EMBEDDEDPLANET

dBUG

Reference Manual
MPC5200

Developing Embedded Applications and Products Utilizing Freescale™ MPC5200 Integrated Processors

700M0125RA1

Copyright

Notice

Trademarks

Contents

List of Tables

Introduction

Chapter 1

dBUG is a traditional ROM monitor/debugger that offers a comfortable and intuitive command line interface that can be used to download and execute code. It contains all the primary features needed in a debugger to create a useful debugging environment.

Command Line Usage

The user interface to dBUG is the command line. A number of features have been implemented to achieve an easy and intuitive command line interface.

dBUG assumes that an 80x24 ASCII character dumb terminal is used to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit (8N1). The baud rate is 9600 bps by default (if not set differently by the user) — a speed commonly available from workstations, personal computers, and dedicated terminals.

The command line prompt is:

dBUG>

Any dBUG command may be entered from this prompt. dBUG does not allow command lines to exceed 80 characters. Wherever possible, dBUG displays data in 80 columns or less. dBUG echoes each character as it is typed, eliminating the need for any local echo on the terminal side.

The **BACKSPACE** and **DELETE** keys are recognized as rub-out keys for correcting typographical mistakes.

Command lines may be recalled using the **CTRL-U**, **CTRL-D**, and **CTRL-R** key sequences. **CTRL-U** and **CTRL-D** cycle up and down through previous command lines. **CTRL-R** recalls and executes the last command line.

In general, dBUG is not case-sensitive. Commands may be entered either in uppercase or lowercase, depending upon the user's equipment and preference. Only symbol names require that the exact case be used.

Most commands can be recognized by using an abbreviated name. For instance, entering **h** is the same as entering **help**. It is not necessary to type the entire command name.

The commands **DI**, **GO**, **MD**, **STEP**, and **TRACE** are used repeatedly when debugging. dBUG recognizes this and allows for repeated execution of these commands with minimal typing. After a command is entered, press **RETURN** or **ENTER** to invoke the command again. The command is executed as if no command line parameters were provided.

Document Conventions

This document uses standard text conventions to represent keys, display items, and user data inputs:

Display Item

Italic - Identifies an item that displays on the screen such as a menu option or message (e.g., *File* > *Open*).

User Data Input

Bold - Identifies any part of a command or user entry that is not optional or variable and must be entered exactly as shown.

Italic - Identifies any part of a command or user entry that is a variable parameter.

[ ] - Identifies any part of a command or user entry that is an optional parameter; text within the brackets follows the previously described conventions.

KEY - Identifies a specific key that is not alphabetic, numeric, or punctuation:

Press **ENTER**
Press **ESC V M** (press and release each key in sequence)
Press **CTRL-ALT-DEL (**press all keys in sequence simultaneously).

File Names

`Name` - Indicates a file or directory name. Example:

```
file.h
/bin
```

Commands

Chapter 2

This section lists the commands that are available with dBUG.

dBUG Command Summary

Table 2-1 summarizes the avialable dBUG commmands and their usage.

Table 2-1. dBUG Command Summary

Command	Description	Usage
ASM	Assemble	**ASM** [[*addr*] *stmt*]
BF	Block Fill	**BF** [*width*] *begin end data* [*inc*]
BM	Block Move	**BM** *begin end dest*
BS	Block Search	**BS** [*width*] *begin end data*
DC	Data Convert	**DC** *value*
DIS	Disassemble	**DI** [*addr*]
DL	Download Serial	**DL** [*offset*]
DN	Download Network	**DN** [**-c**] [**-e**] [**-i**] [**-s** [**-o** *offset*]] [*filename*]
FS	FLASH Status	**FS**
FE	FLASH Erase	**FE** *begin* [*end*]
FP	FLASH Program	**FP** *begin end source*
GO	Execute	**GO** [*addr*]
GT	Execute To	**GT** *addr*
HELP	Help	**HELP** [*command*]
LR	Loop Read	**LR** *addr*
LW	Loop Write	**LW** *addr data*
MD	Memory Display	**MD** [*width*] [*begin*] [*end*]
MM	Memory Modify	**MM** [*width*] *addr* [*data*]
PING	Network Ping	**PING**
RD	Register Display	**RD** [*reg*]
RM	Register Modify	**RM** *reg data*
RESET	Reset	**RESET**
SET	Set Configurations	**SET** [*option value*]
SHOW	Show Configurations	**SHOW** [*option*]
STEP	Step (Over)	**STEP**
STORE	Store Configuration	**STORE**
SYM	Symbol Management	**SYM** [*symb*] [**-a** *symb value*] **[-r** *symb*] [**-c|l|s**]
TRACE	Trace (Into)	**TRACE** [*num*]
VER	Show Version	**VER**

