necessary data.

The function should return:

- > 0 if the search data is greater than the pointed-to element.
- < 0 if the search data is less than the pointed-to element.
- 0 if the search data is the same as the pointed-to element.

## SysCopyStringResource Function

**Purpose** Obtain a copy of a resource string.

**Declared In** SysUtils.h

**Prototype** `void SysCopyStringResource (char *string,  
DmOpenRef dbRef, DmResourceID resID)`

**Parameters** ← *string*  
Location to which the resource string is to be copied.

## System Utilities

### *SysCopyStringResourceV50*

---

→ *dbRef*

Reference to an open resource database that contains the resource string.

→ *resID*

ID of the resource string to be copied.

**Returns** Returns nothing.

## SysCopyStringResourceV50 Function

**Purpose** Obtain a copy of a resource string.

**Declared In** SysUtils.h

**Prototype** void SysCopyStringResourceV50 (char \*string,  
DmResourceID resID)

**Parameters** → *string*

Location to which the resource string is to be copied.

→ *resID*

ID of the resource string to be copied.

**Returns** Returns nothing.

**Compatibility** This function is provided for compatibility purposes. It searches the resource chain for a resource with the specified ID. New and updated applications should use [SysCopyStringResource\(\)](#) instead.

## SysCreateDataBaseList Function

**Purpose** Generate a list of databases matching a specified type and creator criterion, and return the result.

**Declared In** SysUtils.h

**Prototype** Boolean SysCreateDataBaseList (uint32\_t type,  
uint32\_t creator, uint16\_t \*dbCount,  
MemHandle \*dbIDs, Boolean lookupName,  
dmFindType findType)

**Parameters** → *type*

The type of database to find. Supply 0 to find databases of any type.

→ *creator*

The creator ID of the database to find. Supply 0 to find databases with any creator ID.

← *dbCount*

A pointer to an integer value that is updated by this function to the number of matching databases.

→ *dbIDs*

A pointer to a handle that gets allocated to contain the database list. Upon return, this references an array of [SysDBListItemType](#) structures. See the Comments section below for more information.

→ *lookupName*

If `true`, this function uses a name in a `tAIN` resource instead of the database's name, and it sorts the list.

→ *findType*

Flags indicating the type of database to be searched for: schema, extended, classic, or a combination of the three. See [DmFindType](#) (in *Exploring Palm OS: Memory, Databases, and Files*) for more information.

**Returns** Returns `false` if no databases were found, and `true` if any databases were found. The value of *dbCount* is updated to reflect the number of databases that were found. If at least one database is found, *dbIDs* is updated to reference a array of [SysDBListItemType](#) structures; this array contains *dbCount* items.

**Comments** This function creates a list of unique databases, where unique is defined as having a different type and creator ID. Two or more databases with the same type and creator ID are counted as one. Thus, you cannot use `SysCreateDataBaseList()` to build a list of databases that share a common type and creator. There are two exceptions to this rule, however. If *type* is 0 or if *creator* is not 0, the code that removes "non-unique" databases isn't run. It also isn't run if the type is `sysFileTpqa`, since web-clipping databases all have the same type and creator ID.

Only the last version of a database is returned. Databases with multiple versions are listed only once.

## System Utilities

### *SysCreateDataBaseListV50*

---

If this function returns true and the value of *dbCount* is greater than 0, then you can iterate through the list of database items, as shown in [Listing 37.1](#)

#### Listing 37.1 Using the SysCreateDatabaseList function

---

```
SysDBListItemType *dbListIDsP;
MemHandle dbListIDsH;
UInt16 dbCount = 0;
Boolean status;
UInt16 counter;
SysDBListItemType theItem;

status = SysCreateDatabaseList(sysFileTpqa, 0, &dbCount,
    &dbListIDsH, false);

if (status == true && dbCount > 0) {
    dbListIDsP = MemHandleLock (dbListIDsH);
    for (counter = 0; counter < dbCount; counter++)
        if (StrCompare(dbListIDsP[counter].name, "MINE") == 0)
            // we found my database
            MemPtrFree (dbListIDsP);
}
```

---

**NOTE:** It is your responsibility to free the memory allocated by this function for the list of databases.

---

**See Also** [SysCreatePanelList\(\)](#)

## SysCreateDataBaseListV50 Function

- Purpose** Generate a list of Classic databases matching a specified type and creator criterion, and return the result.
- Declared In** SysUtils.h
- Prototype** Boolean SysCreateDataBaseListV50(uint32\_t type, uint32\_t creator, uint16\_t \*count, MemHandle \*dbIDs, Boolean lookupName)
- Parameters** → *type*  
The type of database to find. Supply 0 to find databases of any type.

→ *creator*

The creator ID of the database to find. Supply 0 to find databases with any creator ID.

← *count*

A pointer to an integer value that is updated by this function to the number of matching databases.

→ *dbIDs*

A pointer to a handle that gets allocated to contain the database list. Upon return, this references an array of [SysDBListItemType](#) structures. See the Comments section below for more information.

→ *lookupName*

If `true`, this function uses a name in a `tAIN` resource instead of the database's name, and it sorts the list.

**Returns** Returns `false` if no databases were found, and `true` if any databases were found. The value of *dbCount* is updated to reflect the number of databases that were found. If at least one database is found, *dbIDs* is updated to reference a array of [SysDBListItemType](#) structures; this array contains *dbCount* items.

**Comments** This function is provided for compatibility purposes only. Palm OS Cobalt applications should use [SysCreateDataBaseList\(\)](#) instead.

For additional comments, see the Comments section under [SysCreateDataBaseList\(\)](#).

## SysCreatePanelList Function

**Purpose** Generate a list of panels and return the result. Multiple versions of a panel are listed once.

**Declared In** `SysUtils.h`

**Prototype** `Boolean SysCreatePanelList (uint16_t *panelCount, MemHandle *panelIDs)`

**Parameters** ← *panelCount*  
The number of panels in the list.

## System Utilities

### *SysErrString*

---

→ *panelIDs*

A pointer to a handle that gets allocated to contain the panel list. Upon return, this references an array of [SysDBListItemType](#) structures.

**Returns** Returns `false` if no panels were found, and `true` if any panels were found. The value of *panelCount* is updated to reflect the number of panels that were found. If at least one panel is found, *panelIDs* is updated to reference a array of [SysDBListItemType](#) structures; this array contains *panelCount* items.

**Comments** If this function returns `true` and the value of *panelCount* is greater than 0, than you can iterate through the list of panel items, as shown in [Listing 37.1](#). It is your responsibility to free the memory allocated for the panel list.

**See Also** [SysCreateDataBaseList\(\)](#)

## SysErrString Function

**Purpose** Given a system error number, this function looks up the textual description of the error in the appropriate List resource and creates a string that can be used to display that error.

**Declared In** `SysUtils.h`

**Prototype** `char *SysErrString (status_t err, char *strP, uint16_t maxLen)`

**Parameters** → *err*

Error number.

← *strP*

Pointer to a buffer into which the error text will be written.

→ *maxLen*

Length of the *strP* buffer, in bytes.

**Returns** Returns a pointer to the error text.

**Comments** The actual string will be of the form: "<error message> (XXXX)" where XXXX is the error number in hex.

This function looks for a resource of type 'tst1' and resource ID of (*err*>>8). It then copies the string at index (*err* & 0x00FF) out of that resource.

---

**NOTE:** The first string in the resource is index #1 in Constructor, NOT #0. For example, an error code of 0x0101 will fetch the first string in the resource.

---

**See Also** [SysErrStringTextOnly\(\)](#)

## SysErrStringTextOnly Function

**Purpose** Given a system error number, this function looks up the textual description of the error in the appropriate List resource and creates a string that can be used to display that error.

**Declared In** SysUtils.h

**Prototype** `char *SysErrStringTextOnly (status_t err,  
char *strP, uint16_t maxLen)`

**Parameters** → *err*

Error number.

← *strP*

Pointer to a buffer into which the error text will be written.

→ *maxLen*

Length of the *strP* buffer, in bytes.

**Returns** Returns a pointer to the error text.

**Comments** Unlike [SysErrString\(\)](#), this function only returns the error message obtained from the list resource. The error number isn't included in the resulting string.

This function looks for a resource of type 'tst1' and resource ID of (*err*>>8). It then copies the string at index (*err* & 0x00FF) out of that resource.

---

**NOTE:** The first string in the resource is index #1 in Constructor, NOT #0. For example, an error code of 0x0101 will fetch the first string in the resource.

---

## System Utilities

### *SysFormPointerArrayToStrings*

---

#### **SysFormPointerArrayToStrings Function**

- Purpose** Form an array of pointers to strings packed in a block. Useful for setting the items of a list.
- Declared In** `SysUtils.h`
- Prototype** `MemHandle SysFormPointerArrayToStrings (char *c, int16_t stringCount)`
- Parameters**
- *c*  
Pointer to packed block of strings, each terminated by a null character.
  - *stringCount*  
The number of strings in the block.
- Returns** Unlocked handle to an allocated array of pointers to the strings in the passed block. The returned array points to the strings in the original block. Note that you'll need to free the returned handle when you no longer need it.
- See Also** [LstSetListChoices\(\)](#)

#### **SysGetOSVersionString Function**

- Purpose** Return the operating system version number.
- Declared In** `SysUtils.h`
- Prototype** `char *SysGetOSVersionString (void)`
- Parameters** None.
- Returns** Returns a string such as "v. 6.0."
- Comments** You must free the returned string after you no longer need it.

## SysGetROMTokenV40 Function

- Purpose** Get a ROM token value.
- Declared In** `SysUtils.h`
- Prototype**  
`status_t SysGetROMTokenV40 (uint16_t cardNo,  
uint32_t token, uint8_t **dataP,  
uint16_t *sizeP)`
- Parameters**
- *cardNo*  
The card on which the ROM to be queried resides. This value must be 0.
  - *token*  
The value to retrieve, as specified by one of the tokens listed under “[ROM Tokens](#)” on page 434.
  - ← *dataP*  
Pointer to a buffer that holds the requested value when the function returns.
  - ← *sizeP*  
The number of bytes in the *dataP* buffer.
- Returns** Returns `errNone` if the requested token was successfully retrieved, or an error code if an error occurred. If this function returns an error, or if the returned pointer to the buffer is `NULL`, or if the first byte of the text buffer is `0xFF`, then no serial number is available.

