

Bbug User's Manual

A Bootstrap Mode Operation Debugger

Ver 1.1

11/27/98

1.0 Introduction

BBug is developed to provide users a simple way to debug any MC68EZ328 or MC68VZ328 based target systems and to download boot code or application code to RAM or Flash memory. MC68EZ328 is the first MPU equipped with this new bootstrap mode feature.

BBug is a debugger program, which runs under DOS environment of PC and communicates to the target system through their serial ports. The target system is set to Bootstrap mode operation. For more information on putting system in bootstrap mode, please refer to MC68EZ328 or MC68VZ328 user's manual.

BBug takes no system memory of target system and requires no external bootcode, so BBug can still run on the system even the RAM and ROM subsystems are not available. It is extremely useful for application systems hardware debugging.

Up to the point of releasing this document, the following BBug versions are available for users:

BBUG.EXE – for MC68EZ328 systems running on UART port

BBUGV1.EXE – for MC68VZ328 systems running on UART1 port

BBUGV2.EXE – for MC68VZ328 systems running on UART2 port

2.0 What BBug can do

3.0 BBUG.EXE

The program “BBUG.exe” is a simulation of Windows terminal. It allows user control on transmitting and receiving data from the communication port connected to target system. When the target system is reset to bootstrap mode operation, user may begin to run BBUG.

3.1 Requesting Connection Port

At the start of the program, it requests user to select the PC communication port, i.e. whether the UART port of the target system is connected to COM1 or COM2 of PC. If the user given port address is wrong, target board has not been reset or not properly connected to PC, the program will not be able to function properly and error messages will prompt.

After the communication port is chosen, the function menu will appear on the screen. BBUG detects the communication speed of target system at 9600, 19200 and 115200 baud. When the link is set up, the prompt “Bootstrap:>” is shown without error messages and user may start to use the program commands.

3.2 Target system default setting

When the target system enters bootstrap mode, the communication preferences of target system are set to the following default communication protocols:

- baud rate at 9600 bps
- parity is NONE
- data bit is 8 bit
- stop bit is 1 bit

When BBUG is invoked, BBUG initialize the target system and set PC host to 19200 baud. User can change baud rate to 115.2K bps using BBUG command. If the target system was running in 115.2K bps before run BBUG, the host is set to 115.2K automatically.

3.3 Commands of BBug

The program consists of eight Commands : initialize system, download file, memory display, memory modify, memory execution, change baud rate, help, and exit. Each command can be executed by typing in the correct command strings.

If the executing function involves file manipulation, the program will prompt “Cannot Open File “ when the specified file cannot be found with the given file path.

If user has input a command string, which is not recognizable to the program, an error message of “Syntax Error” will be prompted.

In cases of wrong communication between the PC and the target system, the data sent across may not be correct. “ Receive Time Out Error” messages will alert user about the sending errors.

3.31 Initialize System

command: IN (filename)
or INIT (filename)
e.g. init ads_init.b

The target system must be initialized with its own configuration before other functions could work properly. This initializing function sends the B-record to the target system. When the file name is not included in the command, BBug will search for a file called “init.b” in the current directory, and download it if it is present. An init file is a file of b-records, which initialize the MPU internal registers. User can also use MM command to initialize the internal register.

3.32 Download File

command: LO (file name) (file type)
e.g. LO INIT.B B
e.g. LO A:\EXAMPLE.S S

This function is for downloading s-record file or b-record file to the target system. The

If the sending file is in b-record format, the file will be directly sent through the UART port of target system, character by character. If the sending file is in s-record format, the program first converts the file from s-record format to b-record format and save it as a b-record file. The output b-record file is then downloaded to the target system. While the file is being sent, the b-records are displayed on the screen.

To just convert an s-record to a b-record file, user can either use BBug or use another utility program STOB.EXE.