Symbols

Symbol tables can be downloaded and symbols can be set with the SYM command. For downloaded files, only COFF and ELF formats have symbol tables. Binary and S-record format files do not have symbol tables. COFF is an MC68000 family-specific format. For Power PC, the ELF format is the only format that contains symbols.

Symbols can be set manually with the SYM debug command.

Data Width Modifiers

For commands that accept an optional *width* to modify the memory access size, the valid values are:

.b	8-bit (byte) access
.h	16-bit (half-word) access
.w	32-bit (word) access

When no *width* option is provided, the default width is **.w** for 32 bits.

ASM - Assembler

Usage **ASM** [*addr* [*stmt*]]

The ASM command is a primitive assembler. The *stmt* is assembled and the resulting code placed at *addr*. This command has an interactive and non-interactive mode of operation.

The value for address *addr* may be an absolute address specified as a hexadecimal value, or a symbol name. The value for *stmt* must be a valid assembler mnemonic for the CPU.

For the interactive mode, the user enters the command and the optional *addr*. If the address is not specified, then the last address is used. The memory contents at the address are disassembled, and the user prompted for the new assembly. If valid, the new assembly is placed into memory, and the address incremented accordingly. If the assembly is not valid, then memory is not modified, and an error message produced. In either case, memory is disassembled and the process repeats.

The user may press **ENTER** or **RETURN** to accept the current memory contents and skip to the next instruction, or a enter period to quit the interactive mode.

In the non-interactive mode, the user specifies the address and the assembly statement on the command line. The statement is the assembled, and if valid, placed into memory, otherwise an error message is produced.

Examples To place a NOP instruction at address 0x00010000, the command is:

```
asm 10000 nop
```

To interactively assembly memory at address 0x00400000, the command is:

```
asm 400000
```

BF - Block Fill

Usage **BF** [*width*] *begin end data* [*inc*]

The BF command fills a contiguous block of memory starting at address *begin,* stopping at address *end,* with the value *data*. *width* modifies the size of the data that is written.

The value for addresses *begin* and *end* may be an absolute address specified as a hexadecimal value or a symbol name. The value for *data* may be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

The optional value *inc* can be used to increment (or decrement) the data value during the fill.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples To fill a memory block starting at 0x00010000 and ending at 0x00040000 with the value 0x1234, the command is:

```
bf 10000 40000 1234
```

To fill a block of memory starting at 0x00010000 and ending at 0x0004000 with a byte value of 0xAB, the command is:

```
bf.b 10000 40000 AB
```

To zero out the BSS section of the target code (defined by the symbols bss_start and bss_end), the command is:

```
bf bss_start bss_end 0
```

BM - Block Move

Usage **BM** *begin end dest*

The BM command moves a contiguous block of memory starting at address *begin* and stopping at address *end* to the new address *dest*. The BM command copies memory as a series of bytes and does not alter the original block.

The values for addresses *begin*, *end*, and *dest* may be absolute addresses specified as hexadecimal values or symbol names. If the destination address overlaps the block defined by *begin* and *end*, an error message is produced, and the command exits.

Examples To copy a block of memory starting at 0x00040000 and ending at 0x00080000 to the location 0x00200000, the command is:

```
bm 40000 80000 200000
```

To copy the target code's data section (defined by the symbols data_start and data_end) to 0x00200000, the command is:

```
bm data_start data_end 200000
```

BS - Block Search

Usage **BS** [*width*] *begin end data*

The BS command searches a contiguous block of memory starting at address *begin*, stopping at address *end*, for the value *data*. *width* modifies the size of the data that is compared during the search.

