D-Bug12XZ
Reference Manual

HCS12(X/Z)
Microcontrollers

DBUG12XZRMV1

Rev. 1.06

2/2016

nxp.com

Date	Revision	Description
11/2013	1.02	Added note to BR command description regarding mapping of RAM to 0x4000 – 0x7fff in S12XE devices.
11/2014	1.03	Added notes to the end of Section 3.0 on new options available when unsecuring target devices.
1/2016	1.04	Added documentation on TRACE command which was first added to v6.0.0b15.
1/2016	1.05	Added documentation on new feature added to the TRACE command.
2/2016	1.06	Updated documentation on the TRACE command to reflect changes in display of data in “HISTORY” option. Other minor updates throughout text.

Reference Guide
For
D–Bug12XZ Version 6.x.x

A Debug Monitor
For
The S12, S12X and S12Z Microcontrollers

Written By
Gordon Doughman
Field Applications Engineer
Software Specialist

1.0 Introduction

The first version of D-Bug12 was originally written as a ROM monitor for the HC12 microcontroller family as an aid to debug the initial silicon of the MC68HC812A4. Subsequent versions added the ability to operate in 'Pod' mode, which allowed D-Bug12 to communicate with a target HC12/S12 device through its BDM interface. This version of D-Bug12, D-Bug12XZ, supports the S12 and S12X 0.25μ and 0.18μ families and the new S12Z family of devices. It is designed to operate in 'Pod' mode only and run on NXP's BDM Flash Programmer (P/N LFBDMPGMR) or equivalent hardware. See Appendix C for the requirements of equivalent hardware. In addition to supporting the CPU12 on S12 devices and the CPU12X on S12X devices, D-Bug12XZ also contains support for the XGate core.

2.0 Terminal Emulator Requirements

The D-Bug12XZ user interface consists of a simple command line interface controlled through a terminal emulator program running on a host computer. The communications settings required by D-Bug12XZ are listed in the table in Figure 1 below.

Parameter	Setting
Baud	600 – 230,400
Data Bits	8
Stop/Start bits	1
Parity	None
Handshake	XOn/XOff

Figure 2.1, Communications Settings

The D-Bug12XZ firmware is designed to automatically detect standard baud rates from 600 to 230,400 baud. Although not supported by most terminal emulation programs, the D-Bug12X Zfirmware also supports a communication rate of 500,000 baud. Note that built in serial ports in many PCs may not support Baud rates above 115,200. To achieve a communications rate above this will require the use of a PCI expansion card, a USB-to-Serial adapter or use of the built-in USB interface on the LFBDMPGMR.

Each time the hardware is powered up or reset, the firmware waits for the reception of an ASCII carriage return (0x0d) to determine the communication speed. This communication rate is saved and used until the next time the Programmer is powered up or reset. Entering a character other than an ASCII carriage return may cause the programmer to select an incorrect baud rate and result in incorrect characters being displayed in the terminal window. Pressing the System Reset button on the programmer and entering a carriage return from the keyboard will remedy this situation. Note that the auto baud detection feature may be disabled through the use of the BAUD command. See Section 7.1 of this manual for details and cautions regarding this option.

2.1 Terminal Communications Programs

Many popular terminal emulation programs for the Windows™ operating system meet the BDM Programmer's communication requirements. The Hyperterminal communications program, included with all versions of Windows™ through Windows™ XP, is one such program. For Windows™ Vista or other Windows™ installations that do not include Hyperterminal, a free terminal program named TeraTerm can be downloaded from the following web site:

http://ttssh2.sourceforge.jp/

Note: When D-Bug12XZ is connected to an S12(X) device, the terminal emulation program can be set to a width of 80 characters or greater. When D-Bug12XZ is connected to an S12Z device, the terminal emulation program should be set to a width of 100 characters or greater.

2.2 D-Bug12XZ Installation

As mentioned in the Introduction section, D-Bug12XZ was written to run on NXP's BDM Flash Programmer. As the Programmer is shipped from the factory, it contains the most recent version of the BDM Flash Programmer firmware and a Serial Bootloader. The Serial Bootloader can be used to replace the Programmer firmware with the D-Bug12XZ firmware. Before installing the D-Bug12XZ firmware in the programmer, be sure that the software that came with the Programmer is installed on the host computer. In addition to the Programmer GUI software, the installation software also installs USB virtual COM port drivers. These drivers allow the Programmer's USB port to be accessed as a serial COM port by any terminal emulator program.

Please refer to the documentation that comes with the BDM Flash Programmer for details on installing the software and drivers.

2.2.1 Serial Bootloader

The S-Record bootloader program utilizes the on-chip SCI for communications and does not require any special programming software on the host computer. The only host software required is a simple terminal program that is capable of communicating at 9600 – 230,400 baud, supports XOn/XOff handshaking and can send an S-Record text file to the programmer.

Invoking the bootloader is accomplished by pressing and holding the 'Program Start' push button while pressing and releasing the 'System Reset' pushbutton. Invoking the bootloader causes the prompt similar to the one in Figure 2.2 to be displayed on the host terminal's screen.

```
BDMPgmr Bootloader v2.0.5

a.) Erase Flash
b.) Program Flash
c.) Set Baud Rate
d.) Execute Application
e.) Erase EEE
?
```

Figure 2.2, Serial Bootloader Prompt

Note: When entering the bootloader by pressing and holding the 'Program Start' push button while pressing and releasing the 'System Reset' pushbutton, the bootloader communication rate defaults to 9600 baud.

Note: Only lower case letters are recognized for bootloader command entry to prevent unintentional command execution in the case that the 'Program Flash' command fails.

2.2.2 Erase Flash Command

Selecting the Erase function by typing a lower case '`a`' on the terminal erases all of the Programmer's Flash except for the portion where the S-Record bootloader resides. After the erase operation is completed, a verify operation is performed to ensure that all locations were properly erased. If the erase operation is successful, the bootloader's prompt is redisplayed.

If any locations were found to contain a value other than $FF, an error message is displayed on

the screen and the bootloader's prompt is redisplayed. If the Flash will not erase after one or two attempts the device may be damaged.

2.2.3 Program Flash Command

To increase the efficiency of the programming process, the S-Record bootloader uses interrupt driven, buffered serial I/O in conjunction with XOn/XOff software handshaking to control the flow of S-Record data from the host computer. This allows the bootloader to continue receiving S-Record data from the host computer while the data from the previously received S-Record is programmed into the Flash. The terminal program **must** support XOn/XOff handshaking to properly reprogram the Programmer's Flash memory.

Typing a lower case '`b`' on the terminal causes the bootloader to enter the programming mode and wait for S-Records to be sent from the host computer. The bootloader will continue to receive and process S-Records until it receives an 'S8 or 'S9' end of file record. If the object file being sent to the bootloader does not contain an 'S8' or 'S9' record, the bootloader will not return to its prompt and will continue to wait for the end of file record. Pressing and holding 'Program Start' push button while pressing and releasing the 'System Reset' pushbutton will cause the bootloader to revert to 9600 baud and display the prompt.

If a Flash memory location will not program properly, an error message is displayed on the terminal screen and the bootloader's prompt is redisplayed. If the Programmer's Flash will not program after one or two attempts the device may be damaged or an S-Record with a load address outside the range of the available on-chip Flash may have been received. The S-Record data must have load addresses in the range $700000 - $7FDFFF.

2.2.4 Set Baud Rate Command

While the default communications rate of the bootloader is 9600 baud, this speed is much too slow if the majority of the Programmer's Flash is to be programmed, however, it provides the best compatibility for initial communications with most terminal programs. The Set Baud Rate command allows the bootloader communication rate to be set to one of four standard baud rates.

Typing a lower case '`c`' on the terminal causes the prompt shown in Figure 2.3 to be displayed on the host terminal's screen. Entering a number '1' through '5' on the keyboard will select the associated baud rate and issue a secondary prompt indicating that the terminal baud rate should be changed. After changing the terminal baud rate, pressing the enter or return key will return to the main bootloader prompt.

```
1.) 9600
2.) 38400
3.) 57600
4.) 115200
5.) 230400
? 4
Change Terminal Baud Rate and Press Return
```

Figure 2.3, Change Baud Rate Prompt

2.2.5 Execute Application Command

The 'Execute Application' command must be used if the bootloader was entered through the use of the BDM Programmer's `BOOT` command. See the description of the BOOT command for further details.

Before jumping to the location pointed to by the secondary reset vector, the contents of the secondary reset vector is checked to see if it contains a value other than the erased state of the Flash ($FFFF). If so, it jumps to the start of the firmware.

2.2.6 Erase EEE Command

The D-Bug12XZ firmware saves various options and settings in the Programmer MCU's Emulated EEPROM (EEE). One of these settings is the communications BAUD rate. If the automatic BAUD rate detection feature is turned off (see BAUD command for details), a fixed BAUD rate, stored in the EEE is used after power up or reset of the Programmer. In this circumstance, if communication cannot be established after trying several standard BAUD rates, the bootloader can be used to erase the MCU's EEE which will return all settings to their factory default.

3.0 Initial Communication With D-Bug12XZ

As mentioned in Section 2.0, Terminal Emulator Requirements, after the Programmer hardware is powered up or reset, by default, the firmware waits for the reception of an ASCII carriage return (0x0d) to determine the communication speed. If the Programmer is not connected to a target device, the prompt in Figure 3.1 is displayed. If the Programmer is connected to a target device a prompt similar to that in Figure 3.1 or the prompt Figure 3.2 will be displayed depending on the state of the target.

```
D-Bug12XZ 6.0.0b17
Copyright 1996 - 2016 NXP Semiconductors
For Commands type "Help"


Target VDD is not present.

Current Target Architecture: S12(X)

1.) Reset Target
2.) Hot Connect to Target
3.) Erase & Unsecure
4.) Enter BDM debugger
5.) Set MCU Family: S12(X)
6.) Set MCU Family: S12Z
?
```

Figure 3.1, D-Bug12XZ Prompt 1

```
D-Bug12X 6.0.0b17
Copyright 1996 - 2016 NXP Semiconductors
For Commands type "Help"

R>
```

Figure 3.2, D-Bug12XZ Prompt 2

After the reception of the carriage return D-Bug12XZ sets the SCI baud rate and displays the sign-on message which contains the firmware version and copyright information. Following this, the firmware attempts to communicate with a target MCU of the currently selected architecture. If the current target architecture is incorrect, communication will fail. Item '5' or '6' should be chosen to select the proper target architecture.

Once the correct target architecture is chosen and the debugger hardware is properly connected to a target device, an attempt is made to communicate with a connected target. If the target's V_{DD} level cannot be measured or is out of range, a message indicating such, is displayed as shown in Figure 3.1. Note that D-Bug12XZ requires target V_{DD} to be present on pin 6 of the target BDM connector to establish communication with the target.

Communication with the target begins by using the BDM SYNC command to measure the target's BDM clock frequency. If the SYNC command can properly measure the target's BDM clock frequency and can establish BDM communication, D-Bug12XZ will automatically 'hot connect' with the target displaying the prompt shown in Figure 3.2 with either the '`R>`' or '`S>`'.

NOTE: In D-Bug12 v4.x.x entering any character other than the numbers listed in the menu choice would cause the target device to be reset. This behavior posed a danger of accidentially resetting a target when attempting to perform a 'hot connect'. The behavior of D-Bug12XZ was modified so that if any character or number other than those listed in the menu is entered, communication with the target is attempted and the menu redisplayed if BDM communication can not be established.

If communication cannot initially be established with a connected device and target V_{DD} is present on pin 6 of the target BDM connector, one of several conditions may exist. For some older S12 0.25μ devices, the BDM module does not support the SYNC command. Therefore, to hot connect to such a target device, the target device's crystal frequency must be specified by chosing option 2.

As shown in Figure 3.3, D-Bug12XZ will ask for the target crystal frequency. If, after entering the target crystal frequency the detected target device is an MC9S12DP256, mask sets 0K79X, 1K79X, or 2K79X or an MC9S12H256 mask sets 0K78X, 1K78X or 2K78X D-Bug12XZ will request the target **bus** speed. This information is required to maintain reliable BDM communication with the target due to a BDM synchronization errata that exists in these devices.

```
1.) Reset Target
2.) Hot Connect to Target
3.) Erase & Unsecure
4.) Enter BDM debugger
5.) Set MCU Family: S12(X)
6.) Set MCU Family: S12Z
? 2
Enter Target Crystal Frequency (kHz): 8000
Enter Target Bus Frequency (kHz): 24000
R>
```

Figure 3.3, Hot Connect option

If repeated attempts to establish BDM communication with the target fails, it is possible that the target MCU is secured. If D-Bug12XZ can measure the target's BDM clock frequency through the use of the SYNC command but cannot establish communication with the target, it will display a message indicating the target may be secured before displaying the menu. One way to verify if the target device is secured is to use the BDM Debugger, option 4, to attempt to read the PARTID register at 0x001a for S12 and S12X. Even if an S12 or S12X device is secured, all of the I/O registers can be read and written. For S12Z devices, ***no*** target memory locations can be read when the target device is secured. Attempted reads of target memory locations using the BDM debugger will return a value of 0xEE. Details on the BDM debugger can be found in Appendix A and B.

If a target device is secured, choosing option three will cause D-Bug12XZ to attempt to completely erase the target's Flash and EEPROM. If this is successful, the target's security byte will be programmed with a value of 0xfe, placing the target device in the unsecured state. If the erase operation or programming of the security byte fails, an error message will be displayed and the "Can't communicate…" menu will be redisplayed.

When attempting to communicate with the target to perform the erase and unsecure operation, if the target device's PARTID does not match any of the devices in D-Bug12XZ's device table, a menu, as shown in Figure 3.4, is displayed allowing the connected device's part family to be chosen. Note that option three, "S12P/S12G" applies to any S12 devices manufactured using 0.18μ device geometry.

```
Select Target MCU Type:
1.) S12/S12XD
2.) S12XE/S12XS
3.) S12P/S12G
?
```

Figure 3.4, "Erase and Unsecure" Device Family Selection

If the incorrect device family is chosen for the connected device, the erase and unsecure operation will likely fail.

3.1 Command Prompt

The command line prompt displayed by D-Bug12XZ indicates the current state of the target. The 'S>' prompt indicates that the target CPU12 or CPU12X is in active background mode. In this mode, all of D-Bug12XZ's commands, as they apply to the target MCU, may be used. When the 'R>' prompt is displayed, the CPU12 or CPU12X is running user application code.

3.2 Command Syntax

Each of D-Bug12XZ's commands is entered following the 'S>' or 'R>' prompt with each command name followed by zero or more command line parameters.

3.2.1 Target Addresses

Many of D-Bug12XZ's commands, such as the Memory Display (MD) and Memory Modify (MM), require one or more target addresses as command line parameters. To support the paged architecture of the S12 and S12X, D-Bug12XZ accepts addresses in one of four formats: Local, Paged, Global/Linear and XGate.

Any hexadecimal address entered with a value less than or equal to 0xffff is interpreted as a Local address. Local addresses allow access to the CPU's 64K address space.

A Paged address is composed of two parts. An 8-bit value specifying the page number and a 16-bit value specifying the page window address. The 8-bit page number, which is enrtered first, is separated from the 16-bit page window address by the ASCII colon character (":"). Valid 8-bit page numbers and page window addresses will vary depending on the target MCU.

Any hexadecimal address entered with a value greater than or equal to 0x10000 is interpreted as a Global or linear address. For 0.25μ S12 devices, these addresses are interpreted as a linear Flash address. Only Flash memory can be accessed on these devices using linear addresses. For S12 0.18μ devices, a global address has the ability to access the entire 256k Global address space. On S12X devices, Global addresses provide access to the 8MB Global address space. Any entered address can be specifically designated as a Global or linear address by appending "`'G`" to the address. The letter "G" may be upper or lower case. This allows entered addresses less than 0x10000 to be interpreted as a Global or linear address.