## SysInsertionSort Function

- Purpose** Using an insertion sort algorithm, sort elements in an array according to the passed comparison function.
- Declared In** `SysUtils.h`
- Prototype**  
`void SysInsertionSort (void *baseP,  
uint16_t numOfElements, int16_t width,  
const CmpFuncPtr comparF, const int32_t other)`
- Parameters**
- ↔ *baseP*  
Base pointer to an array of elements.
  - *numOfElements*  
Number of elements to sort (must be at least 2).

## System Utilities

### *SysInsertionSort*

---

→ *width*

Width of each element.

→ *comparF*

Comparison function. See Comments, below.

→ *other*

Other data passed to the comparison function.

**Returns** Returns nothing.

**Comments** Only elements which are out of order move. Moved elements are moved to the end of the range of equal elements. If a large amount of elements are being sorted, [SysQSort\(\)](#) may be faster.

This is the insertion sort algorithm: Starting with the second element, each element is compared to the preceding element. Each element not greater than the last is inserted into sorted position within those already sorted. A binary search for the insertion point is performed. A moved element is inserted after any other equal elements.

In order to use `SysInsertionSort()` you must write a comparison function with the following prototype:

---

```
int16_t _comparF (void *p1, void *p2, int32_t other);
```

---

Your comparison function must return zero if *p1* equals *p2*, a positive integer value if *p1* is greater than *p2*, and a negative integer value if *p1* is less than *p2*. Note that the value of the parameter named *other* is passed through from the `SysInsertionSort()` call and can be used to control the behavior of the *comparF* function if appropriate for your application.

**See Also** [SysBinarySearch\(\)](#), [SysQSort\(\)](#)

## SysQSort Function

- Purpose** Using a quicksort algorithm, sort elements in an array according to the supplied comparison function.
- Declared In** `SysUtils.h`
- Prototype** `void SysQSort (void *baseP,  
uint16_t numOfElements, int16_t width,  
const CmpFuncPtr comparF, const int32_t other)`
- Parameters**
- `↔ baseP`  
Base pointer to an array of elements.
  - `→ numOfElements`  
Number of elements to sort (must be at least 2).
  - `→ width`  
Width of each element.
  - `→ comparF`  
Comparison function. See Comments, below.
  - `→ other`  
Other data passed to the comparison function.
- Returns** Returns nothing.
- Comments** Equal records can be in any position relative to each other because a quick sort tends to scramble the ordering of records. As a result, calling `SysQSort ( )` multiple times can result in a different order if the records are not completely unique. If you don't want this behavior, use [SysInsertionSort \( \)](#) instead.
- To pick the pivot point, the quick sort algorithm picks the middle of three records picked from around the middle of all records. That way, the algorithm can take advantage of partially sorted data.
- These optimizations are built in:
- The function contains its own stack to limit uncontrolled recursion. When the stack is full, an insertion sort is used because it doesn't require more stack space.
  - An insertion sort is also used when the number of records is low. This avoids the overhead of a quick sort which is noticeable for small numbers of records.

## System Utilities

### *SysRandom*

---

- If the records seem mostly sorted, an insertion sort is performed to move only those few records that need to be moved.

In order to use `SysQSort()` you must write a comparison function with the following prototype:

```
Int16 comparF (void *p1, void *p2, Int32 other)
```

Your comparison function must return zero if *p1* equals *p2*, a positive integer value if *p1* is greater than *p2*, and a negative integer value if *p1* is less than *p2*. Note that the value of the parameter named *other* is passed through from the `SysQSort()` call and can be used to control the behavior of the *comparF* function if appropriate for your application.

**See Also** [SysBinarySearch\(\)](#), [SysInsertionSort\(\)](#)

## SysRandom Function

**Purpose** Return a random number anywhere from 0 to `sysRandomMax`.

**Declared In** `SysUtils.h`

**Prototype** `int16_t SysRandom (int32_t newSeed)`

**Parameters** `→ newSeed`  
New seed value, or 0 to use existing seed.

**Returns** Returns a random number.

## SysStringByIndex Function

**Purpose** Copy a string out of a string list resource.

**Declared In** `SysUtils.h`

**Prototype** `char *SysStringByIndex (DmOpenRef dbRef,  
DmResourceID resID, uint16_t index,  
char *strP, uint16_t maxLen)`

**Parameters** `→ dbRef`  
Reference to an open resource database that contains the string list resource.

- *resID*  
Resource ID of the string list.
- *index*  
String to get out of the list.
- ← *strP*  
Pointer to a buffer into which the string is copied.
- *maxLen*  
Size of *strP* buffer.

**Returns** Returns a pointer to the copied string. The string returned from this call will be the prefix string appended with the designated index string. Indices are 0-based; index 0 is the first string in the resource.

**Comments** String list resources are of type 'tSTL' and contain a list of strings and a prefix string.  
ResEdit always displays the items in the list as starting at 1, not 0. Consider this when creating your string list.

## **SysStringByIndexV50 Function**

**Purpose** Copy a string out of a string list resource.

**Declared In** SysUtils.h

**Prototype** `char *SysStringByIndexV50 (DmResourceID resID, uint16_t index, char *strP, uint16_t maxLen)`

**Parameters**

- *resID*  
Resource ID of the string list.
- *index*  
String to get out of the list.
- ← *strP*  
Pointer to a buffer into which the string is copied.
- *maxLen*  
Size of *strP* buffer.

**Returns** Returns a pointer to the copied string. The string returned from this call will be the prefix string appended with the designated index string. Indices are 0-based; index 0 is the first string in the resource.

## System Utilities

### *SysStringByIndexV50*

---

**Comments** String list resources are of type 'tSTL' and contain a list of strings and a prefix string.

ResEdit always displays the items in the list as starting at 1, not 0. Consider this when creating your string list.

**Compatibility** This function is provided for compatibility purposes. It searches the resource chain for a string list resource with the specified ID. New and updated applications should use [SysStringByIndex\(\)](#) instead.

# Time Manager

---

This chapter provides reference material for those APIs used to get and set the handheld's clock, and to get the amount of time elapsed since the handheld was last reset. It is organized as follows:

<a href="#">Time Manager Constants</a> . . . . .	505
<a href="#">Time Manager Functions and Macros</a> . . . . .	506

The header file `TimeMgr.h` declares the API that this chapter describes. For more information on using the Time Manager APIs, see [Chapter 11, "Time,"](#) on page 107.

## Time Manager Constants

### Time Manager Error Codes

<b>Purpose</b>	Error codes returned by the various Time manager functions.
<b>Declared In</b>	<code>TimeMgr.h</code>
<b>Constants</b>	<pre>#define timErrBadParam (timErrorClass   3)     Returned from <a href="#">TimSetSeconds()</a> if the specified date and     time is outside the range of dates and times that the device     can handle.  #define timErrMemory (timErrorClass   1)     Insufficient memory.  #define timErrNoLunarCalendarSupport     (timErrorClass   2)</pre>

# Time Manager Functions and Macros

## TimGetSeconds Function

- Purpose** Return the current date and time of the device in seconds since 12:00 A.M. on January 1, 1904.
- Declared In** `TimeMgr.h`
- Prototype** `uint32_t TimGetSeconds (void)`
- Parameters** None.
- Returns** The number of seconds elapsed from 12:00 A.M. on January 1, 1904 to the current date and time on the device.
- See Also** `TimSetSeconds()`

## TimGetTicks Function

- Purpose** Return the tick count since the last reset. The tick count does not advance while the device is in sleep mode.
- Declared In** `TimeMgr.h`
- Prototype** `uint64_t TimGetTicks (void)`
- Parameters** None.
- Returns** Returns the tick count.
- Comments** You can call the [SysTicksPerSecond\(\)](#) routine to determine the number of ticks per second.

## TimInit Function

- Purpose**
- Declared In** `TimeMgr.h`
- Prototype** `status_t TimInit (void)`
- Parameters** None.
- Returns**

## **TimSetSeconds Function**

- Purpose** Set the clock of the device to the date and time passed as the number of seconds since 12:00 A.M. on January 1, 1904.
- Declared In** `TimeMgr.h`
- Prototype** `status_t TimSetSeconds (uint32_t seconds)`
- Parameters** `→ seconds`  
The number of seconds since 12:00 A.M. on January 1, 1904.
- Returns** Returns `errNone` if the operation completed successfully, or `timErrBadParam` if the specified date and time is outside the range of dates and times that the device can handle. Note that this range is defined by each Palm Powered device manufacturer, so the behavior of this function with certain dates may vary from one device to another.
- Comments** This function broadcasts the `sysNotifyTimeChangeEvent` to all interested parties.
- See Also** `TimGetSeconds()`

## **Time Manager**

*TimSetSeconds*

---

# Trace Manager

---

The Trace Manager allows you to output a buffer of data, in hex, or a string to the external trace reporting tool.

The contents of this chapter are organized as follows:

[Trace Manager Functions and Macros](#) . . . . . 509

The header file `TraceMgr.h` declares the API that this chapter describes.

For more information on debugging Palm OS applications, see [Chapter 13, “Debugging Strategies,”](#) on page 113.

## Trace Manager Functions and Macros

### TM Macro

<b>Purpose</b>	Provides a shorthand method for calling the Trace Manager functions. This macro is identical to <a href="#">TraceOutput()</a> .
<b>Declared In</b>	<code>TraceMgr.h</code>
<b>Prototype</b>	<code>#define TM (X)</code>
<b>Parameters</b>	→ <i>X</i> The portion of the function name following <code>TraceOutput</code> that indicates the type of the value being output.
<b>Returns</b>	Returns nothing.
<b>Comments</b>	Rather than calling the Trace Manager functions directly, you can use this macro instead.