3.33 Memory Display

command: MD (address) (byte count)
e.g. MD 10000 30

This function is for displaying the content in user specified memory. The (address) is the starting address of memory to be read, and (byte count) to be less than 100 hex, is the number of bytes to be read. Both the address and byte count are hexadecimal numbers. The memory contents are also displayed as hexadecimal.

3.34 Memory Modify

command: MM (address)
e.g. MM 10000

This function is for modifying the content in the system RAM or MPU internal registers. The (address) is the address location to be modified. When the above command string is input, the content of the associated location is displayed. User may then input the data to overwrite the current displayed content. The data is restricted to hexadecimal numbers from 00 to FF. While the content has been modified, the function will read the next location and request for another input. If user presses [ENTER] with no typing, the program will skip the current register, jump to the next one and wait for input. If user type ".", the memory modifying function will terminate and return to the main routine.

For inputs that are not [ENTER], ".", or not recognized as hexadecimal numbers from 00 to FF, an error message will prompt. The memory modifying function will terminate and return to the main routine again.

3.35 Program Execution

command: GO (address)
e.g. GO 10000

This function is for executing a program in the memory content. The (address) is the memory location where the program begins. Note that the program in the memory which is executing may not have the command for the program control to return to BBug. Therefore, the target system may have to be reset before any further functions are used. To return to BBug after running a program, the last instruction of the program will be “jmp \$FFFFFFF44”.

3.36 Change Baud Rate

command: CB

After target system reset, the target system is set to 9600 baud by default. However, BBug reinitialize target system to 19200 when a target system reset is detected. CB function is use to change the communication baud rate between the PC and the target system. When the command is typed, the current baud rate is displayed. User is then asked to select the new baud rate. The available baud rates are:

1. 19200
2. 115200

For example, if the desired baud rate is 115200 bps, the user should type “2”. Any other inputs (besides 1 and 2) will result no response from the program.

3.37 Help

command: HE
or HELP (command)

This is the help function which displays instructions on using the other functions the specific commands. With “HE” typed only, the function menu will be displayed on screen. If the (command) is typed as one of the eight function commands in this program, the

3.38 Exit

command: EX
or EXIT

This function is to exit the running program BBug, and return to the DOS environment.

3.4 Target System Reset

If the connection between PC and target system is disturbed during execution of BBug, user may restore the physical port connection by resetting target system. The communication is then restored.

4.0 Useful Hints

1. build an init.b file which initializes the target system. Using INIT command to download this file to target system each time the system is power up, so the system is initialized.
2. In using BBug to burn system Flash memory, please refer to the following procedures:
 - 1) Build s-record file for the program (e.g. CODE.S) to be burn in Flash at address ADD1.
 - 2) Write a Flash burning program and build s-record at RAM space not interference with ADD1. This program will be run and to move data from RAM ADD1 to Flash ADD3. Please refer to FLASH1.S attached below for more info.
 - 3) Initialize system using a init.b file.
 - 4) Download CODE.S
 - 5) Download FLASH.S
 - 6) Run FLASH.S using go command.
3. To display message when running a program using BBug, user can write the characters to EZ UART transmit register. BBug will receive the characters and store in a buffer and print on screen when it receives a break character (zero value).
4. To return to bootstrap mode after running a user program, the last instruction of the

4.0 Examples

4.1 Init file example

```
*****
* init.b -- Init ADS to default monitor config
* date: 04/20/98
*
*****

FFFFF1180130 emucs init
FFFFF000011C SCR init
FFFFFB0A0100 Disable WD
FFFFF42B0183 enable clko
FFFFF40B0100 enable chip select
FFFFFD0D0108 disable hardmap
FFFFFD0E0107 clear level 7 interrupt
FFFFF100020100 CSA 2M - 4M
FFFFF1100201A7
FFFFF102020000 CSB 0 - 256K
FFFFF112020091
FFFFFC00028F00 DRAM Config
FFFFFC02029667 DRAM Control
FFFFF106020200 CSD init -- RAS0 4M-6M, RAS1 6M-8M
FFFFF11602029D enable DRAM cs
FFFFF3000140 IVR
FFFFF30404007FFFF IMR
```