The XGate coprocessor on the S12XD and S12XE families is a 16-bit RISC CPU that has the ability to address 64K of memory. An XGate address is specified by appending "`'X`" to a 16-bit address.

When connected to a target device containing the S12Z CPU core only linear addresses with a value between 0x000000 and 0xffffff are accepted as target addresses.

4.0 Command Summary

The following list summarizes the D-Bug12XZ command set. Each command's function and command line syntax are described in detail in the following sections.

- ASM - Single line assembler/disassembler.
- BAUD - Set the SCI communications BAUD rate.
- BDMDB - Enter the BDM command debugger.
- BF - Block Fill user memory with data.
- BR - Set/Display user breakpoints.
- BS – Block Search, search a block of memory for a specified data pattern.
- BULK - Bulk erase on-chip EEPROM.
- CALL - Execute a user subroutine ending with RTS and return to command prompt.
- CALLF - Execute a user subroutine ending with RTC and return to command prompt.
- DEVICE – Display list of current/all supported target MCU device(s).
- EEE - Enable, Disable or Query EEE
- FBULK - Erase the target processor's on-chip Flash.
- FLOAD - Program the target processor's on-chip Flash from S-Records.
- G – Begin execution of user program.
- GT - Go Till, Set a temporary breakpoint and begin execution of user program.
- HELP - Display D-Bug12XZ command set and command syntax summary.
- LOAD - Load user program into RAM in S-Record format.
- LOG – Log target data to terminal display.
- MD - Memory Display, display memory contents in hex bytes/ASCII format.
- MDW - Memory Display Words, display memory contents in hex words/ASCII format.
- MM - Memory Modify, interactively examine/change memory contents.
- MMW - Memory Modify Words, interactively examine/change memory contents.
- MOVE - Move a block of memory.
- NOBR - Remove one/all user breakpoints.
- PARTDF - Display/Set EEE/DFlash Partition.
- RD - Register Display, display the CPU register contents.
- RESET - Reset the target MCU.
- RM - Register Modify, interactively examine/change CPU register contents.
- STOP - Stop the execution of user code and place the target processor in background mode.
- SECURE - Secure target device.
- T – Trace, execute an instruction, disassemble it, and display the CPU registers.
- TO – Trace Over subroutine calls (JSR, BSR, CALL).
- TRACE – Configure or display data from debug module trace buffer.
- UPLOAD - Display memory contents in S-Record format.
- VER - Display the version of D-Bug12X.
- VERF - Verify memory contents against S-Record Data.
- XASM - XGate single line assembler/disassembler.
- XBR - Set/Display XGate software breakpoints.

- XDBG - Enter/Exit XGate thread debug.
- XG - Continue execution of XGate thread.
- XNOBR - Remove one/all Xgate software breakpoints.
- XRD - Display XGate CPU registers.
- XT - Trace XGate instructions.
- <RegisterName> <RegisterValue> - Set CPU <RegisterName> to <RegisterValue>.

4.1 ALTCLK - Specify An Alternate BDM Communications Rate

Command Line Format

ALTCLK [<AltBDMRate>]

Parameter Description

<AltBDMRate> - A 16-bit decimal number

Command Description

An errata was introduced in the BDM module on the MC9S12DP256 (Barracuda II) 0K79X mask set and the MC9S12H256 (Mako) 0K78X mask set. When using EXTAL ÷ 2 as the BDM clock source (default) and the PLL is selected as the bus clock source, BDM communications will be lost when the PLL multiplier is greater than 2 ((synr+1))/(refdv+1)). Once communication is lost, the only way to regain communications is to reset the target MCU.

D-Bug12XZ configures the BDM to use the target bus clock (BDM Status Register CLKSW=1) if either of these parts is connected as the target device. Because the BDM interface is being driven by the target bus clock, BDM communication will be lost if the target firmware changes the bus clock frequency using the PLL. To prevent the loss of communications from disrupting a debug session, the `ALTCLK` command allows an alternate BDM communication clock frequency to be specified. If BDM communication with the target is lost, D-Bug12XZ will automatically attempt communication at the alternate frequency without notifying the user.

The `ALTCLK` command is used to specify the alternate BDM communication frequency, which should be equal to the target bus frequency with the PLL engaged as the bus clock. For example, if a 4 MHz crystal/oscillator is used in a target application and the firmware programs the PLL to generate a 24 MHz bus clock, the `ALTCLK` command should be used to specify an alternate bus frequency of 24000 KHz. The `ALTCLK` command must be used to specify the alternate BDM communication frequency **before** executing the target code that engages the PLL as the bus clock. Note that the alternate BDM communication rate specified using the `ALTCLK` command is saved in D-Bug12XZ's host MCU EEPROM so that it does not have to be reentered each time the development tool is powered up.

Entering the `ALTCLK` command without an alternate BDM communications frequency will display the current alternate clock setting.

Restrictions

Switching between the two BDM communications rates is completely transparent to the developer with one exception. If D-Bug12XZ's memory modify command (MM) is used to engage the PLL as the bus clock by setting the PLLSEL bit in the CLKSEL register, D-Bug12XZ will report that the target memory could not be modified because of the temporary loss of communications. However, after displaying the error message, D-Bug12XZ will resynchronize to the new BDM communications rate and show that the target memory was properly modified.

The `ALTCLK` command can only be used if the MC9S12DP256 (Barracuda II) 0K79X mask set or the MC9S12H256 (Mako) 0K78X mask set device is connected as the target device.

Example

```
S>ALTCLK 24000
S>ALTCLK
Alternate BDM Clock Frequency (KHz): 24000
S>
```

4.2 ASM - Single Line Assembler/Disassembler Command

Command Line Format

ASM <TAddress>

Parameter Description

<TAddress> A Local, Global/Linear or Paged Target address.

Command Description

The assembler/disassembler is an interactive memory editor that allows memory contents to be viewed and altered using assembly language mnemonics. Each entered source line is translated into machine language code and placed into memory at the time of entry. When displaying memory contents, each instruction is disassembled into its source mnemonic form and displayed along with the hexadecimal machine code and any instruction operands.

Assembler mnemonics and operands may be entered in any mix of upper and lower case letters. Any number of spaces may appear between the assembler prompt and the instruction mnemonic or between the instruction mnemonic and the operand. By default, numeric values appearing in the operand field are interpreted as signed decimal numbers. Placing a $ in front of a number will cause the number to be interpreted as a hexadecimal number.

When an instruction has been disassembled and displayed, the assembler prompt is displayed following the disassembled instruction. If a carriage return is entered immediately following the prompt, the next instruction in memory is disassembled and displayed on the next line.

If a CPU12, CPU12X or CPUS12Z instruction is entered following the prompt, the entered instruction is assembled and placed in memory. The line containing the new entry is erased and the new instruction is disassembled and displayed on the same line. The contents of the next memory location(s) is disassembled and displayed on the screen.

The instruction mnemonics and operand formats accepted by the assembler follow the syntax as described in the CPU12/CPU12X Reference Manual or the CPUS12Z Reference Manual.

When entering branch instructions, the number placed in the operand field should be the absolute destination address of the instruction. The assembler will calculate the twos compliment offset of the branch.

Termination of the assembly/disassembly process is accomplished by entering a period (.) and carriage return immediately following the assembler prompt.

Restrictions

The ASM command cannot be used to write code into on-chip Flash memory.

Example

```
>ASM 700

0700  CC1000          LDD   #4096
0703  1803123401FE    MOVW  #$1234,$01FE
0709  0EF9800001F1    BRSET $003F,PCR,#$01,$0700
070F  18FF            TRAP  $FF
0711  183FE3          ETBL  <Illegal Addr Mode>  >.

>
```

Assembly Operand Format

This section describes the operand format used by the assembler when assembling CPU12 or CPU12X instructions. The operand format accepted by the assembler is described separately in the *CPU12/CPU12X Reference Manual* or *CPU S12Z Reference Manual*. Rather that describe the numeric format accepted for each instruction, some general rules will be used. Exceptions and complicated operand formats are described separately.

In general, anywhere the assembler expects a numeric value in the operand field, either a decimal or hexadecimal value may be entered. Decimal numbers are entered as signed constants having a range of -32768..65535. A leading minus sign (-) indicates negative numbers, the absence of a leading minus sign indicates a positive number. A leading plus sign (+) is not allowed. Hexadecimal numbers must be entered with a leading dollar sign ($) followed by one to four hexadecimal digits. The default number base is decimal.

For all branching instructions, (Bcc, LBcc, BRSET, BRCLR, DBEQ, DBNE, IBEQ, IBNE, TBEQ, TBNE) the number entered in the address portion of the operand field must be the ***absolute address of the branch destination***. The assembler will calculate the two's compliment offset to be placed in the assembled object code.

The D–Bug12X assembler allows an optional # symbol to precede the 8-bit mask value in all bit manipulation instructions (BSET, BCLR, BRSET, BRCLR).

Disassembly Operand Format

This section describes the operand format for the disassembler that is used in conjunction with the single line assembler. The operand format used by the disassembler is described separately in the *CPU12/CPU12X Reference Manual*. Rather that describe the numeric format used for each instruction, some general rules can be applied. Exceptions and complicated operand formats will be described separately.

All numeric values disassembled as hexadecimal numbers will be preceded by a dollar sign ($) to avoid being confused with values disassembled as signed decimal numbers.

For all branch (Bcc, LBcc, BRSET, BRCLR, DBEQ, DBNE, IBEQ, IBNE, TBEQ, TBNE) instructions the numeric value of the address portion of the operand field will be displayed as the hexadecimal ***absolute address of the branch destination***.

All offsets used with indexed addressing modes will be disassembled as ***signed*** decimal numbers with the following exception. When an instruction is disassembled utilizing the program counter as an index register, the offset field will contain an ***absolute hexadecimal address*** rather than a decimal offset. The address is calculated by adding the offset in the object code to the value of the program counter at the end of the instruction. Rather than displaying the index register name as 'PC' the mnemonic 'PCR' is used to indicate that the offset field contains an absolute address.

All addresses, whether direct or extended, will be disassembled as four digit hexadecimal numbers.

All 8-bit mask values (BRSET/BRCLR/ANDCC/ORCC) will be disassembled as two digit hexadecimal numbers.

For bit manipulation instructions (BSET, BCLR, BRSET, BRCLR), the disassembler always displays the # symbol preceding the 8-bit mask value.

All 8-bit immediate values will be disassembled as hexadecimal numbers.

All 16-bit immediate values will be disassembled as hexadecimal numbers.

4.3 BAUD - Change The Communications BAUD Rate

Command Line Format

BAUD <BAUDRate> | AUTOON | AUTOOFF

Parameter Description

<BAUDRate>	A unsigned 32-bit decimal number.
AUTOON	The ASCII characters “AUTOON” in upper, lower or mixed case.
AUTOOFF	The ASCII characters “AUTOOFF” in upper, lower or mixed case.

Command Description

Each time the D-Bug12XZ hardware is powered up or reset, by default, the firmware waits for the reception of an ASCII carriage return (0x0d) to determine the communication speed. This ‘auto baud’ feature can be disabled or reenabled using the BAUD command by using the “AUTOOFF” or “AUTOON” command line parameters, respectively. After disabling the auto baud feature, the default baud rate will be the one last set by the auto baud feature unless the baud rate is changed using the BAUD command.

Anytime a new baud rate is set, it is stored in the Programmer’s on–chip EEE so that the new baud rate is used the next time the programmer is powered up or reset.

Restrictions

Because the <BAUDRate> parameter supplied on the command line is a 32-bit unsigned integer, BAUD rates greater than 65535 baud may be set using this command. The BAUD command will only accept the following standard baud rates: 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 230400 and 500000.

Error Conditions

If <BAUDRate> is not supplied, an appropriate error message shall be issued, command execution will be terminated and the communications BAUD rate will not be changed.

If parameters other than a valid unsigned decimal number for <BAUDRate>, “AUTOOFF” or “AUTOON” appear on the command line, command execution will be terminated and the communications BAUD rate will not be changed.

If the value is supplied for <BAUDRate> other than those listed under **Restrictions**, an appropriate error message shall be issued, command execution will be terminated and the communications BAUD rate will not be changed.

Example

```
S>baud 50
Bad BAUD Rate
S>baud 115200
Change Terminal BR, Press Return
S>
```

4.4 BDMDB - Enter BDM Debugger

Command Line Format

BDMDB

Parameter Description

None.

Command Description

The BDMDB command halts normal D-Bug12XZ operation and enters the BDM debugger. Using D-Bug12X's low level BDM driver routines, the BDM debugger allows individual BDM commands to be sent to a target device directly from the command line. As shown in the example, upon entering the BDM debugger, the command line prompt changes to a question mark (?). Note that while running the BDM debugger, no target BDM communication occurs other than during the execution of a command. Unlike D-Bug12X, when running the BDM debugger, no checks are performed to ensure a valid target connection with a target MCU exists before executing a command.

See Appendix A for a complete description of the BDM debugger commands.

Restrictions

None.

Error Conditions

None.

Example

```
S>bdmdb
BDM Command Debugger
For Commands type "HELP"

?
```

4.5 BF - Fill memory with data

Command Line Format

BF <StartAddress> <EndAddress> [<Data>..<Data>] [;nv]

Parameter Description

<StartAddress>	A Local, Global/Linear, Paged or XGate Target address.
<EndAddress>	A Local, Global/Linear, Paged or XGate Target address.
<Data>	An 8-bit hexadecimal number
;nv	The ASCII string ';nv' in upper, lower or mixed case.

Command Description

The Block Fill command is used to place a pattern of 8-bit values into a range of memory locations. <StartAddress> is the first memory location written with data and <EndAddress> is the last memory location written with data. If the optional <data> pattern is omitted the memory range is filled with the value $00.

Normally the Block Fill command verifies each memory location as it is written. The ';nv' option prevents the Block Fill command from verifying writes to the specified memory range. This option can be useful for testing a range of memory, especially RAM, for defective locations.

Restrictions

The <StartAddress> and <EndAddress> parameters must specify the same memory space. That is, both addresses must be Local, Global/Linear, Paged or XGate target addresses.

The '`;nv`' option must appear on the command line after the specified data pattern.

When the BF command is used to fill the on-chip EEPROM of a target device, the <StartAddress> and <EndAddress> must correspond to target addresses that lie within the EEPROM array.

Error Conditions

If <StartAddress> and <EndAddress> specify different memory spaces, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be altered.

If <StartAddress> is greater than <EndAddress>, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be altered.

If an invalid hexadecimal value is provided as part of the <StartAddress>, <EndAddress> or <Data> parameters, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be altered.