## Trace Manager

### *TmOutputB*

---

To call...	Use...
<a href="#">TmOutputT()</a>	TM(T ( <i>errorClass</i> , "format", ... ) )
<a href="#">TmOutputTL()</a>	TM(TL ( <i>errorClass</i> , "format", ... ) )
<a href="#">TmOutputB()</a>	TM(B ( <i>errorClass</i> , <i>addr</i> , <i>count</i> ) )
<a href="#">TmOutputVT()</a>	TM(VT ( <i>errorClass</i> , "format", <i>va_list</i> ) )
<a href="#">TmOutputVTL()</a>	TM(VTL( <i>errorClass</i> , "format", <i>va_list</i> ) )

**See Also** [TraceOutput\(\)](#)

## TmOutputB Function

<b>Purpose</b>	Output a buffer of data, in hex dump format, to the external trace reporting tool.
<b>Declared In</b>	TraceMgr.h
<b>Prototype</b>	<pre>void TmOutputB (status_t traceModule,                 const void *aBuffer, long aBufferLen)</pre>
<b>Parameters</b>	<p>→ <i>traceModule</i> The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in <code>CmnErrors.h</code>. Applications should use <code>appErrorClass</code>.</p> <p>→ <i>aBuffer</i> Pointer to a buffer of raw data to be output.</p> <p>→ <i>aBufferLen</i> The number of bytes of data in the buffer.</p>
<b>Returns</b>	Returns nothing.
<b>See Also</b>	<a href="#">TmOutputT()</a> , <a href="#">TmOutputTL()</a> , <a href="#">TmOutputVT()</a> , <a href="#">TmOutputVTL()</a>

## TmOutputT Function

- Purpose** Output a formatted text string to the external trace reporting tool.
- Declared In** TraceMgr.h
- Prototype** void TmOutputT (status\_t traceModule,  
const char \*aFormatString, ...)
- Parameters**
- *traceModule*  
The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in CmnErrors.h. Applications should use appErrorClass.
  - *aFormatString*  
A format string, as used in standard C-library calls such as printf(). See the Comments section, below, for more detail.
  - ...  
The list of variables to be formatted for output.
- Returns** Returns nothing.
- Comments** The format string has the following form:

% *flags width type*

[Table 39.1](#) shows the flag types that you can use in the format specification for the tracing output functions.

**Table 39.1 Trace function format specification flags**

Flag	Description
-	Left-justified output. By default, output is right-justified.
+	Always display the sign symbol. By default, only a leading minus sign is displayed.
space	Display a space when the value is positive, rather than a '+' symbol.

[Table 39.2](#) shows the output types that you can use in the format specification for the tracing output functions.

**Table 39.2 Trace function format specification types**

Flag	Description
%	Display the '%' character.
s	Display a null-terminated string value.
c	Display a character value.
ld	Display an <code>int32_t</code> value.
lu	Display a <code>uint32_t</code> value.
lx or lX	Display a <code>uint32_t</code> value in hexadecimal.
hd	Display an <code>int16_t</code> value.
hu	Display a <code>uint16_t</code> value.
hx or hX	Display a <code>uint16_t</code> value in hexadecimal.

**See Also** [TmOutputB\(\)](#), [TmOutputTL\(\)](#), [TmOutputVT\(\)](#), [TmOutputVTTL\(\)](#)

## TmOutputTL Function

**Purpose** Output a formatted text string, followed by a newline, to the external trace reporting tool. This function performs the same operation as [TmOutputT\(\)](#) but adds the newline character.

**Declared In** `TraceMgr.h`

**Prototype** `void TmOutputTL (status_t traceModule,  
const char *aFormatString, ...)`

**Parameters**  $\rightarrow$  `traceModule`

The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`. Applications should use `appErrorClass`.

→ *aFormatString*

A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [TmOutputT\(\)](#) for more detail.

→ ...

The list of variables to be formatted for output.

**Returns** Returns nothing.

**See Also** [TmOutputB\(\)](#), [TmOutputVT\(\)](#), [TmOutputVTL\(\)](#)

## TmOutputVT Function

**Purpose** Output a formatted text string to the external trace reporting tool.

**Declared In** `TraceMgr.h`

**Prototype** `void TmOutputVT (status_t traceModule,  
const char *aFormatString, va_list arglist)`

**Parameters** → *traceModule*

The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`. Applications should use `appErrorClass`.

→ *aFormatString*

A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [TmOutputT\(\)](#) for more detail.

→ *arglist*

A structure containing the variable argument list. This is the same kind of variable argument list used for standard C-library functions such as `vprintf()`.

**Returns** Returns nothing.

**See Also** [TmOutputB\(\)](#), [TmOutputTL\(\)](#), [TmOutputVTL\(\)](#)

## TmOutputVTL Function

- Purpose** Output a formatted text string, followed by a newline, to the external trace reporting tool. This function performs the same operation as [TmOutputVT\(\)](#) but adds the newline character.
- Declared In** `TraceMgr.h`
- Prototype** `void TmOutputVTL (status_t traceModule,  
const char *aFormatString, va_list arglist)`
- Parameters**
- *traceModule*  
The ID of the Palm OS subsystem from which this output originates. You can use this with the external tracing tool to filter traces according to their origin. The ID must match one of the error classes defined in `CmnErrors.h`. Applications should use `appErrorClass`.
  - *aFormatString*  
A format string, as used in standard C-library calls such as `printf()`. See the Comments section under [TmOutputT\(\)](#) for more detail.
  - *arglist*  
A structure containing the variable argument list. This is the same kind of variable argument list used for standard C-library functions such as `vprintf()`.
- Returns** Returns nothing.
- See Also** [TmOutputB\(\)](#), [TmOutputT\(\)](#), [TmOutputTL\(\)](#)

## TraceDefine Macro

- Purpose** Constructs a custom error class value.
- Declared In** `TraceMgr.h`
- Prototype** `#define TraceDefine (x, y)`
- Parameters**
- *x*  
The error class “base” value.
  - *y*  
The offset from the “base” value.
- Returns** This macro evaluates to the sum of the two supplied values.

**Comments**    Use this macro to construct custom error class values.

## TraceOutput Macro

**Purpose**    Provides a shorthand method for calling the Trace Manager functions. This macro is identical to [TM\(\)](#).

**Declared In**    `TraceMgr.h`

**Prototype**    `#define TraceOutput (X)`

**Parameters**    `→ X`

The portion of the function name following `TraceOutput` that indicates the type of the value being output.

**Returns**    Returns nothing.

**Comments**    Rather than calling the Trace Manager functions directly, you can use this macro instead.

**To call...**

**Use...**

[TmOutputT\(\)](#)    `TraceOutput(T (errorClass, "format", ...) )`

[TmOutputTL\(\)](#)    `TraceOutput(TL (errorClass, "format", ...) )`

[TmOutputB\(\)](#)    `TraceOutput(B (errorClass, addr, count) )`

[TmOutputVT\(\)](#)    `TraceOutput(VT (errorClass, "format", va_list) )`

[TmOutputVTL\(\)](#)    `TraceOutput(VTL(errorClass, "format", va_list) )`

**See Also**    [TM\(\)](#)

## **Trace Manager**

*TraceOutput*

---

# Index

---

## A

- Abs() 488
- actvErrorClass 169
- Alarm Manager
  - alarm sound 24
  - and Attention Manager 11, 20
  - reminder dialog boxes 24
- alarms
  - and expansion 61
  - canceling 127
  - delayed in sleep mode 25
  - displaying 20
  - multiple 4
  - playing a sound 24
  - setting 25, 127
- AlmEnableNotification() 126
- almErrFull 124, 127
- almErrMemory 124, 127
- almErrorClass 169
- AlmGetAlarm 26
- AlmGetAlarm() 126
- almProcCmdCustom 124
- AlmProcCmdEnum 122
- AlmProcCmdEnumTag 124
- almProcCmdReschedule 124
- almProcCmdTriggered 124
- AlmSetAlarm 25, 26
- AlmSetAlarm() 127
- amErrorClass 169
- appErrorClass 169, 239, 244
- application preferences 416
- application-defined features 33
- appMgrErrorClass 169
- april 191
- architecture
  - expansion 55
- attention indicator 5, 9
  - enabling and disabling 23
- Attention Manager 3–23
  - and Alarm Manager 11, 20
  - and existing applications 11, 20
  - and HotSync 19, 21
  - and SMS application 13
  - attention indicator 5, 9, 23
  - callback function 11
  - deleting items 21
  - detail dialog 6
  - dismissing items 7
  - displaying 10, 24
  - displaying alarms 20
  - enumerating items 19
  - getting the user's attention 11
  - go to item 8
  - insistent items 5, 6, 21
  - invalid items 18
  - list dialog 6
  - multiple items 10, 19
  - operation 5
  - redrawing items 22
  - sleep mode operation 22
  - snoozing items 7, 19
  - sounds 17
  - special effects 17
  - subtle items 5, 8, 21
  - triggering a custom effect 17
  - updating items 20, 21
  - validating items 19
- Attention Manager commands 13, 17
  - draw detail 14
  - draw list 15
  - go there 17
  - got it 18
  - iterate 19
  - play sound 17
  - snooze 19
  - trigger special effects 19
- Attention Manager dialogs
  - detail dialog 6
  - drawing 13
  - list dialog 6
  - text and background colors 15
- AttnCommandArgsDrawDetailTag 129
- AttnCommandArgsDrawListTag 130
- AttnCommandArgsGotItTag 132
- AttnCommandArgsIterateTag 132
- AttnCommandArgsType 133
- AttnCommandType 135
- AttnDoSpecialEffects 20
- AttnDoSpecialEffects() 145
- attnErrItemNotFound 138

---

attnErrMemory 138  
attnErrorClass 169  
AttnFlagsType 135  
AttnForgetIt 16, 18, 19, 21  
AttnForgetIt() 146  
AttnForgetItV40() 146  
AttnGetAttention 11, 17, 20  
AttnGetAttention() 148  
AttnGetAttentionV40() 149  
AttnGetCounts 20  
AttnGetCounts() 152  
AttnGetCountsV40() 152  
AttnIndicatorEnable() 154  
AttnIndicatorEnabled() 155  
AttnIterate 19  
AttnIterate() 155  
AttnIterateV40() 156  
AttnLaunchCodeArgsType 136  
AttnLevelType 137  
AttnListOpen 24  
AttnListOpen() 157  
AttnNotifyDetailsType 138  
AttnUpdate 21  
AttnUpdate() 157  
AttnUpdateV40() 159  
august 191  
auto-off 103  
azmErrorClass 170