The values for addresses *begin* and *end* may be absolute addresses specified as hexadecimal values, or symbol names. The value for *data* may be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples To search for the 16-bit value 0x1234 in the memory block starting at 0x00040000 and ending at 0x00080000:

MC68000 and ***ColdFire:***

bs 40000 80000 1234

PowerPC:

bs.h 40000 80000 1234

This reads the 16-bit word located at 0x00040000 and compares it against the 16-bit value 0x1234. If no match is found, then the address is incremented to 0x00040002 and the next 16-bit value is read and compared.

To search for the 32-bit value 0xABCD in the memory block starting at 0x00040000 and ending at 0x00080000:

MC68000 and ***ColdFire***:

bs.l 40000 80000 ABCD

PowerPC:

bs 40000 80000 ABCD

This reads the 32-bit word located at 0x00040000 and compares it against the 32-bit value 0x0000ABCD. If no match is found, then the address is incremented to 0x00040004 and the next 32-bit value is read and compared.

DC - Data Conversion

Usage **DC** *data*

The DC command displays *data* in hexadecimal, binary, and decimal notation.

The value for *data* may be a symbol name or an absolute value. If an absolute value passed into the DC command is prefixed by ‘0x’, then *data* is interpreted as a hexadecimal value. Otherwise *data* is interpreted as a decimal value.

All values are treated as 32-bit quantities.

Examples To display the decimal equivalent of 0x1234, the command is:

dc 0x1234

To display the hexadecimal equivalent of 1234, the command is:

dc 1234

DIS - Disassemble

Usage **DIS** [*addr*]

The DIS command disassembles target code pointed to by *addr*. The value for *addr* may be an absolute address specified as a hexadecimal value or a symbol name.

Wherever possible, the disassembler will use information from the symbol table to produce a more meaningful disassembly. This is especially useful for branch target addresses and subroutine calls.

The DIS command attempts to track the address of the last disassembled opcode. If no address is provided to the DIS command, the DIS command uses the address of the last opcode that was disassembled.

Examples To disassemble code that starts at 0x00040000, the command is:

dis 40000

To disassemble code of the C function main(), the command is:

dis main

DL - Download Console

Usage **DL** [*offset*]

The DL command performs an S-record download of data obtained from the console, typically a serial port. The value for *offset* is converted according to the user-defined radix, normally hexadecimal.

If *offset* is provided, the destination address of each S-record is adjusted by *offset*.

The DL command checks the destination download address for validity. If the destination is an address outside the defined user space, an error message is displayed and downloading aborted.

If the S-record file contains the entry point address, the program counter is set to reflect this address.

Examples To download an S-record file through the serial port, the command is:

dl

To download an S-record file through the serial port and adjust the destination address by 0x40, the command is:

dl 0x40

DN - Download Network

Usage **DN** [**-c**] [**-e**] [**-i**] [**-s**] [**-o** *offset*] [*filename*]

The DN command downloads code from the network. The DN command handle files which are either S-record, COFF, ELF, or Image formats. The DN command uses Tiny File Transfer Protocol (TFTP) to transfer files from a network host.

In general, the type of file to be downloaded and the name of the file must be specified to the DN command. The **-c** option indicates a COFF download; the **-e** option indicates an ELF download; the **-i** option indicates an Image download, and the **-s** indicates an S-record download. The **-o** option works only in conjunction with the **-s** option to indicate an optional *offset* for S-record download. The *filename* is passed directly to the TFTP server and, therefore, must be a valid file name on the server.

If neither of the **-c**, **-e**, **-i**, **-s** or *filename* options are specified, a default file name and file type will be used. Default file name and file type parameters are manipulated using the SET and SHOW commands.

The DN command checks the destination download address for validity. If the destination is an address outside the defined user space, an error message is displayed and downloading is aborted.

For ELF and COFF files, which contain symbolic debug information, the symbol tables are extracted from the file during download and used by dBUG. Only global symbols are kept in dBUG. The dBUG symbol table is not cleared prior to downloading, so it is the user's responsibility to clear the symbol table as necessary, prior to downloading.

If an entry point address is specified in the S-record, COFF, or ELF file, the program counter is set accordingly.