If the ';nv' parameter is not present on the command line and a target memory location cannot be properly written, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>bf 400 fff 0
S>bf 8000'x 8fff'x 12 34 56
S>
```

4.6 BR - Set/Display User Breakpoints

Command Line Format

BR [<TAddress>..<TAddress>]

Parameter Description

<TAddress> A Local, Global/Linear, Paged or XGate Target address.

Command Description

The BR command is used to set a breakpoint at a specified address or to display any previously set breakpoints. The function of a breakpoint is to halt CPU12(X) or XGate program execution when the program reaches the breakpoint address. When a breakpoint address is encountered, D-Bug12XZ will disassemble the instruction at the breakpoint address, display the CPU12, CPU12X, CPU12Z or XGate's register contents, and wait for the next D-Bug12XZ command to be entered by the user.

Breakpoints are set by entering the breakpoint command followed by one or more breakpoint addresses. Entering the breakpoint command without any breakpoint addresses will display all the currently set breakpoints.

If a breakpoint is set within a page window (i.e. PPAGE, EPAGE, RPAGE) and a page number is not supplied as part of the address, when breakpoints are inserted into the target hardware, the current page value is read from the target and used as the page value for the breakpoint address.

Since D-Bug12XZ uses address only hardware breakpoints, the maximum number of breakpoints that can be set is dependent on the target MCU.

Note: The S12XE family of devices can be configured to map on-chip RAM into the 0x4000 – 0x7fff area in the local memory map, which is normally occupied by PFlash. Global addresses, used by the breakpoint logic, that are associated with the 0x2000 – 0x7fff range in the local map are different depending on whether RAM or PFlash is mapped into the 0x4000 – 0x7fff area.

D-Bug12XZ reads the state of the RAMHM and ROMHM bits in the MMCCTL1 register to determine whether RAM or PFlash is mapped into the 0x4000 – 0x7fff area in order to convert any breakpoint local addresses into correct Global addresses. Therefore, for applications that map RAM into the 0x4000 – 0x7fff area and execute code in the 0x2000 – 0x7fff range with breakpoints set using local addresses, the RAMHM and ROMHM bits should both be written to '1' prior to code execution.

Restrictions

D–Bug12X utilizes the target breakpoint hardware with the instruction fetch tagging. Therefore, this only allows breakpoints to be set on instruction opcodes.

New breakpoints may not be set with the BR command when the 'R>' prompt is being displayed. However, the BR command may be used to display breakpoints that are currently set in the user's running program.

Error Conditions

If an invalid hexadecimal value is provided as part of the <TAddress>, an appropriate error message shall be issued, command execution will be terminated and a breakpoint for that address will not be set.

If an attempt is made to enter a number of breakpoint addresses greater than that supported by the target hardware, command execution will be terminated and a breakpoint(s) for that address will not be set.

If the 'R>' prompt is being displayed and an attempt is made to set a breakpoint, command execution will be terminated and a breakpoint(s) for that address will not be set.

Example

```
S>br 35ec 2f80 c592
Breakpoints: 35ec 2f80 c592

S>br
Breakpoints: 35EC 2F80 C592

S>
```

4.7 BS - Block Search, Search an Address Range For A Data Pattern

Command Line Format

BS <StartAddress> <EndAddress> '<ASCIIString>' | <Data8> [<Data8>]

Parameter Description

<StartAddress>	A Local, Global/Linear, Paged or XGate Target address
<EndAddress>	A Local, Global/Linear, Paged or XGate Target address
<ASCIIString>	An ASCII string consisting of any printable characters EXCEPT the single quote (') character
<Data8>	An 8-bit hexadecimal number

Command Description

The block search command can be used to search an address range for a data pattern. The specified data can be supplied as a quoted ASCII string or up to eight hexadecimal bytes. If the data pattern is found in the specified memory range, the address of the first byte of the data pattern is displayed. When using an ASCII string as the data pattern, only printable ASCII characters excluding the single quote (') character may be used.

Restrictions

The <StartAddress> and <EndAddress> parameters must specify the same memory space. That is, both addresses must be Local, Global/Linear, Paged or XGate target addresses.

The Block Search command cannot be used to search the Flash memory in S12 0.25μ devices when the 'R>' prompt is being displayed.

Error Conditions

If <StartAddress> and <EndAddress> specify different memory spaces, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be altered.

If <StartAddress> is greater than <EndAddress>, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be altered.

If an invalid hexadecimal value is provided as part of the <StartAddress> or <EndAddress> parameters, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be searched.

When supplying an ASCII string as the search data, if a closing single quote is not supplied, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>bs 3e:8000 3f:bfff 'HCS'
Data Found at: $3F:B0AD
S>md 3f:b0ad

3F:B0A0  D6 33 3D F3 - 19 F2 62 F2 - 23 F1 DD 0D - 0A 48 43 53    .3=...b.#....HCS
S>bs c000 ffff 'HCS'
Data Found at: $F0AD
S>
```

4.8 BULK - Bulk Erase on-chip EEPROM or DFlash

Command Line Format

BULK

Parameter Description

No parameters are required

Command Description

The BULK command is used to erase the entire contents of the on-chip EEPROM or DFlash in a single operation. After the bulk erase operation has been performed, each on-chip EEPROM or DFlash location is verified to ensure it is erased .

Restrictions

This command is not valid on devices that do not contain EEPROM or DFlash.

Error Conditions

If the BULK command is entered for a target device not containing EEPROM or DFlash, an appropriate error message shall be issued and command execution will be terminated.

If the target device's EEPROM or DFlash is not properly erased, an appropriate error message shall be issued and command execution will be terminated.

Example

```
>BULK
>
```

4.9 CALL - Execute A Subroutine That Ends With The RTS Instruction

Command Line Format

CALL [<TAddress>]

Parameter Description

<TAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The CALL command is used to execute a subroutine and return to the D-Bug12XZ '`S>`' prompt when the final RTS of the subroutine is executed. After displaying the the '`S>`' prompt, the target CPU register contents will be displayed. All CPU registers contain the values at the time the final RTS instruction was executed with the exception of the program counter (PC). The PC will contain the starting address of the subroutine. If a subroutine address is not supplied on the command line, the current value of the Program Counter (PC) will be used as the starting address.

NOTE: No breakpoints are placed in memory before execution is transferred to user code.

Restrictions

If the called subroutine modifies the value of the stack pointer during its execution, it MUST restore the stack pointer's original value before executing the final RTS of the called subroutine. This restriction is required because D–Bug12X places data on the target stack to return the target CPU to active background when the final RTS of the subroutine is executed. Obviously, any subroutine must obey this restriction to execute properly.

The CALL command cannot be issued when the 'R>' prompt is being displayed indicating that the target system is already running a user program.

Error Conditions

If an invalid hexadecimal value is provided as part of the <TAddress> parameter, an appropriate error message shall be issued, command execution will be terminated.

If an XGate address is provided for the <TAddress> parameter, an appropriate error message shall be issued, command execution will be terminated

If the CALL command is entered when the 'R>' prompt is being displayed, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>call 820
Subroutine Call Returned

 PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0820  0A00  057C  0000     0F:F9        1001 0000
0820  CCFFFF        LDD   #$0FFF

S>
```

4.10 CALLF - Execute A Subroutine That Ends With The RTC Instruction

Command Line Format

CALLF [<TAddress>]

Parameter Description

<TAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The CALLF command is used to execute a 'far' subroutine and return to the D-Bug12XZ 'S>' prompt when the final RTC of the subroutine is executed. After displaying the the 'S>' prompt, the target CPU register contents will be displayed. All CPU registers contain the values at the time the final RTC instruction was executed with the exception of the program counter (PC). The PC will contain the starting address of the subroutine. If a subroutine address is not supplied on the command line, the current value of the Program Counter (PC) will be used as the starting address.

NOTE: No breakpoints are placed in memory before execution is transferred to user code.

Restrictions

If the called subroutine modifies the value of the stack pointer during its execution, it MUST restore the stack pointer's original value before executing the final RTC of the called subroutine. This restriction is required because D–Bug12X places data on the target stack to return the target CPU to active background when the final RTC of the subroutine is executed. Obviously, any subroutine must obey this restriction to execute properly.

The CALLF command cannot be issued when the 'R>' prompt is being displayed indicating that the target system is already running a user program.

Error Conditions

If an invalid hexadecimal value is provided as part of the <TAddress>, an appropriate error message shall be issued, command execution will be terminated and no target memory locations will be searched.

If an XGate address is provided for the <TAddress> parameter, an appropriate error message shall be issued, command execution will be terminated

If the CALLF command is entered when the 'R>' prompt is being displayed, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>callf 30:8006

Subroutine Call Returned

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
30 8006  3FFF  0000  0000     00:00        1101 1000
30:8006  4D0101        BCLR  $0001,#$01

S>
```

4.11 DEVICE - Display The Connected Target MCU Device Type

Command Line Format

DEVICE
DEVICE ?

Parameter Description

? The ASCII '?' character

Command Description

When D-Bug12XZ connects to a target device, it reads the device's PARTID register to identify the target device. The device command can be used to display the target device names and various parameters related to the target device. Because some S12 and S12X devices with different Flash memory sizes share the same die, some target devices may seem to be 'misidentified'. However, a complete list of the devices included in the identified mask set are listed.

Entering the '?' parameter on the command line will display a list of all of the devices supported by D-Bug12XZ.

Restrictions

None.

Error Conditions

If any character(s) appear on the command line other than the ASCII '?', an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>device

Device: MC9S12XEP100, MC9S12XEP768
PFlash: $700000 - $7FFFFF (1048576 bytes)
EEE: $13F000 - $13FFFF (4096 bytes)
DFlash: $100000 - $107FFF (32768 bytes)
RAM: $0F0000 - $0FFFFF (65536 bytes)
I/O Regs: $0000 - $07FF
Target BDM Speed: 5000 KHz

S>device ?

Supported Devices:
```

```
MC9S12XEP100, MC9S12XEP768
MC9S12XEQ512, MC9S12XEQ384, MC9S12XEG384, MC9S12XES384
MC9S12XET256, MC9S12XEG256, MC9S12XEA256, MC9S12XEG128, MC9S12XEA128
MC9S12XS256
MC9S12XS128, MC9S12XS64
MC9S12XHY256, MC9S12XHY128
MC9S12XDP512, MC9S12XDT512, MC9S12XA512, MC9S12XDT384
MC9S12XDT256, MC9S12XDQ256, MC9S12XD256, MC9S12XB256, MC9S12XA256
MC9S12XDG128, MC9S12XD128, MC9S12XA128, MC9S12XB128
MC9S12XHZ512, MC9S12XHZ384, MC9S12XHZ256
MC9S12DP512, MC9S12DT512, MC9S12DJ512, MC9S12A512
MC9S12DP256, MC9S12DT256, MC9S12DJ256, MC9S12DG256
MC9S12DT128, MC9S12DG128, MC9S12DJ128, MC9S12DB128, MC9S12A128
MC9S12DJ64, MC9S12D64, MC9S12A64, MC9S12D32, MC9S12A32
MC9S12B128, MC9S12B64
MC9S12C128, MC9S12C96, MC9S12C64, MC9S12GC128, MC9S12GC96, MC9S12GC64
MC9S12C32, MC9S12C16, MC9S12GC32, MC9S12GC16
MC9S12HZ256, MC9S12HZ128
MC9S12P128, MC9S12P96, MC9S12P64, MC9S12P32
MC9S12G128, MC9S12G96
MC9S12GN32, MC9S12GN16

S>
```

4.12 EEE – Enable, Disable or Query Emulated EEPROM

Command Line Format

EEE ENABLE | DISABLE | QUERY

Parameter Description

ENABLE The ASCII characters 'ENABLE' in upper, lower or mixed case.
DISABLE The ASCII characters 'DISABLE' in upper, lower or mixed case.
QUERY The ASCII characters 'QUERY' in upper, lower or mixed case.

Command Description

The EEE command is only valid for those devices containing Emulated EEPROM, currently the S12XE family. It allows the EEE to be enabled, disabled or queried from the command line rather than having to execute those commands by writing to the FTM registers.

Restrictions

The EEE command is only valid if D-Bug12 is connected to a device containing EEE.

Error Conditions

If any parameters made up of ASCII characters other than those listed in the Parameter Description section, an appropriate error message shall be issued and command execution will be terminated.

If the EEE cannot be enabled, disabled or queried, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>eee enable
S>eee query
DFPART: 0 (0 Bytes)
ERPART: 8 (2048 Bytes)
Erased Sector Count: 1
Dead Sector Count: 0
Ready Sector Count: 4
S>
```

4.13 FBULK - Erase target on-chip Program Flash Memory

Command Line Format

FBULK

Parameter Description

None.

Command Description

The FBULK command is used to erase the entire contents of the on-chip Program Flash. After the bulk erase operation is performed, the on-chip Program Flash is checked to verify it is erased. After erasure, the security byte is programmed with a value of 0xfe to place the target in an unsecured state. Before erasing the Program Flash, the target device is reset so that the target is placed in a known state and the Flash state machine can be properly configured.

Restrictions

None.

Error Conditions

If the Flash can not be erased, an appropriate error message shall be issued and command execution will be terminated.

If the security byte cannot be programmed to a value of 0xfe, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>fbulk
S>
```

4.14 FLOAD - Program on-chip PFlash, DFlash or EEPROM from S-Records

Command Line Format

FLOAD [<AddressOffset> | ;b]

Parameter Description

<AddressOffset>	A 32-bit hexadecimal number
;b	The ASCII string ';b' or ';B'

Command Description

The FLOAD command is used to program a target device's PFlash, DFlash or EEPROM memory with the data contained in S-Record object files. The address offset, if supplied, is added to the load address of each S-Record before an S-Record's data bytes are programmed into memory. Providing an address offset other than zero allows object code or data to be programmed into memory at a location other than the address for which it was assembled or compiled.

The time required to program target on-chip Flash memory varies with the different Flash memory technologies used on the M68HC12 family. Because of this variability, D–Bug12X uses XOn/XOff handshaking to control the flow of S-Record data. As each S-Record is received and processed, an ASCII asterisk character (*) is sent to the screen to indicate programming progress.

The FLOAD command is terminated when D-Bug12 receives an'S7', 'S8' or 'S9' logical end of file S-Record. If the object file being loaded does not contain a logical end of file S-Record, D–Bug12X will not return its prompt and will continue to wait for the end of file record. Entering an ASCII Control-C (0x03) or ASCII Escape character (0x1b) will return D–Bug12X to its command line prompt.

The ;b option, which is mutually exclusive with the address offset option, is used to allow loading of 'banked' or 'paged' S-Records produced by some compilers and assemblers.

If an error occurs during the execution of the FLOAD command, an error message is displayed, but D-Bug12XZ will not return to its command prompt. Instead, D-Bug12XZ waits in a loop receiving characters from the host allowing the operator to cancel the file transmission. Within this loop, the firmware waits for 0.5 second after the last character is received before returning to the D-Bug12XZ command prompt.

Restrictions

The FLOAD command cannot be used with S-Records containing a code/data field longer than 64 bytes. Sending an S-Record with a code/data field longer than 64 bytes will cause D-Bug12XZ to terminate the FLOAD command and issue an error message.

When connected to an S12 or S12X 0.25μ family device, each PFlash or EEPROM S-Record must have an even load address and contain an even number of bytes. This restriction is due to the programming granularity of one word (two bytes) on an even byte boundary.

When connected to an S12 or S12X 0.18μ family device, each PFlash S-Record must have a load address that is evenly divisible by eight and have a code/data field containing a number of bytes that is evenly divisible by eight. This restriction is due to the programming granularity of eight bytes (a phrase) on an eight byte boundary. For DFlash or EEPROM, each S-Record must have an even load address and contain an even number of bytes.

For all S12 and S12X devices, the file used for programming may not contain an S-Record with a value for the security byte since the security byte is programmed to a value of 0xfe by the FBULK command.

Error Conditions

If an S-Record containing a code/data field longer than 64 bytes is received, an appropriate error message shall be issued and command execution will be terminated.

For S12 or S12X 0.25μ family devices, if an S-Record is received containing an odd load address or an odd number of bytes in the code/data field, an appropriate error message shall be issued and command execution will be terminated.

For S12 or S12X 0.18μ family devices, if an S-Record is received containing a load address not evenly divisible by eight or the code/data field is not a multiple of eight bytes, an appropriate error message shall be issued and command execution will be terminated.

If the received S-Record checksum value does not match the value calculates by the firmware, an appropriate error message shall be issued and command execution will be terminated.

If an S-Record is received with a load address not corresponding to the target devices PFlash, DFlash or EEPROM, an appropriate error message shall be issued and command execution will be terminated.

If any PFlash, DFlash or EEPROM location cannot be programmed, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>fload
************************************************************************
************************************************************************
**********************************
S>
```

4.15 Go, begin execution of application code

Command Line Format

G [<TAddress>]

Parameter Description

<TAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The G command is used to begin or continue the execution of application code in real time. Before beginning execution of application code, any breakpoints set using the BR command are placed in memory. Execution of the application program will continue until a breakpoint is encountered or an exception occurs that causes the target MCU to enter active background. When application code halts for one of these reasons and control is returned to D-Bug12X, a message is displayed explaining the reason for program termination. In addition, D-Bug12XZ displays the target CPU's register contents, disassembles the instruction at the current PC address, and waits for the next D-Bug12XZ command to be entered.