## **B**

Background  
    launching in 93  
Background Process 88, 93  
Background process 88  
Background threads 88  
base 10 form of floating-point number 289  
battery 104  
    conservation using modes 103  
    life, maximizing 102  
battery timeout 436, 489  
battery voltage warning threshold 436, 489  
block device driver  
    plug and play 55

bltErrorClass 170  
blthErrorClass 170  
bndErrBadInterface 176  
bndErrBadTransact 176  
bndErrBadType 176  
bndErrDead 176  
bndErrMissingArg 176  
bndErrorClass 170  
bndErrOutOfStack 176  
bndErrTooManyLoopers 176  
bndErrUnknownMethod 176  
bndErrUnknownProperty 177  
bndErrUnknownTransact 177  
booting 83  
BUILD\_TYPE 114

## **C**

canceling alarms 127  
card insertion and removal 61  
CardMetricsType 257  
cards  
    expansion 55  
catmErrorClass 170  
CatMgrSyncGetModifiedCategories() 161  
CatMgrSyncGetPurgeCounter() 163  
CatMgrSyncReleaseStorage() 163  
certErrorClass 170  
clock, real-time 107  
cmpErrorClass 170  
CmpFuncPtr 486  
cncErrorClass 170  
Commands 143  
CompactFlash 54  
conserving battery using modes 103  
Conversions 192  
cpmErrorClass 170  
Crc16CalcBigBlock() 183  
Crc16CalcBlock() 184  
Crc32CalcBlock() 184

## **D**

dalErrorClass 170  
Data Manager

---

and the VFS Manager 58  
date and time manager 107  
DateAdjust() 202  
DateDaysToDate() 202  
DateFormatTag 195  
DateFormatType 188  
DatePtr 189  
DateSecondsToDate() 203  
dateStringLength 191, 207  
dateTemplateChar 193  
dateTemplateDayNum 197  
dateTemplateDOWNName 197  
dateTemplateLeadZeroModifier 193  
dateTemplateLongModifier 194  
dateTemplateMonthName 197  
dateTemplateMonthNum 197  
dateTemplateRegularModifier 194  
dateTemplateShortModifier 194  
DateTemplateToAscii() 203  
dateTemplateYearNum 197  
DateTimePtr 188  
DateTimeType 188  
DateToAscii() 206  
DateToDays() 207  
DateToDOWDMFormat() 208  
DateToInt() 209  
DateType 189  
DaylightSavingsTag 197  
DaylightSavingsTypes 189  
DayOfMonth() 209  
DayOfMonthType 189  
DayOfWeek() 210  
DayOfWeekTag 198  
DayOfWeekType 189, 190  
Days 192  
daysInFourYears 192  
daysInLeapYear 193  
DaysInMonth() 210  
daysInSeconds 193  
daysInWeek 193  
daysInYear 193  
DbgBreak() 219  
DbgBreakMessage() 220  
DbgBreakMessageIf() 220  
DbgFatalErrorInContext() 221  
DbgGetChar() 221  
DbgIsPresent() 222  
DbgLookupSymbol() 222  
DbgMessage() 223  
DbgOnlyFatalError() 238  
DbgOnlyFatalErrorIf() 238  
DbgOutputSync() 223  
DbgPrintf() 223  
DbgRestartMallocProfiling() 224  
DbgSetMallocProfiling() 224  
DbgUnmangleSymbol() 225  
DbgVPrintf() 225  
DbgWalkStack() 226  
december 192  
defaultAutoOffDuration 408  
defaultAutoOffDurationSecs 412  
dfDMYLong 195  
dfDMYLongNoDay 196  
dfDMYLongWithComma 196  
dfDMYLongWithDot 195  
dfDMYWithDashes 195  
dfDMYWithDots 195  
dfDMYWithSlashes 195  
dfMDYLongWithComma 195  
dfMDYWithDashes 196  
dfMDYWithSlashes 195  
dfMYMed 196  
dfMYMedNoPost 196  
dfYMDLongWithDot 196  
dfYMDLongWithSpace 196  
dfYMDWithDashes 195  
dfYMDWithDots 195  
dfYMDWithSlashes 195  
dialog boxes  
    Attention Manager 6, 13  
    reminder 24  
digitizer  
    after reset 84  
    polling 107  
dirErrorClass 170  
dispErrorClass 170  
DlkCallAppReplyParamType 227

---

DlkControl() 231  
DlkCtlEnum 228  
DlkCtlEnumTag 229  
dlkCtlFirst 229  
dlkCtlGetCondFilterTable 229  
dlkCtlGetHSTCPPort 230  
dlkCtlGetLANSync 230  
dlkCtlGetPCHostAddr 230  
dlkCtlGetPCHostMask 230  
dlkCtlGetPCHostName 229  
dlkCtlGetPostSyncErr 230  
dlkCtlLAST 230  
dlkCtlSendCallAppReply 230  
dlkCtlSetCondFilterTable 230  
dlkCtlSetHSTCPPort 230  
dlkCtlSetLANSync 230  
dlkCtlSetPCHostAddr 230  
dlkCtlSetPCHostMask 230  
dlkCtlSetPCHostName 229  
dlkCtlSetPostSyncErr 230  
dlkErrIncompatibleProducts 228  
dlkErrInterrupted 228  
dlkErrLostConnection 228  
dlkErrMemory 228  
dlkErrNoSession 229  
dlkErrNPOD 229  
dlkErrorClass 170  
dlkErrParam 229  
dlkErrSizeErr 229  
dlkErrUserCan 229  
DlkGetSyncInfo() 234  
dlkMaxUserNameLength 229  
DlkSetLogEntry() 236  
dlkSyncStateAborted 231  
dlkSyncStateCompleted 231  
dlkSyncStateIncompatibleProducts 231  
dlkSyncStateInProgress 230  
dlkSyncStateLocalCan 231  
dlkSyncStateLostConnection 231  
dlkSyncStateLowMemoryOnTD 231  
dlkSyncStateNeverSynced 230  
dlkSyncStateNPOD 231  
dlkSyncStateRemoteCan 231  
DlkSyncStateType 230  
dlkUserNameBufSize 229  
dmErrorClass 170  
DmWrite 34  
dom1stFri 198  
dom1stMon 198  
dom1stSat 198  
dom1stSun 198  
dom1stThu 198  
dom1stTue 198  
dom1stWen 198  
dom2ndFri 199  
dom2ndMon 199  
dom2ndSat 199  
dom2ndSun 199  
dom2ndThu 199  
dom2ndTue 199  
dom2ndWen 199  
dom3rdFri 199  
dom3rdMon 199  
dom3rdSat 199  
dom3rdSun 199  
dom3rdThu 199  
dom3rdTue 199  
dom3rdWen 199  
dom4thFri 200  
dom4thMon 199  
dom4thSat 200  
dom4thSun 199  
dom4thThu 200  
dom4thTue 199  
dom4thWen 200  
domLastFri 200  
domLastMon 200  
domLastSat 200  
domLastSun 200  
domLastThu 200  
domLastTue 200  
domLastWen 200  
dowDateStringLength 191, 208  
dowLongDateStrLength 191, 208  
doze mode 102  
    SysTaskDelay 444

---

drivers, restarting 83  
drvErrorClass 170  
dsAustralia 197  
dsAustraliaShifted 198  
dsEasternEuropean 197  
dsGreatBritain 198  
dsMiddleEuropean 197  
dsNone 197  
dsRumania 198  
dsTurkey 198  
dsUSA 197  
dsWesternEuropean 197  
dynamic heap  
    soft reset 83

## E

ECANCEL 182  
em68kErrorClass 170  
emuErrorClass 170  
emulator  
    preference file location 349  
ENOTSUP 182  
ErrAlert() 239, 244  
ErrCatch 253, 254  
ErrCatch() 248  
ErrCatchWithAddress() 249  
ErrDisplay 115, 116  
ErrDisplay() 240  
ErrDisplayFileLineMsg() 240  
errDlgResCancel 237  
errDlgResNo 237  
errDlgResOK 237  
errDlgResRetry 237  
ErrDlgResultType 237  
errDlgResYes 237  
ErrEndCatch() 250  
ErrExceptionListAppend() 250  
ErrExceptionListGetByThreadID() 251  
ErrExceptionListRemove() 251  
ErrExceptionPtr 247  
ErrExceptionType 247  
ErrFatalDisplay() 241  
ErrFatalDisplayIf 115

ErrFatalDisplayIf() 241  
ErrFatalError() 242  
ErrFatalErrorIf() 242  
ErrFatalErrorInContext() 243  
ErrGetErrorMsg() 243  
errInfoClass 171  
ErrJumpBuf 248  
ErrLongJump() 252  
errNone 175  
ErrNonFatalDisplay() 244  
ErrNonFatalDisplayIf() 245  
error manager 113–255  
    try-and-catch mechanism 116  
errorClassMask 176  
errReportStripContext 176  
errResponseBreak68K 175  
errResponseBreakBoth 175  
errResponseBreakNative 175  
errResponseDefaultSet 175  
errResponseIgnore 175  
errResponseKillProcess 175  
errResponseShutdown 175  
errResponseSoftReset 175  
errResponseTryHighLevel 175  
ErrSetJump() 252  
ErrThrow 116  
ErrThrow() 253  
ErrThrowIf() 253  
ErrThrowWithAddress() 253  
ErrThrowWithHandler() 254  
ErrTry() 255  
evtErrorClass 171  
EvtGetEvent 103  
EWOULDBLOCK 182  
exgErrorClass 171  
expansion 53–70  
    and legacy applications 61  
    and security 67  
    applications on cards 60  
    architecture 55  
    card-launched applications 60  
    enumerating slots 68  
    file systems 57  
    lifetime of card-launched applications 60