Examples To download an S-record file with the name "srec.out", the command is:

```
dn -s srec.out
```

To download a COFF file with the name "coff.out", the command is:

```
dn -c coff.out
```

To download a file using the default file type with the name "bench.out", the command is:

```
dn bench.out
```

To download a file using the default file name and file type, the command is:

```
dn
```

FS - FLASH Status

Usage **FS**

The FS command gives the status of the on-board FLASH memory. Both the socketed FLASH and the FLASH SIMMs are included. The status information that is displayed includes the type of FLASH, socketed or SIMM, the FLASH size, the manufacturer and device ID numbers, and the FLASH usage. The usage can indicate “dBUG” which means that the associated FLASH contains the dBUG firmware. When programming or erasing FLASH, any dBUG firmware FLASH will require extra confirmation from the user before it can be re-programmed or erased so that the dBUG firmware is not unintentionally overwritten. Other usage types are “used” which indicate that the FLASH is programmed. The usage of “----” indicates that the FLASH is erased.

Example

To display the FLASH status, the command is:

fs

For a board which has a single socketed FLASH device, the status display will look similar to the following display:

```
8M X 8 X 1 Bank = 8 MBytes : Manufacturer ID = 01, Device ID = 93
FF800000:---- FF810000:---- FF820000:---- FF830000:used
FF840000:---- FF850000:used FF860000:used FF870000:used
...
FFF00000:dBUG FF850000:----  FF860000:---- FF870000:----
```

FE - FLASH Erase

Usage

FE *begin* [*end*]

The FE command erases FLASH memory starting at address *begin* and stopping at address *end*. If *end* is not specified, the memory is erased to the end of the bank.

The value for addresses *begin* and *end* may be an absolute address specified as a hexadecimal value or a symbol name.

As a precaution, the user must confirm that the area of FLASH needs be erased before the FE command erases the memory. If the FLASH to be erased contains dBUG firmware, an additional confirmation is required.

Examples

To erase a FLASH block starting at 0xFF650000 and ending at the end of the block (address 0xFF65FFFF), the command is:

fe FF650000

The resulting display and confirmation prompt for a used, non-dBUG bank of FLASH is:

```
Warning : This will erase existing code.
To continue enter YES : YES
Erasing 0xFF650000 - 0xFF65FFFF
.
FLASH processing complete - no errors
```

To erase a FLASH block starting at 0xFF650000 and ending at the end of the block (address 0xFF65FFFF) when the symbol *flsh* has been assigned a value of 0xFF650000, the command is:

fe flsh

The resulting display when the bank of FLASH is already blank is:

```
Erasing 0xFF650000 - 0xFF65FFFF
.
FLASH processing complete - no errors
```

FP - FLASH Program

Usage **FP** *begin end source*

The FP command programs FLASH memory starting at address *begin* and stopping at address *end*. The information to be programmed into the FLASH is retrieved from memory beginning at address *source*.

The value for addresses *begin*, *end*, and *source* may be an absolute address specified as a hexadecimal value or a symbol name.

As a precaution, if the area of FLASH to be programmed is not erased, the user must confirm that FLASH needs to be programmed before the FP command programs the memory. If the FLASH to be re-programmed contains dBUG firmware, an additional confirmation is required.

Example: To program an area of FLASH memory beginning at symbol *flsh* (0xFF650000) and ending at 0xFF650100 from memory beginning at symbol *main*, the command is:

fp flsh ff650100 main

The resulting display for an area of FLASH memory which is already erased is:

```
Programing 0xFF650000 - 0xFF650100
.
FLASH processing complete - no errors
```

GO - Execute

Usage **GO** [*addr*]

The GO command executes target code starting at address *addr*. The value for *addr* may be an absolute address specified as a hexadecimal value or a symbol name.

If no argument is provided, the GO command begins executing instructions at the current program counter.

When the GO command is executed, all user-defined breakpoints are inserted into the target code, and the context is switched to the target program. Control is only regained when the target code encounters a breakpoint, illegal instruction, or other exception which causes control to be handed back to dBUG.

Examples To execute code at the current program counter, the command is:

go

To execute code at the C function main(), the command is:

go main

To execute code at the address 0x00040000, the command is:

go 40000

GT - Execute To

Usage **GT** *addr*

The GT command inserts a temporary breakpoint at *addr* and then executes target code starting at the current program counter. The value for *addr* may be an absolute address specified as a hexadecimal value, or a symbol name.