If a starting address is not supplied on the command line, program execution will begin at the address contained in the Program Counter.

Restrictions

The G command cannot be issued when the 'R>' prompt is being displayed indicating that the target system is already running a user program.

Error Conditions

If the Go command is entered when the 'R>' prompt is being displayed, an appropriate error message shall be issued and command execution will be terminated.

If an invalid target address is supplied for the <TAddress> parameter, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>g 800
R>
User Breakpoint Encountered

 PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0820  09FE  057C  0000     00:00        1001 0100
0820  08            INX
S>
```

4.16 GT - Go Until, Execute application code until temporary breakpoint reached

Command Line Format

GT <TAddress>

Parameter Description

<TAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The GT command is similar to the G command except that a temporary breakpoint is placed at the target address supplied on the command line. Any breakpoints set by the BR command are NOT placed in the application code before program execution begins. Program execution begins at the address defined by the current value of the Program Counter. When application code reaches the temporary breakpoint, control is returned to D-Bug12XZ and a message is displayed explaining the reason for user program termination. In addition, D-Bug12XZ displays the target CPU's register contents, disassembles the instruction at the current PC and waits for the next D-Bug12XZ command to be entered by the user.

Restrictions

The GT command cannot be issued when the 'R>' prompt is being displayed indicating that the target system is already running a user program.

Error Conditions

If the GT command is entered when the 'R>' prompt is being displayed, an appropriate error message shall be issued and command execution will be terminated.

If an invalid target address is supplied for the <TAddress> parameter, an appropriate error message shall be issued and command execution will be terminated.

Example

```
S>gt 820
R>
Temporary Breakpoint Encountered

 PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0820  09FE  057C  0000     00:00        1001 0100
0820  08            INX
S>
```

4.17 HELP - Display D-Bug12 command summary

Command Line Format

HELP

Parameter Description

None.

Command Description

The HELP command is used to display a summary of the D-Bug12XZ command set. Each command is shown with its command line format and a brief description of the command's function. The commands are listed in alphabetical order.

Restrictions

None.

Error Conditions

None.

Example

```
S>help
ASM <TAddress>  Single line assembler/disassembler
    <CR>       Disassemble next instruction
    <.>        Exit assembly/disassembly
BAUD <baudrate> | AUTOON | AUTOOFF  Set communications rate for the terminal
BDMDB  Enter the BDM command debugger
BF <TStartAddress> <TEndAddress> [<data>..<data>] [;nv]  Fill memory with data
BR [<TAddress>]  Set/Display breakpoints
BS <StartAddress> <EndAddress> '<String>' | <Data8> [<Data8>]  Block Search
BULK  Erase on-chip EEPROM or DFlash contents
CALL [<TAddress>]  Call subroutine at <TAddress> that ends with RTS
CALLF [<TAddress>]  Call Far subroutine at <TAddress> that ends with RTC
DEVICE [?]  display target device or list of supported devices
EEE ENABLE | DISABLE | QUERY  Enable, Disable or Query EEE
FBULK  Erase entire target (P)FLASH contents
FLOAD [<AddressOffset> | ;b]  Load S-Records into target Flash, DFlash or EEPROM
G [<TAddress>]  Begin/continue execution of user code
GT <TAddress>  Set temporary breakpoint at <Address> & execute user code
HELP  Display D-Bug12 command summary
LOAD [<AddressOffset>]  Load S-Records into RAM
LOG [[<TAddress> <NumBytes> <mSInterval>] | START | CLEAR]  Log Data to Terminal
MD FLASH | DFLASH | EE | EEE | IO | RAM  Memory Display Bytes
```

```
   <StartAddress> [<EndAddress>]  Memory Display Bytes
MDW FLASH | DFLASH | EE | EEE | IO | RAM  Memory Display Words
    <TStartAddress> [<TEndAddress>]  Memory Display Words
MM <StartAddress>   Modify Memory Bytes
   <CR>             Examine/Modify next location
   </> or <=>       Examine/Modify same location
   <^> or <->       Examine/Modify previous location
   <.>              Exit Modify Memory command
MMW <TStartAddress>  Modify Memory Words (same subcommands as MM)
MOVE <TStartAddress> <EndAddress> <DestAddress> Move a block of memory
NOBR [<TAddress>]  Remove One/All Breakpoint(s)
PARTDF [<DFlashBytes> <EEEBytes>]  Display/Set EEE/DFlash Partition
RD  Display CPU registers
RESET  Reset target CPU
RM  Modify CPU Register Contents
SECURE [<SecByteVal>]  Secure targt device (default <SecByteVal> is 0xfd)
STOP  Stop target CPU
T [<count>]  Trace <count> instructions
TO Trace Over subroutine calls
UPLOAD FLASH | DFLASH | EE | EEE | IO | RAM [;b] [;<SRecSize>]  S-Record Memory
display
        <TStartAddress> <TEndAddress> [;b] [;<SRecSize>]  S-Record Memory display
VER Display D-Bug12's Version Number
VERF [<AddressOffset>]  Verify S-Records against memory contents
XASM <TAddress>  XGate single line assembler/disassembler
     <CR>       Disassemble next instruction
     <.>        Exit assembly/disassembly
XDBG ENTER [<Channel>] | EXIT  Enter/Exit XGate thread debug
XG  Continue execution of XGate thread
XRD  Display XGate CPU registers
XT [<count>]  Trace <count> XGate instructions
<Register Name> <Register Value>  Set register contents
       Register Names: PC, SP, X, Y, A, B, D, PP EP RP GP
       CCR Status Bits: S, XM, H, IM, N, Z, V, C IPL
<XGRegister Name> <XGRegister Value>  Set XGate register contents
       Register Names: XPC, XCCR, R1 - R7, XVBR, CHID, MCTL
R>
```

4.18 LOAD - Load S-Records Into Target RAM, EEPROM or EEE

Command Line Format

LOAD [<AddressOffset>]

Parameter Description

<AddressOffset> A 32-bit hexadecimal number

Command Description

The Load command is used to load S-Record object files into target RAM. The address offset, if supplied, is added to the load address of each S-Record before an S-Record's data bytes are placed in memory. Providing an address offset other than zero allows object code or data to be loaded into memory at a location other than that for which it was assembled.

During the loading process, the S-Record data is not echoed to the screen. However, for each S-Record that is successfully loaded, an ASCII asterisk character (*) is displayed. When an S-Record file has been successfully loaded, D-Bug12XZ will display its prompt.

The LOAD command is terminated when D-Bug12XZ receives an'S7', 'S8' or 'S9' logical end of file S-Record. If the object file being loaded does not contain a logical end of file S-Record, D–Bug12X will not return its prompt and will continue to wait for the end of file record. Entering an ASCII Control-C (0x03) or ASCII Escape character (0x1b) will return D–Bug12X to its command line prompt.

Restrictions

The LOAD command can only be used to load S-Records into RAM, EEPROM or EEE.

When the LOAD command is used to load data into the on-chip EEPROM of a target device, the memory locations defined by the load address of any single S-Record plus the number of bytes in that S-Record's data field must correspond to target addresses that lie entirely within the EEPROM array.

Error Conditions

If an invalid hexadecimal number is provided for the <AddressOffset> parameter, an appropriate error message shall be issued and command execution will be terminated.

If any non-hexadecimal characters appear in the S-Record, Length, Load address, Code/Data or Checksum fields, an appropriate error message shall be issued and command execution will be terminated.

If the received S-Record checksum value does not match the value calculates by the firmware, an appropriate error message shall be issued and command execution will be terminated.

Example

```
>load 1000
********************
>
```

4.19 LOG - Display target memory at selected locations and time intervals

Command Line Format

LOG [<LogAddress> <Bytes> <mSInterval> | START [;VT] [;BC] | CLEAR]

Parameter Description

<LogAddress>	A Local, Global/Linear, Paged or XGate CPU12/CPU12X address.
<Bytes>	A decimal number in the range 1 ≤ <Bytes> ≤ 250 indicating the number of bytes to display beginning at <LogAddress>
<mSInterval>	A 16 bit unsigned decimal number indicating the rate, in mS, at which the data at <LogAddress> is displayed.
START	The ASCII characters "START" in upper, lower or mixed case.
CLEAR	The ASCII characters "CLEAR" in upper, lower or mixed case.
;VT	The ASCII characters ";VT" in upper, lower or mixed case.
;BC	The ASCII characters ";BC" in upper, lower or mixed case.

Command Description

The LOG command can be used to display memory locations of a connected target device at selected time intervals. Entering the LOG command with no parameters will display a table containing the start address (<LogAddress>), number of bytes (<Bytes >) and the time interval (<mSInterval >) at which the data at <LogAddress> will be displayed.

Placing entries in the log table may be accomplished by supplying the LogAddress, the number of bytes and the time interval as a command line parameter as shown above in the **Command Line Format** section.

While the minimum interval allowed for the display of data is 1 mS, the communication rate between the Programmer and the PC and the BDM communication rate will ultimately restrict the maximum rate at which data can be displayed.

Entering the LOG command followed by the START parameter will begin displaying data on the screen. The data for each <LogAddress> in the table is displayed with the address followed by the associated bytes of data. The ASCII dash character ("-") is used to separate the address from the data. Each line is terminated with the ASCII carriage return (0x0d) and line feed (0x0a).

Logging of the data at the specified locations will continue indefinately until an ASCII Control-C (0x03) or ASCII Escape (0x1b) character is received by the Programmer.

If the optional '`;VT`' parameter is included on the command line with the START parameter, the Programmer will use VT100 terminal commands to control cursor placement so that each <LogAddress> and the associated data is displayed on its own line. Each time the data is

displayed, the previous data is overwritten. Immediately after entering the LOG command, the cursor will be placed at the 'home' position (column 1, row 1) and the terminal screen will be erased.

After a target device is reset, the BDM module's communication rate is based on the oscclk ÷ 2. If the optional '`;BC`' parameter is included on the command line with the START parameter, the Programmer will instead use the target MCU's bus clock for the BDM module. This can signifigantly increase the amount of data that can be read from the target MCU in a given time period.

Note: Target MCUs whose applications vary the bus speed cannot use the ;BC option. Doing so will result in a loss of communication with the target device and termination of the LOG command.

Note: The ;BC option can not be used with some of the early S12 devices that do not incorporate the Enhanced BDM Module. It also cannot be used with any of the S12 0.18μ devices because these devices do not allow the MCU bus clock to be used as the source of the BDM module clock.

Entering the LOG command followed by the CLEAR parameter will clear all the entries in the Programmer's log table.

Restrictions

A maximum of 16 log table entries is allowed.

The number of data locations that can be displayed and the rate at which the data can be displayed will be determined by the communication rate between the Programmer and the PC and the BDM communication rate.

Error Conditions

When entering a log table entry on the command line, if <LogAddress> is an invalid hexadecimal number, an error message will be issued and command execution will be terminated.

When entering a log table entry on the command line, if <Bytes> is an invalid decimal number or has a value greater than 250, an error message will be issued and command execution will be terminated.

When entering a log table entry on the command line, if <mSInterval> is an invalid decimal

number, an error message will be issued and command execution will be terminated.

Example

```
S> log

Address    Bytes Interval (mS)
--------   ----- -------------
001000      4      1000
001004      2      500

S>log start
001004:FFF7
001000:FB77DFDF
001004:FFF7
001004:FFF7
001000:FB77DFDF
001004:FFF7
001004:FFF7
001000:FB77DFDF
001004:FFF7
001004:FFF7
S>
```

4.20 MD - Display memory in hexadecimal bytes and ASCII format

Command Line Format

MD <StartAddress> [<EndAddress>] | FLASH | DFLASH | EE | EEE | IO | RAM

Parameter Description

<StartAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<EndAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
FLASH	The ASCII characters “FLASH” in upper, lower or mixed case.
DFLASH	The ASCII characters “DFLASH” in upper, lower or mixed case.
EE	The ASCII characters “EE” in upper, lower or mixed case.
EEE	The ASCII characters “EEE” in upper, lower or mixed case.
IO	The ASCII characters “IO” in upper, lower or mixed case.
RAM	The ASCII characters “RAM” in upper, lower or mixed case.

Command Description

The memory display command displays the contents of memory in both hexadecimal bytes and ASCII, 16-bytes on each line. The <StartAddress> parameter must be supplied, however, the <EndAddress> parameter is optional. When the <EndAddress> parameter is not supplied, a single line is displayed.

The number supplied as the <StartAddress> parameter is rounded down to the next lower multiple of 16. While the number supplied as the <EndAddress> parameter is rounded up to the next higher multiple of 16 - 1. This causes each line to display memory in the range of $xxx0 through $xxxF. For example if the user entered $205 as the start address and $217 as the ending address, the actual memory range displayed would be $200 through $21F.

The parameters ‘FLASH’, ‘DFLASH’, ‘EE’, ‘EEE’, ‘IO’ or ‘RAM’can be used to display the contents of the associated on-chip memory resource without having to enter its start and end address. This feature is particularly helpful when performing a ‘hot-connect’ to an unresponsive target MCU since it allows some of the critical memory resources to be displayed quickly. This can be particularly advantages for devices in the 0.25μ S12 family where the RAM, EEPROM and I/O registers may be relocated in the 64K local memory map by the running application. Because the RAM, EEPROM and I/O registers can overlap, only the visible portion of the associated resource will be displayed when using one of the command line mnemonic parameters.

Entering an ASCII Escape (0x1b) or control C (0x03) character during the display of target memory data will cause D-Bug12XZ to return to the command prompt.

Restrictions

The address space defined by the <StartAddress> and <EndAddress> must all be in the same address space. That is, both addresses must be a Local or Global/Linear or Paged CPU12/CPU12X or XGate address.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <StartAddress> and <EndAddress> parameters while the 'R>' prompt is displayed.

Example

```
S>md 800

0800  AA 04 37 6A - 00 06 27 F9 - 35 AE 78 0D - B7 56 78 20   ..7j..'.5.x..Vx
S>md 800 87f

0800  AA 04 37 6A - 00 06 27 F9 - 35 AE 78 0D - B7 56 78 20   ..7j..'.5.x..Vx
0810  B6 36 27 F9 - 35 AE 27 F9 - 35 9E 27 F9 - 35 BE B5 28   .6'.5.'.5.'.5..(
0820  27 F9 35 D6 - 37 B8 00 0F - 37 82 01 0A - 37 36 FF F0   '.5.7...7...76..
0830  7C 10 37 B3 - 00 00 37 B6 - 00 0F AA 04 - A5 02 37 B6   |.7...7.......7.
0840  00 0F 27 78 - 37 6A 00 06 - 27 F9 35 78 - 27 F9 35 56   ..'x7j..'.5x'.5V
0850  78 0D B7 10 - 78 3B 37 86 - 00 DC 27 F9 - 35 48 78 57   x...x;7...'.5HxW
0860  37 86 00 DE - F5 01 EA 09 - 37 B5 0D 0A - 27 F9 36 2A   7.......7...'.6*
0870  A5 00 37 65 - 00 02 27 F9 - 35 E8 37 9C - 37 4C F5 02   ..7e..'.5.7.7L..
S>md io