---

- mounted volumes 67
- notifications 61, 62
- ROM 55
- slot reference number 56
- start.prc 66
- expansion cards 55
  - capabilities 69
  - checking for 67
  - in slots 69
  - insertion and removal 61
- Expansion Manager 54, 59
  - checking card capabilities 69
  - enumerating slots 68
  - functions 59
  - overriding notification handlers 63
  - purpose 59
  - registered notifications 62
  - slot reference number 56, 64
  - verifying presence of 67
- expansion slot 54
- expCapabilityHasStorage 263
- expCapabilityReadOnly 263
- expCapabilitySerial 263
- ExpCardInfo 70
- ExpCardInfo() 265
- expCardInfoStringMaxLen 264
- ExpCardInfoType 259
- ExpCardIsFilesystemSupported() 266
- ExpCardMediaType() 266
- ExpCardMetrics() 267
- ExpCardPresent 69
- ExpCardPresent() 268
- ExpCardSectorRead() 269
- ExpCardSectorWrite() 270
- expErrCardBadSector 260
- expErrCardNoSectorReadWrite 261
- expErrCardNotPresent 261
- expErrCardProtectedSector 261
- expErrCardReadOnly 261
- expErrEnumerationEmpty 261
- expErrIncompatibleAPIVer 261
- expErrInvalidSlotRefNum 261
- expErrNotEnoughPower 261
- expErrNotOpen 261
- expErrorClass 171
- expErrSlotDeallocated 261
- expErrStillOpen 261
- expErrUnimplemented 261
- expErrUnsupportedOperation 262
- expFtrIDVersion 264
- expHandledSound 264
- expHandledVolume 264
- expInvalidSlotRefNum 263
- expIteratorStart 263
- expIteratorStop 263
- expMediaType\_Any 262
- expMediaType\_CompactFlash 262
- expMediaType\_MacSim 262
- expMediaType\_MemoryStick 262
- expMediaType\_MultiMediaCard 262
- expMediaType\_PoserHost 262
- expMediaType\_RAMDisk 262
- expMediaType\_SecureDigital 262
- expMediaType\_SmartMedia 262
- expMgrVersionNum 265
- ExpSlotCustomControl() 271
- ExpSlotEnumerate 68
- ExpSlotEnumerate() 272
- ExpSlotMediaType() 273
- ExpSlotPowerCheck() 274

**F**

- FAT 57
- fatalDoNothing 277
- fatalEnter68KDebugger 277
- fatalEnterBothDebugger 277
- fatalEnterDebugger 277
- fatalReset 277
- feature manager 31–35
- feature memory 34
- features
  - application-defined 33
  - feature memory 34
  - system version 31
- features. *See* functions starting with Ftr
- february 192
- File 327
- file streaming

---

- and the VFS Manager 58
- file systems 57
  - FAT 57
  - implementation 57
  - long filenames 57
  - multiple 57
  - VFAT 57
- fileErrorClass 171
- filenames
  - long 57
- files
  - naming 57
- firstYear 194
- FlpAddDouble() 288
- FlpAddFloat() 288
- FlpBase10Info() 289
- FlpCompareDoubleEqual() 289
- FlpCompareDoubleLessThan() 290
- FlpCompareDoubleLessThanOrEqual() 291
- FlpCompareFloatEqual() 291
- FlpCompareFloatLessThan() 292
- FlpCompareFloatLessThanOrEqual() 292
- FlpCorrectedAdd() 293
- FlpCorrectedSub() 294
- FlpDivDouble() 295
- FlpDivFloat() 295
- FlpDoubleToFloat() 296
- FlpDoubleToInt32() 296
- FlpDoubleToLongDouble() 297
- FlpDoubleToLongLong() 297
- FlpDoubleToUInt32() 297
- FlpDoubleToULongLong() 298
- flpErrorClass 171
- flpErrOutOfRange 287
- FlpFloatToDouble() 298
- FlpFloatToInt32() 299
- FlpFloatToLongDouble() 299
- FlpFloatToLongLong() 299
- FlpFloatToUInt32() 300
- FlpFloatToULongLong() 300
- FlpFToA() 301
- FlpGetExponent() 301
- FlpInt32ToDouble() 301

- FlpInt32ToFloat() 302
- FlpLongDoubleToDouble() 302
- FlpLongDoubleToFloat() 302
- FlpLongLongToDouble() 303
- FlpLongLongToFloat() 303
- FlpMulDouble() 304
- FlpMulFloat() 304
- FlpNegDouble() 305
- FlpNegFloat() 305
- FlpSubDouble() 305
- FlpSubFloat() 306
- FlpUInt32ToDouble() 306
- FlpUInt32ToFloat() 307
- FlpULongLongToDouble() 307
- FlpULongLongToFloat() 307
- flpVersion 287
- flshErrorClass 171
- fplErrorClass 171
- friday 192
- ftrErrInternalErr 280
- ftrErrInvalidParam 280
- ftrErrNoSuchFeature 280, 280, 281, 282, 284, 285, 286
- ftrErrorClass 171
- FtrGet 34, 35
- FtrGet() 280
- FtrGetByIndex() 281
- FtrPtrFree() 282
- FtrPtrGet() 282
- FtrPtrNew 34
- FtrPtrNew() 283
- FtrPtrResize() 284
- FtrSet 34
- FtrSet() 285
- FtrUnregister 34
- FtrUnregister() 286

## **G**

- global variables
  - erasing 83
- grfErrorClass 171
- grmlErrorClass 171

---

## H

halErrorClass 171  
hard reset 83  
Host 323, 325, 326  
HOST\_NAME\_MAX 327  
HostAscTime() 328  
HostBool 310  
HostBoolType 310  
HostClock() 328  
HostClockType 310  
HostCloseDir() 329  
HostControl() 329  
HostControlSelectorType 310  
HostControlTrapNumber 311  
HostCTime() 329  
HostDirEntType 311  
HostDIRType 311  
HostEnteringApp() 330  
HostErr 312  
hostErrBase 323  
hostErrDirNotEmpty 325  
hostErrDiskError 323  
hostErrDiskFull 325  
hostErrExists 325  
hostErrFileNameTooLong 324  
hostErrFileNotFound 324  
hostErrFileTooBig 324  
hostErrInvalidDeviceType 323  
hostErrInvalidParameter 323  
hostErrInvalidRAMSize 323  
hostErrIsDirectory 325  
hostErrMemInvalidPtr 323  
hostErrMemReadOutOfRange 323  
hostErrMemWriteOutOfRange 323  
HostErrNo() 330  
hostErrNone 323  
hostErrNoSignalWaiters 324  
hostErrNotADirectory 324  
hostErrOpNotAvailable 325  
hostErrorClass 323  
hostErrOutOfMemory 323  
hostErrPermissions 324  
hostErrProfilingNotReady 325  
hostErrReadOnlyFS 325  
hostErrRPCall 324  
hostErrSessionNotPaused 324  
hostErrSessionNotRunning 324  
hostErrSessionRunning 324  
hostErrTimeout 323  
hostErrTooManyFiles 324  
HostErrType 312  
hostErrUnknownError 325  
hostErrUnknownGestaltSelector 323  
HostExgLibAccept() 331  
HostExgLibClose() 331  
HostExgLibConnect() 331  
HostExgLibControl() 331  
HostExgLibDisconnect() 331  
HostExgLibGet() 332  
HostExgLibHandleEvent() 332  
HostExgLibOpen() 332  
HostExgLibPut() 332  
HostExgLibReceive() 333  
HostExgLibRequest() 333  
HostExgLibSend() 333  
HostExgLibSleep() 333  
HostExgLibWake() 334  
HostExitedApp() 334  
HostExportFile() 334  
HostFClose() 335  
HostFEOF() 335  
HostFError() 336  
HostFFlush() 336  
HostFGetC() 336  
HostFGetPos() 337  
HostFGetS() 337  
HostFILE 312  
hostFileAttrHidden 327  
hostFileAttrReadOnly 327  
hostFileAttrSystem 327  
HostFILEType 312  
HostFOpen() 338  
HostFPrintf() 339  
HostFPutC() 339  
HostFPutS() 340  
HostFRead() 340

---

HostFree() 341  
HostFReopen() 341  
HostFScanF() 342  
HostFSeek() 342  
HostFSetPos() 343  
HostFTell() 343  
HostFWrite() 344  
HostGestalt() 344  
HostGetChar() 345  
HostGetDirectory() 345  
HostGetEnv() 345  
HostGetFile() 346  
HostGetFileAttr() 346  
HostGetFirstApp() 347  
HostGetHostID() 347  
HostGetHostPlatform() 347  
HostGetHostVersion() 348  
HostGetPreference() 348  
HostGMTime() 349  
HostHostControl68K() 350  
HostID 313  
hostIDPalmOS 325  
hostIDPalmOSEmulator 325  
hostIDPalmOSSimulator 326  
HostIDType 313  
HostImportFile() 350  
HostIsCallingTrap() 350  
HostIsSelectorImplemented() 351  
HostLocalTime() 351  
HostLogEvent() 352  
HostLogFile() 352  
HostMalloc() 352  
HostMkDir() 353  
HostMkTime() 353  
HostOpenDir() 353  
HostPlatform 313  
hostPlatformMacintosh 326  
hostPlatformPalmOS 326  
HostPlatformType 313  
hostPlatformUnix 326  
hostPlatformWindows 326  
HostPrintf() 354  
HostProfileCleanup() 354  
HostProfileDetailFn() 355  
HostProfileDump() 355  
HostProfileInit() 356  
HostProfileStart() 357  
HostProfileStop() 357  
HostPutFile() 358  
HostReadDir() 358  
HostRealloc() 359  
HostRemove() 359  
HostRename() 360  
HostRmDir() 360  
hostSelectorAscTime 318  
hostSelectorBase 318  
hostSelectorClock 318  
hostSelectorCloseDir 318  
hostSelectorCTime 318  
hostSelectorEnteringApp 319  
hostSelectorErrNo 319  
hostSelectorExgLibAccept 319  
hostSelectorExgLibClose 319  
hostSelectorExgLibConnect 319  
hostSelectorExgLibControl 319  
hostSelectorExgLibDisconnect 319  
hostSelectorExgLibGet 319  
hostSelectorExgLibHandleEvent 319  
hostSelectorExgLibOpen 319  
hostSelectorExgLibPut 319  
hostSelectorExgLibReceive 319  
hostSelectorExgLibRequest 319  
hostSelectorExgLibSend 319  
hostSelectorExgLibSleep 319  
hostSelectorExgLibWake 319  
hostSelectorExitedApp 319  
hostSelectorExportFile 319  
hostSelectorFClose 319  
hostSelectorFEOF 319  
hostSelectorFError 319  
hostSelectorFFlush 319  
hostSelectorFGetC 319  
hostSelectorFGetPos 319  
hostSelectorFGetS 319  
hostSelectorFOpen 319  
hostSelectorFFprintf 319