When the GT command is executed, all breakpoints are inserted into the target code, and the context is switched to the target program. Control is only regained when the target code encounters a breakpoint, illegal instruction, or other exception which causes control to be handed back to dBUG.

Example To execute code up to the C function bench(), the command is:

```
gt bench
```

HELP - Help

Usage **HELP** [*command*]

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of the user code may start is given. If *command* is provided, a brief listing of the syntax of the specified command is displayed.

Examples To obtain a listing of all the commands available within dBUG, the command is:

```
help
```

To obtain help on the breakpoint command, the command is:

```
help br
```

LR - Loop Read

Usage **LR** *addr*

The LR command will repetitively read the memory location at address *addr.* This address can be a RAM, ROM, or peripheral memory-mapped address. The value for address *addr* may be an absolute address specified as a hexadecimal value, or a symbol name. Eight and 16 bit reads are enabled with the **.b** and **.h** extensions to the LR command. The loop read is stopped by hitting any key on dBUG serial port.

This command is intended for use in hardware debug.

Example: To read the 16-bit value at address 0x10000, the command is:

```
lr.h 10000
```

LW - Loop Write

Usage **LW** *addr data*

The LW command will repetitively write the *data* value to the memory location at address *addr.* Be careful specifying this address as there are no checks to ensure that the address is not in a read-only area of memory. The value for address *addr* may be an absolute address specified as a hexadecimal value, or a symbol name. Eight and 16 bit writes are enabled with the **.b** and **.h** extensions to the LW command. The loop write is stopped by hitting any key on dBUG serial port.

This command is intended for use in hardware debug.

Example: To write the 8-bit value 0xF4 at address 0x10000, the command is:

```
lw.b 10000 F4
```

MD - Memory Display

Usage **MD** [*width*] [*begin*] [*end*]

The MD command displays a contiguous block of memory starting at address *begin* and stopping at address *end*. The values for addresses *begin* and *end* may be absolute addresses specified as hexadecimal values or symbol names. *Width* modifies the size of the data that is displayed.

Memory display starts at the address *begin*. If no beginning address is provided, the MD command uses the last address that was displayed. If no ending address is provided, MD will display memory up to an address that is 128 beyond the starting address.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples To display memory at address 0x00400000, the command is:

```
md 400000
```

To display memory in the data section (defined by the symbols data_start and data_end), the command is:

```
md data_start
```

To display a range of bytes from 0x00040000 to 0x00050000, the command is:

```
md.b 40000 50000
```

To display a range of 32-bit values starting at 0x00040000 and ending at 0x00050000, the command is:

```
md.w 40000 50000
```

MM - Memory Modify

Usage **MM** [*width*] *addr* [*data*]

The MM command modifies memory at the address *addr.* The value for address *addr* may be an absolute address specified as a hexadecimal value or a symbol name. *width* specifies the size of the data that is modified. The value for *data* may be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

If a value for *data* is provided, the MM command immediately sets the contents of *addr* to *data*. If no value for *data* is provided, the MM command enters into a loop. The loop obtains a value for *data*, sets the contents of the current address to *data*, increments the address according to the data size, and repeats. The loop terminates when an invalid entry for the data value is entered, i.e., a period.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples

To set the byte at location 0x00010000 to be 0xFF, the command is:

mm.b 10000 FF

To interactively modify memory beginning at 0x00010000, the command is:

mm 10000

PING - Network Ping Client / Server

Usage **PING**

The PING command starts a Ping server and client for the network. This is done by utilizing the ICMP protol's echo_request and echo_response datagrams.

The Ping client sends out five echo_request datagrams to the server set by the SHOW command. It also processes the echo_response datagrams from the server and prints the result.

The Ping server is started at the same time the Ping client is started. It responds with an echo_response datagram for every echo_request datagram it receives. It will also print out a message for every echo_response datagram it receives. When the message:

Press ENTER to end

is printed by dBUG, the Ping server can be stopped by hitting **ENTER**.

Example

To start the Ping server and client, the command is:

ping

RD - Register Display

Usage **RD** [*reg*]

The RD command displays the register set of the target. If no argument for reg is provided, all registers are displayed. Otherwise, the value for *reg* is displayed.

dBUG preserves the registers by storing a copy of the register set in a buffer. The RD command displays register values from the register buffer.