0000  00 00 00 00 - 00 00 00 00 - 9F 00 00 80 - D0 00 00 00  ................
0010  00 00 00 00 - 00 0E 00 00 - 00 00 39 80 - C0 00 40 00  ..........9...@.
0020  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
0030  00 00 00 00 - 04 01 03 7B - 00 00 00 00 - 00 00 00 00  .......{........
0040  01 00 00 00 - D5 00 A0 00 - 00 00 00 00 - 01 02 00 80  ................
0050  4E 09 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  N...............
0060  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
0070  0F 2F 00 20 - 05 00 00 00 - 00 00 00 00 - 00 00 00 00  ./. ............
.
.
.
03A0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
03B0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
03C0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
03D0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
03E0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
03F0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
S>
```

4.21 MDW - Display memory in hexadecimal words and ASCII format

Command Line Format

MDW <StartAddress> [<EndAddress>] | FLASH | DFLASH | EE | EEE | IO | RAM

Parameter Description

<StartAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<EndAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
FLASH	The ASCII characters “FLASH” in upper, lower or mixed case.
DFLASH	The ASCII characters “DFLASH” in upper, lower or mixed case.
EE	The ASCII characters “EE” in upper, lower or mixed case.
EEE	The ASCII characters “EEE” in upper, lower or mixed case.
IO	The ASCII characters “IO” in upper, lower or mixed case.
RAM	The ASCII characters “RAM” in upper, lower or mixed case.

Command Description

The memory display command displays the contents of memory in both hexadecimal words and ASCII, 16-bytes on each line. The <StartAddress> parameter must be supplied, however, the <EndAddress> parameter is optional. When the <EndAddress> parameter is not supplied, a single line is displayed.

The number supplied as the <StartAddress> parameter is rounded down to the next lower multiple of 16. While the number supplied as the <EndAddress> parameter is rounded up to the next higher multiple of 16 - 1. This causes each line to display memory in the range of $xxx0 through $xxxF. For example if the user entered $205 as the start address and $217 as the ending address, the actual memory range displayed would be $200 through $21F.

The parameters ‘FLASH’, ‘DFLASH’, ‘EE’, ‘EEE’, ‘IO’ or ‘RAM’can be used to display the contents of the associated on-chip memory resource without having to enter its start and end address. This feature is particularly helpful when performing a ‘hot-connect’ to an unresponsive target MCU since it allows some of the critical memory resources to be displayed quickly. This can be particularly advantages for devices in the 0.25μ S12 family where the RAM, EEPROM and I/O registers may be relocated in the 64K local memory map by the running application. Because the RAM, EEPROM and I/O registers can overlap, only the visible portion of the associated resource will be displayed when using one of the command line mnemonic parameters.

Entering an ASCII Escape (0x1b) or control C (0x03) character during the display of target memory data will cause D-Bug12XZ to return to the command prompt.

Restrictions

The address space defined by the <StartAddress> and <EndAddress> must all be in the same address space. That is, both addresses must be a Local or Global/Linear or Paged CPU12/CPU12X or XGate address.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <StartAddress> and <EndAddress> parameters while the 'R>' prompt is displayed.

Example

```
S>mdw 800

0800  AA04 376A - 0006 27F9 - 35AE 780D - B756 7820   ..7j..'.5.x..Vx
S>mdw 800 87f

0800  AA04 376A - 0006 27F9 - 35AE 780D - B756 7820   ..7j..'.5.x..Vx
0810  B636 27F9 - 35AE 27F9 - 359E 27F9 - 35BE B528   .6'.5.'.5.'.5..(
0820  27F9 35D6 - 37B8 000F - 3782 010A - 3736 FFF0   '.5.7...7...76..
0830  7C10 37B3 - 0000 37B6 - 000F AA04 - A502 37B6   |.7...7.......7.
0840  000F 2778 - 376A 0006 - 27F9 3578 - 27F9 3556   ..'x7j..'.5x'.5V
0850  780D B710 - 783B 3786 - 00DC 27F9 - 3548 7857   x...x;7...'.5HxW
0860  3786 00DE - F501 EA09 - 37B5 0D0A - 27F9 362A   7.......7...'.6*
0870  A500 3765 - 0002 27F9 - 35E8 379C - 374C F502   ..7e..'.5.7.7L..
S>mdw ram

2800  FFBF FDEF - FDE3 BEB7 - E97F E5BA - 4FF7 DB5F  ............O.._
2810  EF7F B78B - F77F F357 - AE7B ECC9 - DCD3 E231  .......W.{.....1
2820  5904 0062 - 700D 0080 - 8A49 C308 - 8482 028A  Y..bp....I......
2830  213A C04C - 2030 9202 - 7965 1224 - A090 D22A  !:.L 0..ye.$...*
2840  F9FB AFDD - B6C7 71BF - F75B F6FD - DAFF 3F7F  ......q..[....?.
2850  7FD7 FDF3 - AACE FFFF - 6FF5 D5A7 - EEB7 B97B  ........o......{
2860  A154 2783 - 10A7 4066 - 8378 41C0 - 80E5 1004  .T'...@f.xA.....
2870  0311 9047 - 0B25 580A - 0D23 B065 - 2324 9002  ...G.%X..#.e#$..
2880  7DBB D7D9 - FF77 E9FF - F6AE D7DD - 33FF CBF9  }....w......3...
2890  E6FF B7BE - 7B9F D1FC - F7CF DFF6 - ADFC 55F8  ....{.........U.
.
.
.
3F60  80A3 8122 - 0A56 603D - 8C98 802E - 8E44 9424  ...".V`=.....D.$
3F70  C2E3 AE02 - 8844 3625 - 40C1 0A96 - 418C 3427  .....D6%@...A.4'
3F80  BFFF DBFC - 9EFF F27D - 7EFF A37E - ECFF FF2F  .......}~..~.../
3F90  3FF3 99DF - CACF B759 - 3BCA 99BE - CE9F F5FF  ?......Y;.......
3FA0  A23D 46B0 - BE18 B506 - 9810 43DD - 2A20 2009  .=F.......C.*  .
3FB0  0C10 02C1 - A200 8CAA - 0342 649A - AC82 9030  .........Bd....0
3FC0  E7BF AAF7 - FFFE A3F0 - E3BB EDB5 - AF36 E160  .............6.`
3FD0  5BE4 7FD1 - DA97 DDD9 - 9BFD 7F9E - D6DC D75F  [..............._
3FE0  C006 3611 - 0491 4294 - 2215 2131 - 000A 9028  ..6...B.".!1...(
3FF0  A901 6031 - 1436 A020 - 4634 0421 - 1886 CC4A  ..`1.6. F4.!...J
S>
```

4.22 MM - Modify memory bytes in hexadecimal format

Command Line Format

MM <TAddress> [<data>..<data>]

Parameter Description

<TAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<data>	An 8-bit hexadecimal number.

Command Description

The memory modify command allows the contents of memory to be examined and/or modified as 8-bit hexadecimal data. If one or more 8-bit data parameters is present on the command line, the byte(s) beginning at memory location at <TAddress> will be written with the supplied data. Up to eight bytes of data may be supplied. If no <data> parameters are present on the command line, D-Bug12XZ will enter the interactive memory modify mode. In the interactive mode, each byte is displayed on a separate line following the data's address. Once the memory modify command has been entered, several sub-commands are used for the modification and verification of memory contents. These sub-commands have the following format:

[<Data>]<CR>	Optionally update current location and display the next location
[<Data>] / or =	Optionally update current location and redisplay the current location
[<Data>] ^ or -	Optionally update current location and display the previous location
[<Data>] .	Optionally update current location and exit Memory Modify

With the exception of the carriage return, the sub-command must be separated from any entered data with at least one space character. If an invalid sub-command character is entered, an appropriate error message will be issued and the contents of the current memory location shall be redisplayed.

Restrictions

While there are no restrictions regarding the use of the MM command, caution should be used when modifying target memory while user code is running. Accidentally modifying target memory containing program code could lead to program run away.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <TAddress> parameter while the 'R>' prompt is displayed.

When the MM command is used to write data to the on-chip EEPROM of a target device, the memory locations defined by <TAddress> plus the number of bytes written must correspond

to target addresses that lie entirely within the EEPROM array.

Example

```
R>mm 800
0800 00 <CR>
0801 F0 FF
0802 00 ^
0801 FF <CR>
0802 00 <CR>
0803 08 55 /
0803 55 .
R>
```

4.23 MMW - Modify memory bytes in hexadecimal format

Command Line Format

MMW <TAddress> [<data>..<data>]

Parameter Description

<TAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<data>	A 16-bit hexadecimal number.

Command Description

The memory modify word command allows the contents of memory to be examined and/or modified as 16-bit hexadecimal data. If one or more 16-bit data parameters is present on the command line, the word(s) beginning at memory location at <TAddress> will be written with the supplied data. Up to eight words of data may be supplied. If no <data> parameters are present on the command line, D-Bug12XZ will enter the interactive memory modify mode. In the interactive mode, each word is displayed on a separate line following the data's address. Once the memory modify command has been entered, several sub-commands are used for the modification and verification of memory contents. These sub-commands have the following format:

[<Data>]<CR>	Optionally update current location and display the next location
[<Data>] / or =	Optionally update current location and redisplay the current location
[<Data>] ^ or -	Optionally update current location and display the previous location
[<Data>] .	Optionally update current location and exit Memory Modify

With the exception of the carriage return, the sub-command must be separated from any entered data with at least one space character. If an invalid sub-command character is entered, an appropriate error message will be issued and the contents of the current memory location shall be redisplayed.

Restrictions

While there are no restrictions regarding the use of the MMW command, caution should be used when modifying target memory while user code is running. Accidentally modifying target memory containing program code could lead to program run away.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <TAddress> parameter while the 'R>' prompt is displayed.

When the MMW command is used to write data to the on-chip EEPROM of a target device, the memory locations defined by <TAddress> plus the number of bytes written must correspond to target addresses that lie entirely within the EEPROM array.

Example

```
S>mmw 800
0800 00F0 <CR>
0802 0008 AA55<CR>
0804 843F ^
0802 AA55 <CR>
0804 843F <CR>
0806 C000 .
S>
```

4.24 MOVE - Move a Block of Memory

Command Line Format

MOVE <StartAddress> <EndAddress> <DestAddress>

Parameter Description

<StartAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<EndAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<DestAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The MOVE command is used to move a block of memory from one location to another one byte at a time. The number of bytes moved is one more than the <EndAddress> - <StartAddress>. The block of memory created beginning at the destination address may overlap the memory block defined by the <StartAddress> and <EndAddress>.

Restrictions

A minimum of one byte may be moved if the <StartAddress> is equal to the <EndAddress>.

Caution should be exercised when moving target memory while user code is running. Accidentally modifying target memory containing program code could lead to program run away.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <StartAddress>, <EndAddress> and <DestAddress> parameters while the 'R>' prompt is displayed.

The address space defined by the <StartAddress>, <EndAddress> and <DestAddress> must all be in the same address space. That is, all three addresses must be a Local or Global/Linear or Paged CPU12/CPU12X or XGate address.

When the MOVE command is used to copy data into the on-chip EEPROM of a target device, the memory locations beginning at <DestAddress> plus the number of bytes defined by <EndAddress> - <StartAddress> + 1 must correspond to target addresses that lie entirely within the EEPROM array.

Example

```
S>move 800 8ff 1000
S>
```

4.25 NOBR - Remove one/all user breakpoints

Command Line Format

NOBR [<TAddress>…]

Parameter Description

<TAddress> A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.

Command Description

The NOBR command is used to remove one or more of previously entered breakpoints. If the NOBR command is entered without any arguments, all user breakpoints are removed from the breakpoint table.

Restrictions

Breakpoints may not be removed with the NOBR command when the 'R>' prompt is being displayed.

Example

```
>br 800 810 820 830
Breakpoints: 0800  0810  0820  0830

>nobr 810 820
Breakpoints: 0800  0830

>nobr
All Breakpoints Removed

>
```

4.26 PARTDF - Display/Set EEE/DFlash Partition

Command Line Format

PARTDF [<DFPART> <ERPART>]

Parameter Description

<DFPART> A 16-bit decimal number representing the Size of the DFlash partition in bytes.

<ERPART> A 16-bit decimal number representing the Size of the EEE partition in bytes.

Command Description

On S12X devices containing Emulated EEPROM (EEE), a portion or all of the DFlash can be used for the EEE. Before using the EEE, the DFlash must be partitioned and formatted for EEE use. The PARTDF command may be used to partition and format the DFlash in a connected target MCU directly from the command line. Both command line parameters must be an even multiple of 256 and meet the following restrictions:

DFPART ≤ DFlashSize
ERPART ≤ BuffRAMSize
If ERPART > 0, (DFlashSize – DFPART) ÷ 256 ≥ 12 (min. number sectors to support EEE).
If ERPART > 0, (DFlashSize – DFPART) ÷ ERPART ≥ 8 (min. ratio of DFlash to EEE).

For example, an S12XEP100 has 32768 bytes of DFlash and 4096 bytes of buffer RAM that can be used for EEE. If the following command is entered:

PARTDF 8192 1536

Of the 32768 bytes of DFlash, 8192 bytes of the DFlash would be allocated for application use and 24576 bytes allocated for EEE use. This would easily meet the requirement of having a minimum of 12 sectors allocated for EEE use. Of the 4096 bytes of buffer RAM, 1536 bytes would be allocated for EEE and 2560 bytes would be available as volatile RAM storage for the application. This allocation also meets the requirement of having a minimum ratio of 8:1 of DFlash available for EEE.

Note that either partition may have a size of zero. If an application does not require the direct use of DFlash but requires the maximum amount of EEE, the <DFPART> parameter may have a value of zero.

Entering the PARTDF command without the <DFPART> and <ERPART> parameters will display the current partitioning of the connected target.

Restrictions

This command is only valid when an S12XE target device.

Error Conditions

If <DFPART> or <ERPART> is an invalid decimal number, an appropriate error message will be issued and command execution will be terminated.

If the current target device does not contain Emulated EEPROM, an appropriate error message will be issued and command execution will be terminated.

If <DFPART> and <ERPART> are not an even multiple of 256, an appropriate error message will be issued and command execution will be terminated.

If the target MCU fails to properly partition the EEE/DFlash, an appropriate error message will be issued and command execution will be terminated.

If <DFPART> or <ERPART> do not meet the following restrictions:

DFPART ≤ DFlashSize
ERPART ≤ BuffRAMSize
If ERPART > 0, (DFlashSize – DFPART) ÷ 256 ≥ 12.
If ERPART > 0, (DFlashSize – DFPART) ÷ ERPART ≥ 8.

an appropriate error message will be issued and command execution will be terminated.

If at any point during command execution, BDM communication cannot be established with the target MCU, an appropriate error message will be issued and command execution will be terminated.

Example

```
S>partdf 8192 1536
S>
```

4.27 RD - Display CPU12/CPU12X Register Contents

Command Line Format

RD

Parameter Description

No parameters are required. Any parameters on the command line will be ignored.

Command Description

The Register Display command is used to display the CPU12/CPU12X's registers.

Restrictions

The CPU registers may not be displayed when the 'R>' prompt is being displayed.

Example

```
S>device

Device: MC9S12P128, MC9S12P96, MC9S12P64, MC9S12P32
PFlash: $020000 - $03FFFF (131072 bytes)
DFlash: $004400 - $0053FF (4096 bytes)
RAM: $2800 - $3FFF (6144 bytes)
I/O Regs: $0000 - $03FF
Target BDM Speed: 8000 KHz

S>rd

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C029  4000  0000  0000     00:00        1101 1000
xx:C029  CF3100           LDS   #$3100

S>device

Device: MC9S12XEP100, MC9S12XEP768
PFlash: $700000 - $7FFFFF (1048576 bytes)
EEE: $13F000 - $13FFFF (4096 bytes)
DFlash: $100000 - $107FFF (32768 bytes)
RAM: $0F0000 - $0FFFFF (65536 bytes)
I/O Regs: $0000 - $07FF
Target BDM Speed: 4000 KHz