---

hostSelectorFPutC	319	hostSelectorReadDir	321
hostSelectorFPutS	320	hostSelectorRealloc	321
hostSelectorFRead	320	hostSelectorRemove	321
hostSelectorFree	320	hostSelectorRename	321
hostSelectorFReopen	320	hostSelectorRmdir	321
hostSelectorFScanF	320	hostSelectorSessionClose	321
hostSelectorFSeek	320	hostSelectorSessionCreate	321
hostSelectorFSetPos	320	hostSelectorSessionOpen	321
hostSelectorFTell	320	hostSelectorSessionQuit	321
hostSelectorFWrite	320	hostSelectorSetErrorLevel	321
hostSelectorGestalt	320	hostSelectorSetFileAttr	321
hostSelectorGet68KDebuggerPort	320	hostSelectorSetLogFileSize	321
hostSelectorGetChar	320	hostSelectorSetPreference	321
hostSelectorGetDirectory	320	hostSelectorSignalResume	321
hostSelectorGetEnv	320	hostSelectorSignalSend	322
hostSelectorGetFile	320	hostSelectorSignalWait	322
hostSelectorGetFileAttr	320	hostSelectorSlotHasCard	322
hostSelectorGetFirstApp	320	hostSelectorSlotMax	322
hostSelectorGetHostID	320	hostSelectorSlotRoot	322
hostSelectorGetHostPlatform	320	hostSelectorStat	322
hostSelectorGetHostVersion	320	hostSelectorStrFTime	322
hostSelectorGetPreference	320	hostSelectorTime	322
hostSelectorGMTIME	320	hostSelectorTmpFile	322
hostSelectorHostControl68K	320	hostSelectorTmpNam	322
hostSelectorImportFile	320	hostSelectorTraceClose	322
hostSelectorIsCallingTrap	320	hostSelectorTraceInit	322
hostSelectorIsSelectorImplemented	320	hostSelectorTraceOutputB	322
hostSelectorLastTrapNumber	320	hostSelectorTraceOutputT	322
hostSelectorLocalTime	320	hostSelectorTraceOutputTL	322
hostSelectorLogEvent	321	hostSelectorTraceOutputVT	322
hostSelectorLogFile	321	hostSelectorTraceOutputVTL	322
hostSelectorMalloc	321	hostSelectorTruncate	322
hostSelectorMkdir	321	hostSelectorUTime	322
hostSelectorMkTime	321	hostSelectorVFPrintf	322
hostSelectorOpenDir	321	hostSelectorVFScanF	322
hostSelectorPrintf	321	hostSelectorVPrintf	322
hostSelectorProfileCleanup	321	HostSessionClose()	361
hostSelectorProfileDetailFn	321	HostSessionCreate()	361
hostSelectorProfileDump	321	HostSessionOpen()	362
hostSelectorProfileInit	321	HostSessionQuit()	362
hostSelectorProfileStart	321	HostSetErrorLevel()	363
hostSelectorProfileStop	321	HostSetFileAttr()	363
hostSelectorPutFile	321	HostSetLogFileSize()	364

---

HostSetPreference() 364  
HostSignal 314  
hostSignalIdle 326  
hostSignalQuit 326  
hostSignalReserved 326  
HostSignalResume() 365  
HostSignalSend() 365  
HostSignalType 314  
hostSignalUser 326  
HostSignalWait() 366  
HostSizeType 314  
HostSlotHasCard() 367  
HostSlotMax() 367  
HostSlotRoot() 368  
HostStat() 368  
HostStatType 314  
HostStrFTime() 369  
HostTime() 369  
HostTimeType 316  
HostTmpFile() 370  
HostTmpNam() 370  
HostTmType 317  
HostTraceClose() 370  
HostTraceInit() 371  
HostTraceOutputB() 371  
HostTraceOutputT() 372  
HostTraceOutputTL() 373  
HostTraceOutputVT() 374  
HostTraceOutputVTL() 375  
HostTruncate() 375  
HostUTime() 376  
HostUTimeType 318  
HostVFPrintF() 376  
HostVFScanF() 377  
HostVPrintF() 377  
HotSync 19, 21, 55  
hoursInMinutes 193  
hoursInSeconds 193  
hoursPerDay 193  
hsExgErrorClass 171  
htalErrorClass 171  
HttpErrorClass 171  
hwrErrorClass 171

## I

inetErrorClass 171  
insertion sort 500  
intlErrorClass 172  
iosErrorClass 172  
IrCommErrorClass 172  
IrErrorClass 172

## J

january 192  
july 192  
june 192

## K

kalErrorClass 172  
kAttnCommandCustomEffect 17, 143  
kAttnCommandDrawDetail 14, 143  
kAttnCommandDrawList 15, 143  
kAttnCommandGoThere 17, 143  
kAttnCommandGotIt 18, 143  
kAttnCommandIterate 19, 144  
kAttnCommandPlaySound 17, 144  
kAttnCommandSnooze 19, 144  
kAttnFlagsAllBits 141  
kAttnFlagsAlwaysCustomEffect 142  
kAttnFlagsAlwaysLED 142  
kAttnFlagsAlwaysSound 142  
kAttnFlagsAlwaysVibrate 142  
kAttnFlagsCapabilitiesMask 140  
kAttnFlagsCustomEffectBit 141  
kAttnFlagsEverything 142  
kAttnFlagsHasCustomEffect 140  
kAttnFlagsHasLED 140  
kAttnFlagsHasSound 140  
kAttnFlagsHasVibrate 140  
kAttnFlagsLEDBit 141  
kAttnFlagsNoCustomEffect 142  
kAttnFlagsNoLED 142  
kAttnFlagsNoSound 142  
kAttnFlagsNothing 142  
kAttnFlagsNoVibrate 142  
kAttnFlagsSoundBit 141  
kAttnFlagsUserSettingsMask 140

---

kAttnFlagsUserWantsCustomEffect 140, 415  
kAttnFlagsUserWantsLED 141, 415  
kAttnFlagsUserWantsSound 141, 415  
kAttnFlagsUserWantsVibrate 141, 415  
kAttnFlagsUseUserSettings 141  
kAttnFlagsVibrateBit 141  
kAttnFtrCapabilities 139  
kAttnFtrCreator 139  
kAttnIndicatorHeight 138  
kAttnIndicatorLeft 138  
kAttnIndicatorTop 138  
kAttnIndicatorWidth 139  
kAttnLevelInsistent 144  
kAttnLevelSubtle 144  
kAttnListMaxIconWidth 139  
kAttnListTextOffset 139  
kCancelPerfRequest 402  
kCPUClockInfoVersion\_0 402  
kCreatePerfRequest 402  
kCurrentCPUClockInfoVersion 402  
kDALError 177  
kDALTimeout 177  
kErrorLevelFatal 327  
kErrorLevelNone 327  
kErrorLevelWarning 327  
kGetCPUClockInfo 403  
kGetCPUClockRateArray 403  
kPalmOSEmulatorFeatureCreator 327  
kPalmOSEmulatorFeatureNumber 328  
kPerfClockValueDelta 403  
kPerfClockValueMax 403  
kPerfRequestAny 403  
kSetDefaultCPUClockRate 403  
kSyncAppDescriptionMaxLen 422  
kSyncMaxActiveSessions 422  
kSyncPolicyNonUiAuthentication 422  
kSyncPolicyRestrictAppRegistration 422  
kSyncProductNameMaxLen 422

## L

lastYear 194  
lmAnyLanguage 215  
lmErrorClass 172

LmLocaleType 414  
locale 414  
longDateStrLength 191, 207  
lz77ErrorClass 172

## M

march 192  
maxDays 194  
maximizing battery life 102  
maxSeconds 194  
may 192  
mdmErrorClass 172  
MeasurementSystemType 406  
mediaErrAlreadyConnected 177  
mediaErrAlreadyVisited 177  
mediaErrFormatMismatch 177  
mediaErrNoBufferSource 177  
mediaErrNotConnected 178  
mediaErrorClass 172  
mediaErrStreamExhausted 178  
memErrInvalidParam 283, 284  
memErrNotEnoughSpace 283, 284, 285  
memErrorClass 172  
Memory Stick 54  
menuErrorClass 172  
minutesInSeconds 193  
modes 101  
    efficient use 103  
monday 192  
Months 191  
monthsInYear 193  
MultiMedia (MMC) 54  
multiple preferences 419

## N

netErrorClass 172  
noPreferenceFound 417  
notifications  
    expansion 61, 62  
    registering for expansion 63  
noTime 194  
november 192  
numberOfYears 194