Examples

To display all the registers and their values, the command is:

rd

To display only the program counter, the command is:

rd pc

RM - Register Modify

Usage **RM** *reg data*

The RM command modifies the contents of the register *reg* to *data*. The value for *reg* is the name of the register, and the value for *data* may be a symbol name, or it is converted according to the user-defined radix, normally hexadecimal.

dBUG preserves the registers by storing a copy of the register set in a buffer. The RM command updates the copy of the register in the buffer. The actual value will not be written to the register until the target code is executed.

Examples To change register r0 to contain the value 0x1234, the command is:

rm r0 1234

To change link register to contain the value 0x00010000, the command is:

rm lr 10000

RESET - Reset the Board and dBUG

Usage **RESET**

The RESET command resets the board and dBUG to their initial power-on states.

The RESET command executes the same sequence of code that occurs at power-on. If the RESET command fails to reset the board adequately, cycle the power or press the reset button.

Example To reset the board and clear the dBUG data structures, the command is:

reset

SET - Set Configurations

Usage **SET** [*option value*]

The SET command allows the setting of user-configurable options within dBUG. With no arguments, SET displays the options and values available. The standard set of options is listed below.

- **base** - This is the default radix for use in converting a number from its ASCII text representation to the internal quantity used by dBUG. The default is hexadecimal (base 16), and other choices are binary (base 2), octal (base 8), and decimal (base 10).
- **baud** - This is the baud rate for the first serial port on the board. The default baud rate is 9600 bps. All communications between dBUG and the user occur using eight data bits, no parity, and one stop bit, 8N1. Available baud rates are 9600, 19200, 38400, 57600, and 115200 bps.

- **port** - This is the serial port dBUG uses to communicate. Available options are PSC1, PSC2, and PSC3. The default port is PSC1.
- **mac** - This is the Ethernet MAC address used by the board. The default MAC has to be changed to a valid and unique address to guarantee the functionality of your network. Your local network administrator will have this information.
- **server** - This is the network IP address of the machine which contains files accessible via TFTP. Your local network administrator will have this information and can assist in properly configuring a TFTP server if one does not exist.
- **client** - This is the network Internet Protocol (IP) address of the board. For network communications, the client IP is required to be set to a unique value, usually assigned by your local network administrator.
- **gateway** - This is the network IP address of the gateway for your local subnetwork. If the client IP address and server IP address are not on the same subnetwork, this option must be properly set. Your local network administrator will have this information.
- **netmask** - This is the network address mask to determine if use of a gateway is required. This field must be properly set. Your local network administrator will have this information.
- **filename** - This is the default filename to be used for network download if no name is provided to the DN command.
- **filetype** - This is the default file type to be used for network download if no type is provided to the DN command. Valid values are **srecord**, **image**, **coff**, and **elf**.

Examples: To set the baud rate of the board to be 19200, the command is:

set baud 19200

SHOW - Show Configurations

Usage **SHOW** [*option*]

The SHOW command displays the settings of the user-configurable options within dBUG. When no option is provided, SHOW displays all options and values.

Examples To display all options and settings, the command is:

show

To display the current baud rate of the board, the command is:

show baud

STEP - Step Over

Usage **STEP**

The STEP command can be used to "step over" a subroutine call, rather than tracing every instruction in the subroutine. The ST command sets a temporary breakpoint one instruction beyond the current program counter and then executes the target code.

For ***MC68000*** and ***ColdFire***, the STEP command can be used for BSR and JSR instructions.

For ***PowerPC***, the command can be used for BL, BLA, BCL, and BCLA instructions.

The STEP command will work for other instructions as well, but note that if the STEP command is used with an instruction that will not return, i.e., BRA on MC68000 and ColdFire or BA on PowerPC, the temporary breakpoint may never be encountered and dBUG may never regain control.

Examples: To pass over a subroutine call, the command is:

```
step
```

STORE - Store Configuration

Usage **STORE**

The STORE command stores the user-configurable options within dBUG into a non-volatile memory of the board. This guarantees that the options modified by the user are restored after a power-on or reset.