S>rd

PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
FE 524D  0000  0000  0000     00:00         0  1101 1000  00 FD FE
xx:524D  6B59             STAB  -7,Y
S>
```

4.28 RESET - Reset the target system MCU

Command Line Format

RESET

Parameter Description

No parameters are required. Any parameters on the command line will be ignored.

Command Description

The RESET command is used to reset the target system MCU and place it in active background mode. The program counter is initialized with the contents of the target MCU's reset vector, memory locations $FFFE and $FFFF. The stack pointer is initialized with the highest RAM address visible in the local memory map.

Restrictions

None.

Example

```
S>reset
Target CPU Has Been Reset
S>rd

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C029  4000  0000  0000     00:00        1101 1000
xx:C029  CF3100           LDS   #$3100
S>g
R>reset
Target CPU Has Been Reset
S>
```

4.29 RM - Interactively Modify CPU12 Register Contents

Command Line Format

RM

Parameter Description

No parameters are required. Any parameters on the command line will be ignored.

Command Description

The register modify command is used to examine and/or modify the contents of the CPU12/CPU12X's registers in an interactive manner. As each register and its contents is displayed, D-Bug12XZ allows a new value for the register to be entered in hexadecimal. If modification of the displayed register is not desired, entering a carriage return causes the next CPU12/CPU12X register and its contents to be displayed on the next line. When the last of the registers has been examined and/or modified, the RM command will redisplay the first register providing the an opportunity to make additional modifications to the CPU12/CPU12X's register contents. Typing a period (.) as the first non space character on the line will exit the interactive mode of the register modify command and return to the D-Bug12XZ prompt.

The registers are displayed in the following order, one register per line: PC, SP, X, Y, A, B, CCR.

Restrictions

The CPU registers may not be modified when the 'R>' prompt is being displayed.

Example

```
S>rm

PC=524D <CR>
SP=4000 <CR>
IX=0000 <CR>
IY=0000 <CR>
D=0000 <CR>
CCRW=00D8 <CR>
A=00 <CR>
B=00 <CR>
PC=524D .
S>
```

4.30 STOP - Stop Execution of code in the target MCU

Command Line Format

STOP

Parameter Description

No parameters are required. Any parameters on the command line are ignored.

Command Description

The STOP command is used to halt target program execution and place the target MCU in active background debug mode.

Restrictions

The STOP command cannot be used when the 'S>' prompt is being displayed.

Example

```
R>stop
Target Processor Has Been Stopped

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C0A9  3100  00FA  3101     24:F9        1100 0001
xx:C0A9  20FE              BRA   $C0A9
S>stop
Target CPU Not Running
S>
```

4.31 SECURE – Secure Target MCU

Command Line Format

SECURE [<SecByteVal>]

Parameter Description

<SecByteVal> An 8-bit hexadecimal number used for the target MCU security byte value.

Command Description

The S12/S12X Family of devices contain a security mechanism preventing access to the contents of the internal Flash and EEPROM via the BDM or expanded operating modes. The security mechanism, controlled by the two least significant bits of the security byte located at $FF0F, is automatically programmed to the unsecured state ($FE) by the FBULK command.

The `SECURE` command is used to place a connected target device into the secured state. If the optional command line parameter representing the security byte value is not supplied, the target's security byte is programmed with a value of $FD, placing the target device in a secured state.

After the security byte has been programmed, the next time target MCU is reset, it will be placed in the secured state. Note that the D-Bug12XZ ***does not*** reset the target MCU upon completion of the `SECURE` command.

Note: The 0K36N mask set of the MC9S12DP256 microcontroller cannot be erased and unsecured via the BDM. Therefore, it is recommended that these devices not be placed in secure mode using the `SECURE` command.

Restrictions

The two least significant bits of the supplied command line parameter, <SecByteVal>, must contain a value *other than* 1:0, which is the unsecured state.

The SECURE command cannot be used when the 'R>' prompt is being displayed.

Example

```
S>md ff0f

FF00  FF FF FF FF - FF FF FF FF - FF FF FF FF - FF FF FF FE  ................
S>secure
Are you sure you want to secure the target device? (Y/N) Y
S>md ff0f

FF00  FF FF FF FF - FF FF FF FF - FF FF FF FF - FF FF FF FD  ................
S>reset
Can't Communicate With Target CPU

1.) Reset Target
2.) Hot Connect to Target
3.) Erase & Unsecure
4.) Enter BDM debugger
? 3
S> md ff0f

FF00  FF FF FF FF - FF FF FF FF - FF FF FF FF - FF FF FF FE  ................
S>
```

4.32 T – Trace (Execute) CPU12/CPU12X Instruction(s)

Command Line Format

T [<Count>]

Parameter Description

<Count> An 8-bit decimal number in the range 1..255

Command Description

The Trace command is used to execute one or more program instructions beginning at the current Program Counter (PC) location. As each program instruction is executed, the CPU12's register contents are displayed and the *next* instruction to be executed is displayed. A single instruction may be executed by entering the trace command followed immediately by a carriage return.

Restrictions

The Trace command cannot be executed when the 'R>' prompt is being displayed.

Example

```
S>rd

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C029  4000  0000  0000     00:00         1101 1000
xx:C029  CF3100           LDS   #$3100
S>t

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C02C  3100  0000  0000     00:00         1101 0000
xx:C02C  07D2             BSR   $C000

S>t

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C000  30FE  0000  0000     00:00         1101 0000
xx:C000  FEC033           LDX   $C033

S>t

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
0E C003  30FE  C037  0000     00:00         1101 1000
xx:C003  FDC031           LDY   $C031

S>
```

4.33 TRACE – Configure or display debug module trace data

Command Line Format

TRACE [“EN” | “DI” | “CPU12X” | “XGATE” | “BOTH” | “HISTORY” [;g]]

Parameter Description

Each of the listed ASCII parameters may be entered in upper, lower or mixed case.

Command Description

For those S12, S12X and S12Z devices containing the trace feature as part of the on-chip DBG module, the TRACE command allows setup and display of data captured in the DBG module’s trace buffer. The TRACE command only supports ‘Normal’ mode which captures Change Of Flow (COF) Program Counter addresses in the buffer which can be used to reconstruct program flow. The ‘alignment’ of the collected trace data is such that the debug module begins recording data as soon as the execution of the target application begins (D-Bug12XZ prompt changes from “`S>`” to “`R>`”) and ends the tracing session when the MCU enters active background mode (D-Bug12XZ prompt changes from “`R>`” to “`S>`”).

The “EN” command line option ENables the tracing function on the DBG module. Anytime application code is run via the G, GT or TO command the DBG module records COF Program Counter addresses until a break point is reached or the STOP command is used to halt target application execution. The “DI” option will disable the trace function of the DBG module. The enabled or disabled state of the TRACE command is saved as part of D-Bug12XZ’s configuration data, therefore, once the feature is enabled or disabled, it will remain in that state.

When connected to a target device in the S12X family containing an XGate co-processor, the “CPU12X”, “XGATE” and “BOTH” options are used to choose whether the DBG trace buffer will record only CPU12X or XGate COF addresses or will record COF addresses for both CPU cores. The “CPU12X” option can be used when an S12XS device is connected as the target device, but the option is essentially ignored.

Once valid data has been recorded in the trace buffer, the “HISTORY” command line parameter can be used to display the trace buffer address contents with the disassembled instructions at the recorded addresses. The optional “`;g`” parameter, which is only used in conjunction with the “HISTORY” parameter, will display all CPU addresses in Global or linear format for for S12 and S12X targets. By default addresses are displayed in Paged or Banked format.

Entering the TRACE command with no parameters will display the currently saved settings for the TRACE command. Note that the “`;g`” option used with the HISTORY parameter is not saved.

For devices containing an S12Z CPU, there are three types of COF program counter addresses recorded in the trace buffer. Source addresses of ***taken*** conditional branches, which includes bit-conditional and loop primitive instructions, are indicated by preceeding these disassembled instructions with "`S:`". Destination addresses of ***indexed*** JMP and JSR instructions, RTI and RTS instructions are indicated by preceeding these disassembled instructions with "`D:`". Finally, the vector address of interrupts are recorded in the trace buffer. Vector address entries actually cause two lines of data to be displayed. First, the address of the vector is displayed between brackets "`[ ]`", preceeded by "`V:`". The second line displayed is the first instruction if the interrupt service routine preceeded by "`I:`". An example of the S12Z trace buffer history format is shown below.

```
S:FE00CC  2267                    BHI       $FE00B3
S:FE00B3  028002C009              BRCLR.B   $02C0,#0,$FE00BC
S:FE00BC  029002C007              BRCLR.B   $02C0,#1,$FE00C3
V:[FFFE64]
I:FE00DC  EDB00AE4                BSET.B    $0AE4,#3
D:FE11ED  EDC00A0D                BSET.B    $0A0D,#4
D:FE127A  A465                    LD        D0,(5,SP)
D:FE1049  ECC00A0C                BCLR.B    $0A0C,#4
D:FE1295  A464                    LD        D0,(4,SP)
D:FE0E86  ECC00A0B                BCLR.B    $0A0B,#4
D:FE12A3  A464                    LD        D0,(4,SP)
D:FE0D01  EDC00A0A                BSET.B    $0A0A,#4
D:FE12BE  A460                    LD        D0,(0,SP)
D:FE0B9B  EDC00A09                BSET.B    $0A09,#4
D:FE12CC  A460                    LD        D0,(0,SP)
D:FE0A35  EDC00A08                BSET.B    $0A08,#4
D:FE12E7  0A61                    LEA       SP,(1,SP)
D:FE00F1  0A62                    LEA       SP,(2,SP)
```

The trace buffer in the DBG module of devices containing a S12X or S12S CPU record the same three types of COF program counter addresses. In addition, for those S12X devices containing a XGate coprocessor, the program counter addresses recorded for the XGAte CPU include source address of *taken* conditional branches, destination address of *indexed* JAL instructions and the first XGATE code address in a thread. In addition, on devices where the XGATE module supports multiple interrupt levels (S12XE devices), an entry in the trace buffer is made for the first instruction executed when an interrupted thread resumes execution.

For the CPU12X or CPU12S, the labels for source, destination, vector and first ISR instruction become "`SC:`", "`DC:`", "`VC:`" and "`IC:`", where the "`C`" indicates the CPU12X or CPU12S. For XGate trace buffer entries, the labels "`SX:`", "`DX:`", "`TX:`" and "`CX:`", are used for the source, destination, thread start and thread continuation, respectively.

When trace data for both The CPU12X and XGate are recorded, the COFs occur independently of each other and the profile of the COFs for the two sources is totally different. When a COF occurs in either source (CPU12X or XGate) a trace buffer entry is made and the current PC of

the other source is simultaneously stored to the trace buffer even if no COF has occurred. These addresses are displayed with a label of "`PC:`" for the CPU12X and "`PX:`" for XGate. Refer to the example below.

```
DC:79B589  16BF14            JSR   $BF14
DC:79B58C  06BD5D            JMP   $BD5D
DC:79BD5F  18A600            GLDAA 0,X
SC:79BD62  1826F816          LBNE  $B57C
DC:79B57F  18E600            GLDAB 0,X
DC:78BC22  37                PSHB
SC:78BC4F  2606              BNE   $BC57
DC:79BF31  C60F              LDAB  #$0F
DC:79BF1D  3D                RTS
DC:79B589  16BF14            JSR   $BF14
DC:79B58C  06BD5D            JMP   $BD5D
DC:79BD5F  18A600            GLDAA 0,X
DC:79B494  1B83              LEAS  3,SP
DC:7915AB  1B83              LEAS  3,SP
DC:79197B  1B86              LEAS  6,SP
SC:79197F  2702              BEQ   $9983
SC:7F7BB1  2416              BCC   $BBC9
SC:7F7BCA  2404              BCC   $BBD0
```

CPU12X Trace History Example

```
SX:860A  29F8  BPL    $85FC
SX:8602  3002  BHI    $8608
DX:8406  5AA0  STW    R2,(R5,#0)
SX:840C  2247  BCS    $849C
TX:85B2  F5CC  LDL    R5,#204
SX:85B8  2602  BEQ    $85BE
SX:853E  260D  BEQ    $855A
SX:8562  260A  BEQ    $8578
SX:8590  2401  BNE    $8594
DX:85D0  0200  RTS
TX:85B2  F5CC  LDL    R5,#204
SX:85B8  2602  BEQ    $85BE
SX:853E  260D  BEQ    $855A
SX:8562  260A  BEQ    $8578
SX:8590  2401  BNE    $8594
DX:85D0  0200  RTS
```

XGate Trace History Example

```
PC:[78BC69]
SX:8534  21FE  BCC    $8532
PC:[78BC6B]
SX:8534  21FE  BCC    $8532
PC:[78BC6F]
SX:8534  21FE  BCC    $8532
PC:[78BC6F]
SX:8534  21FE  BCC    $8532
PC:[78BC71]
SX:853E  260D  BEQ    $855A
DC:79BF31  C60F              LDAB  #$0F
PX:[8560]
PC:[79BF31]
SX:8562  260A  BEQ    $8578
PC:[79BF3C]
SX:8590  2401  BNE    $8594
PC:[79BF40]
SX:85A2  2404  BNE    $85AC
DC:79BF1D  3D                RTS
PX:[85B0]
PC:[79BF1D]
DX:85D0  0200  RTS
DC:79B7A8  1862F022          INCW  34,SP
PX:[0000]
SC:79B7B1  26E6              BNE   $B799
PX:[0000]
DC:78BC22  37                PSHB
PX:[0000]
SC:78BC4F  2606              BNE   $BC57
PX:[0000]
DC:79BF31  C60F              LDAB  #$0F
PX:[0000]
DC:79BF1D  3D                RTS
PX:[0000]
DC:79B7A8  1862F022          INCW  34,SP
PX:[0000]
SC:79B7B1  26E6              BNE   $B799
PX:[0000]
DC:78BC22  37                PSHB
PX:[0000]
SC:78BC4F  2606              BNE   $BC57
PX:[0000]
DC:79BF31  C60F              LDAB  #$0F
PX:[0000]
DC:79BF1D  3D                RTS
PX:[0000]
```

CPU12X and XGate Trace History Example

Restrictions

The TRACE command cannot be used when the 'R>' prompt is being displayed.

The "**;g**" option can only be used when an S12 or S12X device is the connected target.

Example

```
S>trace en
S>trace cpu12x

S>trace history
SC:E2:BC52  2606              BNE   $BC5A
DC:E6:BF31  C60F              LDAB  #$0F
DC:E6:BF1D  3D                RTS
.
.
.
SC:FD:BBB1  2416              BCC   $BBC9
SC:FD:BBCA  2404              BCC   $BBD0
DC:E6:B488  1A03              LEAX  3,X
DC:E6:BD5F  18A600            GLDAA 0,X
SC:E6:BD62  1826F816          LBNE  $B57C
DC:E6:B57F  18E600            GLDAB 0,X
DC:E2:BC25  37                PSHB
SC:E2:BC52  2606              BNE   $BC5A
DC:E6:BF31  C60F              LDAB  #$0F
DC:E6:BF1D  3D                RTS
DC:E6:B589  16BF14            JSR   $BF14
DC:E6:B58C  06BD5D            JMP   $BD5D
DC:E6:BD5F  18A600            GLDAA 0,X
SC:E6:BD62  1826F816          LBNE  $B57C
DC:E6:B57F  18E600            GLDAB 0,X
DC:E2:BC25  37                PSHB
SC:E2:BC52  2606              BNE   $BC5A
DC:E6:BF31  C60F              LDAB  #$0F
DC:E6:BF1D  3D                RTS
DC:E6:B589  16BF14            JSR   $BF14
DC:E6:B58C  06BD5D            JMP   $BD5D
DC:E6:BD5F  18A600            GLDAA 0,X
DC:E6:B494  1B83              LEAS  3,SP
DC:E4:95AB  1B83              LEAS  3,SP
DC:E4:997B  1B86              LEAS  6,SP
SC:E4:997F  2702              BEQ   $9983
SC:FD:BBB1  2416              BCC   $BBC9
SC:FD:BBCA  2404              BCC   $BBD0
DC:E4:9989  34                PSHX
DC:E4:999A  1B87              LEAS  7,SP