---

## O

october 192  
oemErrorClass 172  
omErrorClass 172

## P

P\_ABSOLUTE\_TIMEOUT 457  
P\_POLL 457  
P\_RELATIVE\_TIMEOUT 457  
P\_WAIT\_FOREVER 457  
padErrorClass 172  
patches, loading during reset 83  
patchFlagHead 399  
patchFlagReservedMask 399  
patchFlagTail 399  
pdiErrorClass 172  
penErrorClass 172  
perfErrBufferTooSmall 403  
perfErrDeniedPowerLow 403  
perfErrInvalidParams 403  
perfErrLimitReached 403  
perfErrNone 403  
perfErrNotImplemented 403  
perfErrorClass 173  
PerfGenCPUClockInfoType 401  
PerfRefNumType 402  
PerfResultType 402  
pinsErrorClass 173  
plug and play block device driver 55  
posixErrorClass 173  
power modes 101  
pppErrorClass 173  
Predefined 456  
PrefActivePanelParamsPtr 405  
PrefActivePanelParamsType 405  
prefAlarmSoundLevelV20 408  
prefAlarmSoundVolume 411  
prefAllowEasterEggs 409  
prefAnimationLevel 411  
prefAntennaCharAppCreator 412  
prefAppLaunchCmdSetActivePanel 416  
prefAttentionFlags 415  
prefAutoLockTime 413

prefAutoLockTimeFlag 413  
prefAutoLockType 413  
prefAutoOffDuration 408  
prefAutoOffDurationSecs 412  
prefBeamReceive 411  
prefCalcCharAppCreator 410  
prefCalibrateDigitizerAtReset 411  
prefColorThemeID 415  
prefCountry68K 407  
prefDateFormat 407  
prefDaylightSavingAdjustment 413  
prefDaylightSavings 409  
prefDefaultAppCreator 415  
prefDefFepPlugInCreator 415  
prefDefSerialPlugIn 411  
prefDeviceLocked 408  
preference file names 349  
preferences  
    auto-off 103  
    multiple application preferences 419  
prefFormatsLocale68K 414  
prefGameSoundLevelV20 408  
prefGameSoundVolume 411  
PrefGetAppPreferences() 416  
PrefGetPreference 37  
PrefGetPreference() 418  
prefHard1CharAppCreator 410  
prefHard2CharAppCreator 410  
prefHard3CharAppCreator 410  
prefHard4CharAppCreator 410  
prefHardCradle2CharAppCreator 411  
prefHardCradleCharAppCreator 410  
prefHidePrivateRecordsV33 408  
prefLanguage68K 414  
prefLauncherAppCreator 410  
prefLocalSyncRequiresPassword 409  
prefLongDateFormat 407  
prefMeasurementSystem 412  
prefMinutesWestOfGMT 409  
prefNumberFormat 408  
prefRemoteSyncRequiresPassword 409  
prefRonamaticChar 409  
PrefSetAppPreferences() 418  
PrefSetPreference() 420

---

prefShowPrivateRecords 412  
prefStayLitWhenPluggedIn 412  
prefStayOnWhenPluggedIn 412  
prefSysBatteryKind 409  
prefSysPrefFlags 410  
prefSysSoundLevelV20 408  
prefSysSoundVolume 411  
prefSystemKeyboardID 411  
prefTimeFormat 408  
prefTimeZone 412  
prefTimeZoneCountry 414  
prefVersion 407  
prefWeekStartDay 407  
primary storage 54  
privateRecordViewEnum 412  
processes 88  
    dynamic creation of 88  
pwrErrBacklight 427  
pwrErrBeam 427  
pwrErrGeneric 427  
pwrErrNone 428  
pwrErrorClass 173  
pwrErrRadio 428

## R

radioErrorClass 173  
ralErrorClass 173  
RAM  
    expansion 54  
real-time clock 107  
regexErrCorruptedOpcode 178  
regexErrCorruptedPointers 178  
regexErrCorruptedProgram 178  
regexErrInternalError 178  
regexErrInvalidBracketRange 178  
regexErrJunkOnEnd 178  
regexErrMemoryCorruption 178  
regexErrNestedStarQuestionPlus 178  
regexErrorClass 173  
regexErrQuestionPlusStarFollowsNothing 178  
regexErrStarPlusOneOperandEmpty 179  
regexErrTooBig 179  
regexErrTooManyParenthesis 179

regexErrTrailingBackslash 179  
regexErrUnmatchedBracket 179  
regexErrUnmatchedParenthesis 179  
reminder dialog boxes 24  
reset 83  
    digitizer screen 84  
    hard reset 83  
    loading patches 83  
    soft reset 83  
rfutErrorClass 173  
ROM  
    expansion 55  
running mode 102

## S

saturday 192  
SearchFuncPtr 486  
secErrorClass 173  
secondary storage 54  
secondsInSeconds 193  
secSvcErrorClass 173  
Secure Digital (SD) 54  
security  
    and expansion 67  
september 192  
serErrorClass 173  
signErrorClass 173  
sleep mode 101  
    and current time 107  
slkErrorClass 173  
slOff 406  
slOn 406  
slot reference number 64, 69  
slotDrvBootablePartition 263  
slotDrvNonBootablePartition 264  
slotDrvPartitionTypeFAT12 264  
slotDrvPartitionTypeFAT16Over32MB 264  
slotDrvPartitionTypeFAT16Under32MB 264  
slotDrvPartitionTypeFAT32 264  
slots 54  
    checking for a card 69  
    enumerating 68  
    referring to 56  
SMS application

---

and Attention Manager 13  
smsErrorClass 173  
sndErrorClass 173  
sndMaxAmp 411  
snooze timer (Attention Manager) 7  
snoozing items in Attention Manager 19  
soft reset 83  
    dynamic heap 83  
sorting array elements 500  
SoundLevelTypeV20 406  
sslErrorClass 173  
start.prc 66  
statErrorClass 174  
storage  
    primary 54  
    secondary 54  
storage heaps, erasing 84  
sunday 192  
svcErrorClass 174  
SyncAddSynchronizer() 423  
syncMgrErrAccessDenied 421  
syncMgrErrMaxSessionsActive 421  
syncMgrErrMemAllocFailure 421  
syncMgrErrOperationNotSupported 421  
syncMgrErrorClass 174  
syncMgrErrSystemErr 422  
syncMgrErrUserRefusedSyncApp 422  
SyncSessionGetAccess() 424  
SyncSessionReleaseAccess() 424  
SysAlarmTriggeredParamType 122  
SysAppLaunch 17  
sysAppLaunchCmdAlarmTriggered 20, 25, 27,  
    125, 127  
sysAppLaunchCmdAttention 145  
sysAppLaunchCmdCardLaunch 60, 66  
sysAppLaunchCmdDisplayAlarm 20, 25, 27, 125,  
    127  
sysAppLaunchCmdGoto 17  
sysAppLaunchCmdNormalLaunch 60, 66  
SysAppLaunchCmdReset 83  
sysAppLaunchCmdSystemReset 83  
sysAppLaunchStartFlagNoUISwitch 60, 66  
SysAreWeRunningUI() 488  
SysAreWeUIThread() 488  
SysAtomicAdd32() 458  
SysAtomicAnd32() 458  
SysAtomicCompareAndSwap32() 459  
SysAtomicOr32() 460  
SysBatteryInfo 103  
SysBatteryInfo() 436  
SysBatteryInfoV40() 489  
SysBatteryKind 165, 409, 486  
sysBatteryKindAlkaline 166  
sysBatteryKindAntiMatter 166  
sysBatteryKindFuelCell 166  
sysBatteryKindLast 166  
sysBatteryKindLiIon 166  
sysBatteryKindLiIon1400 166  
sysBatteryKindNiCad 166  
sysBatteryKindNiMH 166  
sysBatteryKindPlutonium237 166  
sysBatteryKindRechAlk 166  
SysBatteryKindTag 166  
SysBatteryState 165  
sysBatteryStateCritBattery 167  
sysBatteryStateLowBattery 167  
sysBatteryStateNormal 167  
sysBatteryStateShutdown 167  
SysBatteryStateTag 167  
SysBinarySearch() 490  
SysConditionVariableBroadcast() 460  
SysConditionVariableClose() 461  
sysConditionVariableInitializer 455  
SysConditionVariableOpen() 461  
SysConditionVariableType 451  
SysConditionVariableWait() 462  
SysCopyStringResource() 491  
SysCopyStringResourceV50() 492  
SysCreateDataBaseList() 492  
SysCreateDataBaseListV50() 494  
SysCreatePanelList() 495  
SysCriticalSectionEnter() 463  
SysCriticalSectionExit() 463  
sysCriticalSectionInitializer 455  
SysCriticalSectionType 452  
SysCurrentThread() 464  
SysDBListItemType 486, 493, 495

---

SysDisplayAlarmParamType 123  
sysDoNotVerifySignature 382  
sysEntryNumMain 382, 435  
sysErrBufTooSmall 179  
sysErrCPUArchitecture 179  
sysErrDynamicLinkerError 179  
sysErrInternalError 179  
sysErrInvalidSignature 180  
sysErrLibNotFound 180  
sysErrModuleFound68KCode 180  
sysErrModuleIncompatible 180  
sysErrModuleInvalid 180  
sysErrModuleNotFound 180  
sysErrModuleRelocationError 180  
sysErrNoFreeLibSlots 180  
sysErrNoFreeRAM 180  
sysErrNoFreeResource 180  
sysErrNoGlobalStructure 180  
sysErrNotAllowed 180  
sysErrNotInitialized 181  
sysErrorClass 174  
sysErrOSVersion 181  
sysErrOutOfOwnerIDs 181  
sysErrParamErr 181, 244  
sysErrPrefNotFound 181  
sysErrRAMModuleNotAllowed 181  
sysErrRomIncompatible 181  
SysErrString() 496  
SysErrStringTextOnly() 497  
sysErrTimeout 181  
sysErrWeakRefGone 181  
SysFatalAlert() 278  
SysFatalAlertInit() 278  
sysFileDescStdIn 435  
SysFormPointerArrayToStrings() 498  
sysFtrCreator 31, 435  
sysFtrDefaultBoldFont 428  
sysFtrDefaultFont 428  
sysFtrNumAccessorTrapPresent 428  
sysFtrNumBacklight 428  
sysFtrNumCharEncodingFlags68K 428  
sysFtrNumCountry68K 428  
sysFtrNumDefaultCompression 428  
sysFtrNumDisableSortDuringSyncThreshold 428  
sysFtrNumDisplayDepth 429  
sysFtrNumDisplayUpdateMode 429  
sysFtrNumDmAutoBackup 429  
sysFtrNumEnableSortAfterSyncThreshold 429  
sysFtrNumEncoding68K 430  
sysFtrNumEncryption 430  
sysFtrNumEncryptionMaskDES 435  
sysFtrNumFastBoot 430  
sysFtrNumFiveWayNavVersion 430  
sysFtrNumHwrMiscFlags 430  
sysFtrNumHwrMiscFlagsExt 430  
sysFtrNumInputAreaFlags 430  
sysFtrNumLanguage68K 430  
sysFtrNumNotifyMgrVersion 430  
sysFtrNumOEMCompanyID 430  
sysFtrNumOEMDeviceID 431  
sysFtrNumOEMHALID 431  
sysFtrNumOEMROMVersion 431  
sysFtrNumProcessor328 432  
sysFtrNumProcessor68KIfZero 433  
sysFtrNumProcessorARM710A 432  
sysFtrNumProcessorARM720T 432  
sysFtrNumProcessorARM7TDMI 432  
sysFtrNumProcessorARM920T 432  
sysFtrNumProcessorARM922T 432  
sysFtrNumProcessorARM925 432  
sysFtrNumProcessorARMIfNotZero 433  
sysFtrNumProcessorEZ 432  
sysFtrNumProcessorID 431  
sysFtrNumProcessorIs68K() 437  
sysFtrNumProcessorIsARM() 438  
sysFtrNumProcessorMask 433  
sysFtrNumProcessorStrongARM 432  
sysFtrNumProcessorSuperVZ 432  
sysFtrNumProcessorVZ 432  
sysFtrNumProcessorex86 433  
sysFtrNumProcessorXscale 433  
sysFtrNumProductID 431  
sysFtrNumResetType 431  
sysFtrNumROMBuildType 431  
sysFtrNumROMVersion 32, 431  
sysFtrNumSkipCalibration 431