Example: To store the user-configurable options, the command is:

```
store
```

SYM - Symbol Name Management

Usage **SYM** [*symb*] [**-a** *symb value*] [**-r** *symb*] [**-c** | **l** | **s**]

The SYM command adds or removes symbol names from the symbol table. If only a symbol name is provided to the SYM command, then the symbol table is searched for a match on the symbol name and its information displayed.

The **-a** option adds a symbol name and its value into the symbol table. The **-r** option removes a symbol name from the table.

The **-c** option clears the entire symbol table, the **-l** option lists the contents of the symbol table, and the **-s** option displays usage information for the symbol table.

Symbol names contained in the symbol table are truncated to 31 characters. Any symbol table lookups, either by the SYM command or by the disassembler, will only use the first 31 characters. Symbol names are case-sensitive.

Examples: To define the symbol "main" to have the value 0x00040000, the command is:

```
sym -a main 40000
```

To remove the symbol "junk" from the table, the command is:

```
sym -r junk
```

To see how full the symbol table is, the command is:

```
sym -s
```

To display the symbol table, the command is:

```
sym -l
```

TRACE - Trace Into

Usage TRACE [*num*]

The TRACE command allows single-instruction execution. If *num* is provided, then *num* instructions are executed before control is handed back to dBUG. The value for *num* is a decimal number.

The TRACE command sets bits in the processors' supervisor registers to achieve singleinstruction execution, and the target code executed. Control returns to dBUG after a singleinstruction execution of the target code.

Examples To trace one instruction at the program counter, the command is:

```
tr
```

To trace 20 instructions from the program counter, the command is:

```
tr 20
```

VER - Display dBUG Version

Usage **VER**

The VER command displays the version information for dBUG. The dBUG version, build number and build date are all given.

The version number is separated by a decimal, for example, "v 2b.1c.1a". In this example:

2b	dBUG common major and minor revision
1c	CPU major and minor revision
1a	board major and minor revision

The version date is the day and time at which the entire dBUG monitor was compiled and built.

Example To display the version of the dBUG monitor, the command is:

```
ver
```

Supported Registers

The core PowerPC register set is maintained by dBUG.

- GPR0-31 (referenced as R0-R31)
- IP (SRR0 is IP)
- MSR (SRR1 is MSR)

- CR, CTR, XER, LR

Most registers are accessible via their symbolic names as well as the special-purpose register number. For instance, the Link Register, SPR8, can be referenced as both “spr8” and “LR.”

Additionally, the following MPC6XX family of processors registers are supported:

- FPSCR, FPR0-31 (referenced as f0-f31)
- DEC, PVR, TBL, TBU and SR0-15
- IBATxL, IBATxU, DBATxL, DBATxU, SDR1, DAR, and DSISR

Additional registers are maintained according to the MPC6XX processor in the system.

MPC602	None.
MPC603	• HID0, DMISS, DCMP, HASH1, HASH2, IMISS, ICMP, and RPA • IABR and EAR
MPC603e and MPC603ev	• HID0, HID1, DMISS, DCMP, HASH1, HASH2, IMISS, ICMP, and RPA • IABR and EAR
MPC603eh (MGT5100)	• HID0, HID1, HID2, CSRR0 CSRR1, SPRG4-7, and EAR • IBAT4-7U, IBAT4-7L, DBAT4-7U, DBAT4-7L, DMISS, and DCMP • HASH1, HASH2, IMISS, ICMP, RPA, DABR2, DBCR, andIBCR • IABR, IABR2, DABR, and PIR
MPC604	• HID0, PMC1, PMC2, MMCR0, SDA, SIA, IABR, DABR, EAR, and PIR
MPC604e and MPC604ev	• HID0, PMC1, PMC2, PMC3, PMC4, MMCR0, and MMCR1 • SDA, SIA, IABR, DABR, EAR, and PIR
MPC740 and MPC750	• UPMC1, UPMC2, UPMC3, UPMC4, USIA, UMMCR0, UMMCR1, HID0, and HID1 • PMC1, PMC2, PMC3, PMC4, MMCR0, MMCR1, and SIA • THRM1, THRM2, THRM3, ICTC, L2CR, IABR, DABR, and EAR

Configuring for Network Downloads

Chapter 3

dBUG is capable of downloading over an Ethernet network using the Trivial File Transfer Protocol (TFTP). Prior to using this feature, several parameters are required for network downloads to occur. The information that is required and the steps for configuring dBUG are described in the following paragraphs.