S>
```

4.34 TO – Trace Over Subroutine Calls (JSR, BSR, CALL)

Command Line Format

TO

Parameter Description

No parameters are required. Any parameters on the command line are ignored.

Command Description

When tracing through code it is often unnecessary to trace through subroutines that are known to be bug free. The TO command is similar to the Trace command (T) except that subroutine calls (BSR, JSR and CALL) are traced as a single instruction. When the TO command encounters one of the subroutine call instructions, it places a temporary breakpoint at the instruction following the BSR, JSR or CALL. It then issues a Go command, executing the subroutine at full speed. All other instructions are executed the same as if the Trace command were used.

If a subroutine requires more than a handful of cycles to execute, D-Bug12XZ will display its 'R>' prompt indicating the target is running. When the subroutine returns, the temporary breakpoint is removed, the CPU12/CPU12X's register contents are displayed and the *next* instruction to be executed is displayed.

Restrictions

The TO command cannot be used when the 'R>' prompt is being displayed.

Example

```
S>to

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
00 1000  4000  0000  0000     FF:FE        1101 1100
xx:1000  CCFFFF        LDD   #$FFFF
S>to

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
00 1003  4000  0000  0000     FF:FF        1101 1000
xx:1003  161100        JSR   $1100
S>to
R>

PP  PC    SP    X     Y    D = A:B   CCR = SXHI NZVC
00 1006  4000  0000  0000     FF:FE        1101 1100
xx:1006  A7            NOP
```

4.35 UPLOAD – Display Memory In S-Record Format

Command Line Format

UPLOAD <StartAddress> [<EndAddress>] | FLASH | DFLASH | EE | EEE | IO | RAM [;<SRecSize>]

Parameter Description

<StartAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
<EndAddress>	A Local, Global/Linear, Paged CPU12/CPU12X or XGate address.
FLASH	The ASCII characters “FLASH” in upper, lower or mixed case.
DFLASH	The ASCII characters “DFLASH” in upper, lower or mixed case.
EE	The ASCII characters “EE” in upper, lower or mixed case.
EEE	The ASCII characters “EEE” in upper, lower or mixed case.
IO	The ASCII characters “IO” in upper, lower or mixed case.
RAM	The ASCII characters “RAM” in upper, lower or mixed case.

Command Description

The UPLOAD command is used to display the contents of memory in Motorola S-Record format. In addition to displaying the specified range of memory, the UPLOAD command also displays an S9 end-of-file record after the S-Record containing the data from the last memory address is displayed. The output of this command may be captured by a terminal program and saved to a disk file.

The parameters ‘FLASH’, ‘DFLASH’, ‘EE’, ‘EEE’, ‘IO’ or ‘RAM’can be used to display the contents of the associated on-chip memory resource without having to enter its start and end address. This feature is particularly helpful when performing a ‘hot-connect’ to an unresponsive target MCU since it allows some of the critical memory resources to be displayed quickly. This can be particularly advantages for devices in the 0.25μ S12 family where the RAM, EEPROM and I/O registers may be relocated in the 64K local memory map by the running application. Because the RAM, EEPROM and I/O registers can overlap, only the visible portion of the associated resource will be displayed when using one of the command line mnemonic parameters.

Entering an ASCII Escape (0x1b) or control C (0x03) character during the display of target memory data will cause D-Bug12XZ to return to the command prompt.

The optional <SRecSize> parameter may be used to specify the length of the S-Record data field. Permissible values for <SRecSize> range from 16 through 64. If the <SRecSize> parameter is not specified, the default S-Record length is 32.

Restrictions

The address space defined by the <StartAddress> and <EndAddress> must all be in the same address space. That is, both addresses must be a Local or Global/Linear or Paged CPU12/CPU12X or XGate address.

For 0.25μ S12 devices, only local (i.e. 16-bit) addresses may be used for the <StartAddress> and <EndAddress> parameters while the 'R>' prompt is displayed.

Example

```
S>upload ram
S1232800FFBFFDEFFDE3BEB7E97FE5BA4FF7DB5FEF7FB78BF77FF357AE7BECC9DCD3E2311E
S123282059040062700D00808A49C3088482028A213AC04C2030920279651224A090D22A1D
S1232840F9FBAFDDB6C771BFF75BF6FDDAFF3F7F7FD7FDF3AACEFFFF6FF5D5A7EEB7B97BF6
S1232860A154278310A74066837841C080E51004031190470B25580A0D23B0652324900248
S12328807DBBD7D9FF77E9FFF6AED7DD33FFCBF9E6FFB7BE7B9FD1FCF7CFDFF6ADFC55F8CE
S12328A0D989B4B37019A0324128849382DAA810260420792086B0012BB02059E200852067
S12328C0070AEFD1AFF93DEA9EEFFFCDDD358F63FFDBB8F4FEFD7F5FEDDD4FF3F679397F65
S12328E0D280D119E23132378618D5510125240502304800B197010050A8C080353E880112
S1232900FFFF3BDFF9FE3B77F7DD03FDE7C6D95FE777FBF54FE7F81FD376D0B09EE2BD1E7A
S123292022524D221E6B939086800903004EC01048A2407C344789015260087A4404390074
S1232940FF76FF5DFBD5F1BE1DBEBE77D775ECAD4F8EFF7FAFFFFF8EFE5E7615EFBFFF8F75
S12329608752A3353328578C51012A800036428498490680 85BE4261180565040F2B4A18FD
S1232980BDDF99FEBED9EBC6FC3F09F9DE9ED976E7FFD67BBFFFFE65E6F9FF715CC2DE0D00
.
.
.
S1233EA0BE3000240001B468010A2C000619401825438686460065002C1001102508513007
S1233EC0FA77F49FBDF7FD1FF2EB96DFBFFE4CB9FEE7F4F7FFF69FF6BE87DEE7FF1F1DFF58
S1233EE03C702A8A1B5C1503A50432E428C8092E28108015A088933038321829 92C20388A7
S1233F00F16EFBF87EF3FFF6FFFAF9EA626FFFFE7BFEABF763FAEB36BCBDA617FF36953E64
S1233F200F629410400007C01800021028A1046AB6C0218002A018B9945222404185073928
S1233F40C7FD9ED78DF7EDE7E7F1A0DCDD6B7BE7FF737878C7F1D3F3FF9D79B3FDF7E8BB8F
S1233F6080A381220A56603D8C98802E8E449424C2E3AE028844362540C10A96418C3427D9
S1233F80BFFFDBFC9EFFF27D7EFFA37EECFFFF2F3FF399DFCACFB7593BCA99BECE9FF5FFB5
S1233FA0A23D46B0BE18B506981043DD2A2020090C1002C1A2008CAA0342649AAC82903074
S1233FC0E7BFAAF7FFFEA3F0E3BBEDB5AF36E1605BE47FD1DA97DDD99BFD7F9ED6DCD75F4D
S1233FE0C006361104914294222152131000A9028A90160311436A020463404211886CC4A62
S9030000FC
S>
```

4.36 VER – Display D-Bug12XZ Version and Revision

Command Line Format

VER

Parameter Description

No parameters are required. Any parameters on the command line are ignored.

Command Description

The VER command that displays the current version and revision of D-Bug12X. This allows users or a high-level debugger to determine the version of D-Bug12XZ so they can determine what features are available.

Restrictions

None.

Example

```
S>ver
5.0.0b1
S>
```

4.37 VERF – Compare S-Record File To Memory Contents

Command Line Format

VERF [<AddressOffset>]

Parameter Description

<AddressOffset> A 32-bit hexadecimal number

Command Description

The VERF command is used to compare the data contained in an S-Record object file to the contents of target memory. The address offset, if supplied, is added to the load address of each S-Record before an S-Record's data bytes are compared to the contents of memory. Providing an address offset other than zero allows the S-Record's object code or data to be compared against memory other than that for which the S-Record was assembled.

During the verification process, the ASCII characters '|', '/', '-' and '\' are sent one at a time to the control console to indicate that the verify process is proceeding. Before each character is sent, an ASCII backspace character is sent to the console so that the previously sent progress character is effectively erased from the screen. The displayed effect is a rotating bar. When an S-Record file has been successfully verified, D-Bug12XZ will issue its prompt.

If the contents of target memory does not match the corresponding data in the S–Record, an informational line is displayed on the console showing the S-Record address, the S-Record data and the data at the corresponding target memory location. Note that the displayed S-Record address includes the optional address offset that may have been entered on the command line.

The VERF command is terminated when D-Bug12XZ receives an 'S7', 'S8' or 'S9' end of file record. If the object file being verifyed does not contain an 'S7' 'S8' or 'S9' record, D–Bug12X will not return its prompt and will continue to wait for the end of file record.

Entering an ASCII Escape (0x1b) or control C (0x03) character from the keyboard at any time during the VERIFY command will terminate the command and return to the D–Bug12X command line prompt.

Restrictions

None.

Example

```
S>verf
S-Rec Address  S-Rec Data  Target Address  Target Data
$00200         $00         $0200           $C6
$002C3         $00         $02C3           $87
$00397         $00         $0397           $EB
$005BA         $00         $05BA           $A2
$007DF         $00         $07DF           $61

S>
```

4.38 XASM – XGate Single Line Assembler/Disassembler

Command Line Format

XASM <TAddress>

Parameter Description

<TAddress> A Local, Global/Linear, Paged or XGate Target address.

Command Description

The XGate assembler/disassembler is an interactive memory editor that allows memory contents to be viewed and altered using assembly language mnemonics. Each entered source line is translated into machine language code and placed into memory at the time of entry. When displaying memory contents, each instruction is disassembled into its source mnemonic form and displayed along with the hexadecimal machine code and any instruction operands.

Assembler mnemonics and operands may be entered in any mix of upper and lower case letters. Any number of spaces may appear between the assembler prompt and the instruction mnemonic or between the instruction mnemonic and the operand. By default, numeric values appearing in the operand field are interpreted as signed decimal numbers. Placing a $ in front of a number causes the number to be interpreted as a hexadecimal number.

When an instruction has been disassembled and displayed, the assembler prompt is displayed following the disassembled instruction. If a carriage return is entered immediately following the prompt, the next instruction in memory is disassembled and displayed on the next line.

If an XGate instruction is entered following the prompt, the entered instruction is assembled and placed in memory. The line containing the new entry is erased and the new instruction is disassembled and displayed on the same line. The contents of the next memory location(s) is disassembled and displayed on the screen.

The instruction mnemonics and operand formats accepted by the assembler follow the syntax as described in the MC9S12XE Family Reference Manual.

When entering branch instructions, the number placed in the operand field should be the absolute destination address of the instruction. The assembler will calculate the twos compliment offset of the branch.

Termination of the assembly/disassembly process is accomplished by entering a period (.) and carriage return immediately following the assembler prompt.

Restrictions

None.

Example

```
S>xasm b224'x
X:B224  C702  SUBL   R7,#2       >
X:B226  F102  LDL    R1,#2       >
X:B228  F240  LDL    R2,#64      >
X:B22A  FA03  LDH    R2,#3       >
X:B22C  5145  STB    R1,(R2,#5)  >
X:B22E  F350  LDL    R3,#80      >
X:B230  FBD0  LDH    R3,#208     >
X:B232  4260  LDB    R2,(R3,#0)  >
X:B234  0A1C  LSL    R2,#1       >
X:B236  F428  LDL    R4,#40      >
X:B238  FCD0  LDH    R4,#208     >
X:B23A  F3C0  LDL    R3,#192     >
X:B23C  FB02  LDH    R3,#2       >
X:B23E  4970  LDW    R1,(R3,#16) >
X:B240  7988  STW    R1,(R4,R2)  >.
S>
```

Assembly Operand Format

This section describes the operand format used by the XGate assembler when assembling instructions. The operand format accepted by the assembler is described separately in the *MC9S12XE Family Reference Manual*. Rather that describe the numeric format accepted for each instruction, some general rules apply. Exceptions and complicated operand formats are described separately.

In general, anywhere the XGate assembler expects a numeric value in the operand field, either a decimal or hexadecimal value may be entered. Decimal numbers are entered as signed constants having a range of -32768..65535. A leading minus sign (-) indicates negative numbers, the absence of a leading minus sign indicates a positive number. A leading plus sign (+) is not allowed. Hexadecimal numbers must be entered with a leading dollar sign ($) followed by one to four hexadecimal digits. The default number base is decimal.

For all branch instructions, (Bcc, BRA) the number entered in the address portion of the operand field must be the ***absolute address of the branch destination***. The assembler will calculate the two's compliment offset that is placed in the assembled object code.

Disassembly Operand Format

This section describes the operand format for the disassembler that is used in conjunction with the single line assembler. The operand format used by the disassembler is described

separately in the *MC9S12XE Family Reference Manual*. Rather that describe the format used for each instruction, some general rules can be applied.

All numeric values disassembled as hexadecimal numbers will be preceded by a dollar sign ($) to avoid being confused with values disassembled as signed decimal numbers.

For all branch (Bcc, BRA) instructions the numeric value of the address portion of the operand field will be displayed as the hexadecimal ***absolute address of the branch destination***.

All offsets used with indexed addressing modes will be disassembled as decimal numbers.

4.39 XBR – Set/Display XGate software breakpoints

Command Line Format

XBR [<TAddress> [<TAddress>…<TAddress>]]

Parameter Description

<TAddress> A Local, Global, Paged or XGate RAM Target address.

Command Description

The XBR command is used to set an XGate software breakpoint at an XGate RAM address or to display any previously set XGate software breakpoints. The function of a breakpoint is to halt XGate program execution when the program reaches the breakpoint address. When XGate encounters a software breakpoint, it is placed in debug mode. D-Bug12XZ will disassemble the instruction at the breakpoint address, display the XGate's register contents, and wait for the next D-Bug12XZ command to be entered by the user. Note that when the XGate CPU is placed in debug mode, it does not affect the state of the CPU12X.

XGate software breakpoints are set by entering the XBR command followed by one or more target addresses. Entering the XBR command without any target addresses will display all the currently set XGate software breakpoints. A maximum of 10 XGate software breakpoints may be set at any one time.

XGate software breakpoints may be set or removed (see XNOBR command) regardless of the current state of the XGate CPU. If a breakpoint is set with the XGate in debug mode, the breakpoint is only inserted into the XGate software breakpoint table. If XGate is in active mode (i.e. executing a thread or ready to execute a thread) the breakpoint is inserted into the XGate software breakpoint table and immediately written to memory.

Restrictions

The XBR command may only be used to place breakpoints in XGate RAM memory

XGate software breakpoints may only be placed at even XGate RAM addresses.

Error Conditions

If an invalid hexadecimal value is provided as part of the <TAddress>, an appropriate error message shall be issued, command execution will be terminated and a breakpoint for that address will not be set.

If an attempt is made to enter more than 10 XGate software breakpoints, command execution will be terminated and an appropriate error message will be displayed.

If the supplied target address is outside the range of the target's XGate RAM memory, a breakpoint(s) for that address will not be set and an appropriate error message will be displayed.

If the supplied target address is an odd value, the command will be terminated and an appropriate error message will be displayed.

Example