4.2 FLASH1.S -- an example program to burn user Flash memory in EZ328ADS

```
;;;;;;;;;;;;;
; Program: FLASH1.S
; 1. This program is used to copy data from DRAM(config to 0M-4M space)
; to user flash(config to 4M to 6M space)
; 2. For different memory map, CHANGE OFFSETX TO PROPER VALUES.
```

```

OFFSET3 equ    $50AAAA          ; FLASH bank 2 starting address
OFFSET4 equ    $505554          ; is $300000

        section code

START   movea.l #stack,a7        ;re-istall stack pointer in case bootstrap mode
        add.l   #$400,a7
        move.l  #SOURCE,a0
        move.l  #DEST,a1
        move.l  #SIZE,d0

        move.l  a0,a2            ;a2 & a3 are used to store the starting
address    move.l  a1,a3          ;for comparing.
        move.l  d0,d5            ;d5 is used to store the length for comparing.

        move.w  #$10,d4          ;$10 sections per acknowledge
nextsec   bsr    enable           ;Software Data Protection Enable Algorithm.
        bsr    copy              ;Copying.
        bsr    check             ;Checks for Flash write finish
        sub.l   #512,d0           ;If the whole sector has been copied,then
copies    cmp.l   #0,d0           ;the next sector (nextsec).  If all the
sectors   have
        ble     chk_err          ;been copied, then verify.
        sub.w   #1,d4
        bne.b   nextsec
        move.b   #'.',$fff907    ;display a "."
        move.b   #0,$fff907
        move.w   #$10,d4         ;reinstore d4
        bra     nextsec

chk_err   bsr    compare          ;Compares to see if the right sectors are
copied.   bra     finish

enable    move.w   $AAAA,OFFSET1  ; unlock bank 1
        move.w   $5555,OFFSET2
        move.w   $A0A0,OFFSET1
        move.w   $AAAA,OFFSET3   ; unlock bank 2
        move.w   $5555,OFFSET4
        move.w   $A0A0,OFFSET3
        rts

copy      clr.w    d1             ;d1 is used to count the number of words
copied.
nextwd    move.w   (a0)+,d6
        move.w   d6,(a1)+
        add.w    #1,d1           ;Count the number of words copied.
        cmp.w    #256,d1        ;Copy the next word (nextwd)
        blt     nextwd          ;until the whole sector has been coied.
        rts

check     move.l   #$ffffff,d1

```



```

        bra.w    error_exit ;error exit
con      add.l    #2,d1      ;Increments d1, a2, a3 to check the next word.
        cmp.l    d1,d5      ;Checks to see if all the words copied are
compared.
        bgt      lp_cmp     ;If yes, exits. Otherwise, continue to compare.
        rts

finish
        move.b   #'D',$fff907    ; display DONE
        bsr      wait
        move.b   #'O',$fff907
        bsr      wait
        move.b   #'N',$fff907
        bsr      wait
        move.b   #'E',$fff907
        bsr      wait
        move.b   #0,$fff907
        bra.b    exit

error_exit
        bsr      wait
        move.b   #'e',$fff907    ; display error
        bsr      wait
        move.b   #'r',$fff907
        bsr      wait
        move.b   #'r',$fff907
        bsr      wait
        move.b   #'o',$fff907
        bsr      wait
        move.b   #'r',$fff907
        bsr      wait
        move.b   #0,$fff907
        bra.b    exit

wait     btst.b   #7,$1ffff906
        beq.b    wait
        move.w   #$fff,d1
lp_wt    sub.w    #1,d1
        bne.b    lp_wt
        rts

exit     jmp      $ffffff44      ; return to bootloader
stack   ds.w     $400

end

```

5.0 Technical support

For more info on BBug and the most update to date version, please visit our web site:

<http://www.apspg.com/products.html>