Required Network Parameters

For performing network downloads, dBUG needs six parameters; four are network-related, and two are download-related. The parameters follow with the dBUG designation in parenthesis.

All computers connected to an Ethernet network using the Internet Protocol (IP) need three network-specific parameters. These parameters are:

- IP address for the dBUG-based computer (*client*)
- IP address of the gateway for non-local traffic (*gateway*)
- Network IP netmask for flagging traffic as local or non-local (*netmask*)

In addition, the dBUG network download command requires the following three parameters:

- • IP address of the TFTP server (*server*)
- • Default name of the file to download (*filename*)
- • Default type of the file to download (*filetype*)

Your local system administrator can assign a unique IP address for the board and also provide you the IP addresses of the gateway, netmask, and TFTP server. Fill out the lines of information that follow.

Client:	___.___.___.___	(IP address of the board)
Server:	___.___.___.___	(IP address of the TFTP server)
Gateway:	___.___.___.___	(IP address of the gateway)
Netmask:	___.___.___.___	(Network netmask)

Configuring dBUG Network Parameters

Once the network parameters have been obtained, dBUG must be configured. The following commands are used to configure the network parameters.

set client *client IP*

set server *server IP*

set gateway *gateway IP*

`set netmask` *netmask*

For example, the TFTP server is named *santafe* and has IP address *123.45.67.1*. The board is assigned the IP address of *123.45.68.15*. The gateway IP address is *123.45.68.250*, and the netmask is *255.255.255.0*. The commands to dBUG are:

```
set client 123.45.68.15
set server 123.45.67.1
set gateway 123.45.68.250
set netmask 255.255.255.0
```

The last step is to inform dBUG of the name and type of the file to download. Prior to giving the name of the file, keep in mind the following: Most, if not all, TFTP servers will only permit access to files starting at a particular sub-directory. (This is a security feature that prevents reading of arbitrary files by unknown persons.) For example, SunOS uses the directory `/tftp_boot` as the default TFTP directory. When specifying a file name to a SunOS TFTP server, all file names are relative to `/tftp_boot`. As a result, normally files must be copied into the directory used by the TFTP server.

A default file name for network downloads is maintained by dBUG. To change the default file name, use the command:

set filename *filename*

When using the Ethernet network for downloading, either S-record, COFF, Elf, or Image files may be downloaded. A default file type for network downloads is maintained by dBUG as well. To change the default file type, use the command:

set filetype srecord | coff | elf | image

Continuing with the previous example, the compiler produces an executable COFF file, `a.out`. This file is copied to the `/tftp_boot` directory on the server with the command:

```
rcp a.out santafe:/tftp_boot/a.out
```

Change the default file name and file type with the commands:

```
set filename a.out
set filetype coff
```

Finally, perform the network download with the DN command. The network download process uses the configured IP addresses and the default file name and file type for initiating a TFTP download from the TFTP server.

Troubleshooting Network Problems

Most problems related to network downloads are a direct result of improper configuration. Verify that all IP addresses configured into dBUG are correct. This is accomplished via the SHOW command.

Using an IP address that is already assigned to another machine will cause the dBUG network download to fail and will probably cause other severe network problems. Make certain the client IP address is unique for the board.

Check for proper insertion or connection of the network cable. Are status LEDs lit to indicate that network traffic is present?

Check for proper configuration and operation of the TFTP server. Most Unix workstations can execute the command TFTP which can be used to connect to the TFTP server, as well. Is the default TFTP root directory present and readable?

If *ICMP_DESTINATION_UNREACHABLE* or similar ICMP messages appear, a serious error has occurred. Reset the board and wait one minute for the TFTP server to time out and terminate any open connections. Verify that the IP addresses for the server and gateway are correct.

Embedded Planet
4760 Richmond Road, Suite 400
Warrensville Heights, OH 44128
www.embeddedplanet.com

Phone: 216.245.4180
Fax: 216.292.0561

Form 700M0125RA1   Litho in U.S.A. Nov2004