---

sysFtrNumTextMgrFlags 431  
 sysFtrNumUIHardwareFlags 431  
 sysFtrNumVendor 431  
 sysFtrNumWinVersion 431  
 SysGetEntryAddresses() 384  
 SysGetModuleDatabase() 385  
 SysGetModuleGlobals() 386  
 SysGetModuleInfo() 387  
 SysGetModuleInfoByDatabaseID() 388  
 SysGetOSVersionString() 498  
 SysGetRefNum() 389  
 SysGetROMToken() 438  
 SysGetROMTokenV40() 499  
 sysGetROMVerBuild() 439  
 sysGetROMVerFix() 439  
 sysGetROMVerMajor() 440  
 sysGetROMVerMinor() 440  
 sysGetROMVerStage() 441  
 SysGetRunTime() 464  
 SysGremlins() 378  
 SysHandleEvent() 441  
 SysInsertionSort() 499  
 SysLCDBrightness() 442  
 SysLCDContrast() 442  
 SysLoadModule() 389  
 SysLoadModuleByDatabaseID() 392  
 SysMainEntryPtrType() 395  
 sysMakeROMVersion 32  
 sysMakeROMVersion() 443  
 SysModuleInfoType 380  
 sysNotifyCardInsertedEvent 62  
 sysNotifyCardRemovedEvent 62  
 sysNotifyErrBroadcastBusy 181  
 sysNotifyErrBroadcastCancelled 181  
 sysNotifyErrDuplicateEntry 182  
 sysNotifyErrEntryNotFound 182  
 sysNotifyErrNoServer 182  
 sysNotifyErrNoStackSpace 182  
 sysNotifyErrQueueEmpty 182  
 sysNotifyErrQueueFull 182  
 sysNotifyTimeChangeEvent 507  
 sysNotifyVolumeMountedEvent 62, 66  
 sysNotifyVolumeUnmountedEvent 62  
 sysOEMCompanyIDHandspring 434  
 sysOEMCompanyIDPalmDevices 434  
 sysOEMCompanyIDPalmPlatform 434  
 sysOEMCompanyIDQualcomm 434  
 sysOEMCompanyIDSymbol 435  
 sysOEMCompanyIDTRG 435  
 sysOEMCompanyIDUnspecified 435  
 sysOEMDeviceIDUnspecified 435  
 sysOEMHALIDUnspecified 435  
 SysPatchEntryNumType 397  
 SysPatchInfoType 381  
 sysPatchLaunchCmdClearInfo 383  
 sysPatchLaunchCmdSetInfo 383  
 SysPatchTargetHeaderType 398  
 SysQSort() 501  
 SysRandom() 502  
 sysRandomMax 487  
 SysRegisterPatch() 393  
 SysRequestSleep() 443  
 SysReset 84  
 sysROMStageAlpha 433  
 sysROMStageBeta 433  
 sysROMStageDevelopment 433  
 sysROMStageRelease 434  
 sysROMTokenSnum 434  
 SysSemaphoreCreate() 464  
 SysSemaphoreCreateEZ() 465  
 SysSemaphoreDestroy() 466  
 sysSemaphoreMaxCount 456  
 SysSemaphoreSignal() 466  
 SysSemaphoreSignalCount() 467  
 SysSemaphoreWait() 468  
 SysSemaphoreWaitCount() 469  
 SysSetAutoOffTime 103  
 SysSetAutoOffTime() 444  
 SysSleep() 444  
 SysStringByIndex() 502  
 SysStringByIndexV50() 503  
 SysTaskDelay 103, 108  
 SysTaskDelay() 444  
 system tick interrupts 107  
 system ticks  
     on Palm OS device 108

---

system version feature 31  
SystemMgr.h 32  
SystemPreferencesChoice 37, 39, 406  
SystemPreferencesType 407  
SysThreadChangePriority() 470  
SysThreadCreate() 471  
SysThreadCreateEZ() 472  
SysThreadDelay() 473  
SysThreadEnterFunc() 483  
SysThreadExit() 474  
SysThreadExitCallbackFunc() 483  
SysThreadExitCallbackID 452  
SysThreadGroupCreate() 475  
SysThreadGroupDestroy() 475  
SysThreadGroupHandle 452  
SysThreadGroupTag 453  
SysThreadGroupType 453  
SysThreadGroupWait() 476  
SysThreadInstallExitCallback() 476  
sysThreadNoGroup 455  
sysThreadPriorityBestApp 454  
sysThreadPriorityBestSystem 454  
sysThreadPriorityBestUser 454  
sysThreadPriorityDisplay 454  
sysThreadPriorityHigh 454  
sysThreadPriorityLow 454  
sysThreadPriorityLowered 454  
sysThreadPriorityNormal 454  
sysThreadPriorityRaised 454  
sysThreadPriorityRealTime 454  
sysThreadPriorityTransaction 454  
sysThreadPriorityUrgentDisplay 455  
SysThreadRemoveExitCallback() 477  
SysThreadResume() 478  
sysThreadStackBasic 455  
sysThreadStackUI 456  
SysThreadStart() 479  
SysThreadSuspend() 479  
SysTicksPerSecond 108  
SysTicksPerSecond() 445  
SysTimeInCentiSecs() 445  
SysTimeInMicroSecs() 446  
SysTimeInMilliSecs() 446

SysTimeInMins() 446  
SysTimeInSecs() 447  
SysTimeToMicroSecs() 447  
SysTimeToMilliSecs() 448  
SysTimeToSecs() 448  
SysTSDAllocate() 481  
sysTSDAnonymous 456  
SysTSDDestructorFunc() 484  
SysTSDFree() 482  
SysTSDGet() 482  
SysTSDSet() 482  
SysTSDSlotID 453  
SysTurnDeviceOn() 448  
SysUIBusy() 449  
SysUnloadModule() 394  
SysUnregisterPatch() 394

## T

telErrorClass 174  
Template 196  
tfColon 200  
tfColon24h 201  
tfColonAMPM 200  
tfComma24h 201  
tfDot 201  
tfDot24h 201  
tfDotAMPM 201  
tfHours24h 201  
tfHoursAMPM 201  
thursday 192  
TimAdjust() 211  
TimDateTimeToSeconds 26, 108  
TimDateTimeToSeconds() 211  
time manager 107  
TimeFormatTag 200  
TimeFormatType 190  
TimeGetFormatSeparator() 212  
TimeGetFormatSuffix() 212  
TimeIs24HourFormat() 213  
timeoutFlags\_t 457  
TimePtr 190  
timer 107  
timErrBadParam 505

---

timErrMemory 505  
timErrNoLunarCalendarSupport 505  
timErrorClass 174  
timeStringLength 191  
TimeToAscii() 213  
TimeToInt() 214  
TimeType 190  
timeZoneStringLength 191, 214, 215  
TimeZoneToAscii() 214  
TimeZoneToAsciiV50() 215  
TimGetSeconds 107  
TimGetSeconds() 506  
TimGetTicks 108  
TimGetTicks() 506  
timing 108  
TimInit() 506  
TimSecondsToDateTime 107  
TimSecondsToDateTime() 216  
TimSetSeconds 107  
TimSetSeconds() 507  
TimTimeZoneToUTC() 216  
TimUTCToTimeZone() 217  
tlsErrorClass 174  
TM() 509  
TmOutputB() 510  
TmOutputT() 511  
TmOutputTL() 512  
TmOutputVT() 513  
TmOutputVTL() 514  
TraceDefine() 514  
TraceOutput() 515  
try-and-catch mechanism 116  
    example 117  
tsmErrorClass 174  
tuesday 192  
txtErrorClass 174

## U

udaErrorClass 174  
unitsEnglish 406  
unitsMetric 406  
universal connector 55  
user name  
    obtaining 234

## V

VFAT 57  
VFS Manager 54, 58  
    and file streaming 58  
    and the Data Manager 58  
    functions 58  
    overriding notification handlers 64  
    registered notifications 62  
    starting apps automatically 66  
    verifying presence of 67  
vfsErrorClass 174  
VFSVolumeEnumerate 68  
Virtual File System. See VFS Manager  
voltage warning threshold 436, 489  
volumes  
    and slots 57  
    enumerating 67  
    mounted 67

## W

webErrorClass 174  
wednesday 192  
winErrorClass 174

## X

xSyncErrorClass 174