```
R>xbr 85b2'x fd008'g fc:1842
XGate Breakpoints: X:85B2 X:D008 X:C842

R>xbr
Breakpoints: X:85B2 X:D008 X:C842

R>
```

4.40 XDBG – Enter/Exit XGate thread debug

Command Line Format

XDBG ENTER [<ChannelID>] | EXIT

Parameter Description

ENTER	The ASCII characters “ENTER” in upper, lower or mixed case.
<ChannelID>	A valid XGate channel ID number.
EXIT	The ASCII characters “EXIT” in upper, lower or mixed case.

Command Description

Unlike the CPU12/CPU12X, when an MCU containing the XGate coprocessor exits reset, the XGate CPU is in an idle state until initialized by the CPU12X. Even after being initialized, XGate remains in an idle state until an interrupt, routed to XGate by the interrupt controller, initiates execution of an XGate thread (ISR). The XGate CPU can be placed in ‘debug mode’ when a hardware or software breakpoint is reached, an XGate software error occurs or by the CPU12X.

The XDBG command forces the XGate into debug mode regardless of its current state. If the ENTER parameter is supplied on the command line without the optional Channel ID, the CPU12X simply writes the the XGDBG bit in the XGMCTL register to a ‘1’ placing the XGate CPU into debug mode. If the XGate CPU was executing a thread when the XGDBG bit was written to a ‘1’, the XGate CPU and other registers will be displayed. If the XGate CPU was in an idle state, D-Bug12XZ will simply return to its command prompt.

Supplying the optional <ChannelID> parameter will place XGate in debug mode and initiate an XGate vector fetch for the associated XGate channel from the XGate vector table. Therefore, prior to issuing the XDBG ENTER command with the optional <ChannelID> parameter, the XGate Vector Base Register, XVBR, and the vector table itself must be initialized.

When the XGate CPU is in debug mode with a thread active (i.e. the CHID register ≠ 0x00), the XGate CPU registers may be examined and/or changed, individual instructions executed (traced) and/or the thread restarted.

Entering the XDBG command with the EXIT parameter causes the XGate CPU to exit debug mode. If a thread is active, the thread will be terminated and XGate will return to the idle state.

Note that as long as XGate remains in debug mode, it will not respond to any incomming interrupt requests.

Restrictions

None.

Example

```
S>xdbg enter
XGate Has Been Placed In Debug Mode
S>xdbg exit
XGate Has Exited Debug Mode
S>xdbg enter 3c

   PC     R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B224  0000 0000 0000 0000 0000 0000 0000 B596       0000  3C   01  B000  A1
X:B224  C702  SUBL   R7,#2
S>
```

4.41 XG – Continue Execution of an XGate Thread

Command Line Format

XG

Parameter Description

None.

Command Description

If XGate is in debug mode and a thread is active (i.e. the CHID register ≠ 0x00), the XG command will cause the XGate CPU to exit debug mode and continue execution of the thread at the current XGate PC.

Restrictions

The XG command cannot be used if the XGate CPU is not in debug mode and a thread is not currently active.

Example

```
S>xdbg enter 3c

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B224  0000 0000 0000 0000 0000 0000 0000 B596       0000  3C   01  B000  A1
X:B224  C702  SUBL   R7,#2
S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B226  0000 0000 0000 0000 0000 0000 0000 B594       1000  3C   01  B000  A1
X:B226  F102  LDL    R1,#2

S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B228  0000 0002 0000 0000 0000 0000 0000 B594       1000  3C   01  B000  A1
X:B228  F240  LDL    R2,#64

S>xg
S>
```

4.42 XNOBR – Remove one/all XGate software breakpoints

Command Line Format

XNOBR [<TAddress> [<TAddress>…<TAddress>]]

Parameter Description

<TAddress> A Local, Global, Paged or XGate RAM Target address.

Command Description

The XNOBR command is used to remove XGate software breakpoints set using the XBR command.

XGate software breakpoints are removed by entering the XNOBR command followed by one or more target addresses. Entering the XNOBR command without any target addresses will delete all XGate software breakpoints.

XGate software breakpoints may be set or removed (see XNOBR command) regardless of the current state of the XGate CPU. If a breakpoint is removed with the XGate in debug mode, the breakpoint is only removed from the XGate software breakpoint table. If XGate is in active mode (i.e. executing a thread or ready to execute a thread) the breakpoint is removed from the XGate software breakpoint table and immediately removed from memory.

Restrictions

The XNOBR command may only be used to remove breakpoints that were set using the XBR command.

Error Conditions

If an invalid hexadecimal value is provided as part of the <TAddress>, an appropriate error message shall be issued, command execution will be terminated.

If the supplied target address is outside the range of the target's XGate RAM memory, an appropriate error message will be displayed.

If the supplied target address is an odd value, the command will be terminated and an appropriate error message will be displayed.

Example

```
R>xbr 85b2'x fd008'g fc:1842
XGate Breakpoints: X:85B2 X:D008 X:C842

R>xnobr d008'x
XGate Breakpoints: X:85B2 X:C842

R>xnobr
Breakpoints:

R>
```

4.43 XRD – Display XGate CPU Registers

Command Line Format

XRD

Parameter Description

None.

Command Description

The XRD command is used to display the XGate CPU and other XGate related registers. The XGate CPU register contents are only valid if XGate is in debug mode with a thread active. Otherwise, the XGate CPU register contents will be displayed as '****'. In addition to the CPU register contents, the Channel ID register (CHID), Channel Interrupt Priority (CHPL), XGate Vector Base (XVBR) and the Module Control Register (MCTL) registers are displayed. The contents of these four registers are valid at all times.

Restrictions

None.

Example

```
S>xrd

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B240  0000 0348 0004 02C0 D028 0000 0000 B594       0000  3C   01  B000  A1
X:B240  7988  STW    R1,(R4,R2)
S>xg
S>xrd

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  ****  **** **** **** **** **** **** **** ****       ****  00   00  B000  81

S>
```

4.44 XT – Trace (Execute) XGate Instruction(s)

Command Line Format

XT [<Count>]

Parameter Description

<Count> An 8-bit decimal number in the range 1..255

Command Description

The XT command is used to execute one or more program instructions beginning at the current XGate Program Counter (PC) location. As each program instruction is executed, the XGate's register contents are displayed and the *next* instruction to be executed is displayed. A single instruction may be executed by entering the XT command followed immediately by a carriage return.

Restrictions

The Trace command cannot be executed if XGate is not in debug mode.

Example

```
S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B23A  0000 0002 0004 D050 D028 0000 0000 B594       0000  3C   01  B000  A1
X:B23A  F3C0  LDL    R3,#192

S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B23C  0000 0002 0004 00C0 D028 0000 0000 B594       0000  3C   01  B000  A1
X:B23C  FB02  LDH    R3,#2

S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B23E  0000 0002 0004 02C0 D028 0000 0000 B594       0000  3C   01  B000  A1
X:B23E  4970  LDW    R1,(R3,#16)

S>
```

4.45 <CPU12/CPU12X/CPU12Z RegisterName> - Modify a CPU Register Value

Command Line Format

<RegisterName> <RegisterValue>

Parameter Description

Where <RegisterName> is one of the following CPU12/CPU12X/CPU12Z register names:

Register Name	Description	Valid Range
PC	Program Counter	$0..$FFFF[1]
SP	Stack Pointer	$0..$FFFF[1]
X	X-Index Register	$0..$FFFF[1]
Y	Y-Index Register	$0..$FFFF[1]
A	A Accumulator	$0..$FF
B	B Accumulator	$0..$FF
D	D Accumulator (A:B)	$0..$FFFF
CCR	Condition Code Register	$0..$FF
CCRW	Condition Code Register Word	$0..$FFFF
PP	PPAGE Register	$0..$FF
EP	EPAGE Register	$0..$FF
RP	RPAGE Register	$0..$FF
GP	GPAGE Register	$0..$FF
D0	Data Register D0	$0..$FF
D1	Data Register D1	$0..$FF
D2	Data Register D2	$0..$FFFF
D3	Data Register D3	$0..$FFFF
D4	Data Register D4	$0..$FFFF
D5	Data Register D5	$0..$FFFF
D6	Data Register D6	$0..$FFFFFFFF
D7	Data Register D7	$0..$FFFFFFFF

[1]For the CPU12Z, PC, SP, X and Y are 24-bit registers and accept values in the range $0..$FFFFFF.

Each of the fields in the CCR may be modified by using the following field Names:

CCR Bit Name	Description	Valid Range
S	STOP Enable	0..1
H	Half Carry	0..1
N	Negative Flag	0..1
Z	Zero Flag	0..1
V	Twos Complement Overflow Flag	0..1
C	Carry Flag	0..1
IM	IRQ Interrupt Mask	0..1
XM	XIRQ Interrupt Mask	0..1
IPL	Interrupt Priority Level	0..7

For each of the CPU register names, <RegisterValue> is an 8-, 16-, 24- or 32-bit hexadecimal number. For the CCR bit names, except for the IPL bits, only a value of zero or one may be supplied for the <RegisterValue> parameter.

Command Description

This set of commands uses the CPU12/CPU12X register names to allow changing the contents of individual registers. Each register name or Condition Code Register bit name is entered on the command line followed by a space, then followed by the new register or bit value. The successful alteration of a CPU register or CCR will cause the CPU12's register contents to be displayed.

Restrictions

These commands may not be used when the 'R>' is being displayed.

If a value outside the range for a given register is entered, an error message is displayed and command execution is terminated leaving the register contents unaltered.

Example

```
S>rd

PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
E7 9DC7  3FF7  0000  0000     00:00         0  1100 0100  78 FD FE
E7:9DC7  6B84              STAB  4,SP
S>x 1234

PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
E7 9DC7  3FF7  1234  0000     00:00         0  1100 0100  78 FD FE
E7:9DC7  6B84              STAB  4,SP
S>y 5678
```

```
PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
E7 9DC7  3FF7  1234  5678     00:00         0  1100 0100  78 FD FE
E7:9DC7  6B84              STAB  4,SP
S>ipl 7

PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
E7 9DC7  3FF7  1234  5678     00:00         7  1100 0100  78 FD FE
E7:9DC7  6B84              STAB  4,SP
S>
```

4.46 <XGateRegisterName> - Modify an XGate Register Value

Command Line Format

<XGateRegisterName> <RegisterValue>

Parameter Description

Where <XGateRegisterName> is one of the following XGate register names:

Register Name	Description	Valid Range
XPC	Program Counter	$0..$FFFF
R1 – R7	General Purpose Register	$0..$FFFF
XCCR	Condition Code Register	$0..$0F
XVBR	XGate Vector Base Register	$0..$FFFF
MCTL	Module Control Register	$0..$FFFF
CHID	Channel ID/Interrupt Priority Register	$0..$78, $0..$7807
SP13	Initial R7 value for IPL 1 – 3	$0..$FFFF
SP47	Initial R7 value for IPL 4 – 7	$0..$FFFF

Each of the fields in the CCR may be modified by using the following field Names:

CCR Bit Name	Description	Legal Range
XN	Negative Flag	0..1
XZ	Zero Flag	0..1
XV	Twos Complement Overflow Flag	0..1
XC	Carry Flag	0..1

For each of the CPU register names, <RegisterValue> is an 8- or 16-bit hexadecimal number. For the CCR bit names, only a value of zero or one may be supplied for the <RegisterValue> parameter.

Command Description

This set of commands uses the XGate register names to allow changing the contents of individual registers. Each register name or Condition Code Register bit name is entered on the command line followed by a space, then followed by the new register or bit value. The successful alteration of an XGate CPU register or CCR will cause the XGate's register contents to be displayed.

The XVBR, MCTL, CHID, SP13 and SP47 registers, while part of the XGate module, are not part of the XGate CPU core. Of these registers, the MCTL register can be written whether or not XGate is in debug mode. The XVBR, SP13 and SP47 registers may only be modified if XGATE requests are disabled (XGE = 0) and idle (XGCHID = $00). The

CHID/CHPL register can only be written in debug mode.

The 16-bit MCTL register is divided into two halves. The upper 8-bits consists of ‘mask’ bits controlling write access to the associated control bits in the lower 8-bits of the register. A ‘1’ in a mask bit position enables writing to associated control bits in the lower half of the register. For the MCTL command, if an 8-bit value less than $78 is supplied as the command line parameter, D-Bug12XZ will supply a value of $FF for the mask bits when writing the MCTL register. If a value greater than $FF is supplied on the command line, the value is written directly to the MCTL register.

Of the CHID/CHPL 8-bit register pair, the CHID register (upper byte, lower address) can be written as a single 8-bit byte. However, the CHPL register (lower byte, higher address) can only be written as a single 16-bit write to the CHID address. If the parameter to the CHID command has a value less than or equal to $78, only the CHID register is written. If the parameter is $100 or greater a single 16-bit write is made to the CHID address.

Restrictions

If a value outside the range for a given register is entered, an error message is displayed and command execution is terminated leaving the register contents unaltered.

Example

```
S>xt

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B236  0000 0002 000A D050 0000 0000 0000 B594       0000  3C   01  B000  A1
X:B236  F428  LDL    R4,#40

S>r5 1234

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B236  0000 0002 000A D050 0000 1234 0000 B594       0000  3C   01  B000  A1
X:B236  F428  LDL    R4,#40
S>r6 5678

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B236  0000 0002 000A D050 0000 1234 5678 B594       0000  3C   01  B000  A1
X:B236  F428  LDL    R4,#40
S>xc 1

   PC    R0   R1   R2   R3   R4   R5   R6   R7  CCR = NZVC CHID CHPL XVBR MCTL
  B236  0000 0002 000A D050 0000 1234 5678 B594       0001  3C   01  B000  A1
X:B236  F428  LDL    R4,#40
S>
```

Appendix A

A.1 S12(X) BDM Debugger

The BDM debugger contained within D-Bug12XZ is intended to be used by tool developers or factory engineers when evaluating new silicon or debugging BDM communication problems, however, it may also be useful to end customers. Using D-Bug12XZ's low level BDM driver routines, the BDM debugger allows individual BDM commands to be sent to a target device directly from the command line. Unlike D-Bug12XZ, which maintains constant communications with a connected target, no BDM communication occurs other than during the execution of a command. Also note that the BDM debugger does not perform any checks to ensure a target is connected before executing a command.

The BDM debugger can be entered either from D-Bug12XZ's command line (see the BDMDB command description) or from the startup menu (option 4) if communication cannot be established with a target. When entering the BDM debugger from the D-Bug12XZ command line, the timing of the BDM driver routines is the same as the timing established by D-Bug12XZ. This timing will be either at the crystal divided by two rate (EXTAL/2) or the rate specified by the ALTCLK command. If the BDM debugger is entered by selecting option four of the startup menu, the timing of the BDM driver routines will be established using the displayed (by option one) crystal frequency divided by two.

The BDM debugger BSPEED command can be used to reinitialize the driver routines for an alternate BDM operating frequency if necessary. Note that the command line argument to the BSPEED command is the input frequency to the target's BDM module. If the CLKSW bit in the target's BDM status register is clear, the BDM module's input clock frequency is EXTAL/2. If the CLKSW bit is set, the BDM module's input clock frequency is the same as the target's bus clock.

For devices like the S12P and S12G family, the BDM module's input clock frequency is fixed at $f_{VCO} \div 8$. For the S12P family this is 8 MHz after reset. For the S12G family it is 6.25 MHz.

The following list summarizes the BDM debugger command set for the S12(X) devices. Each command's function and command line syntax are described in detail.

- ACKDI - Disable ACK Handshake protocol.
- ACKEN - Enable ACK Handshake protocol.
- BKGD - Place target in active background mode.
- BSPEED Reinitialize BDM drivers to clock speed of target device.
- D - Read or write the target D-accumulator.
- EXIT - Exit the BDM debugger and (re)enter D-Bug12.
- G - Exit active background, begin target execution at the current PC.
- GU - Exit active background, begin target execution at the current PC.
- PC - Read or write the target Program Counter.
- RB - Read a byte from target memory.
- RBB - Read a byte from target memory with BDM ROM & registers in map.
- RBW - Read a word from target memory with BDM ROM & registers in map.
- RESET - Reset target in special single chip mode.
- RMB – Read Multiple (8) bytes.
- RNX - Pre-increment target X-index register by 2 and read the word pointed to by X.
- RW - Read a word from target memory.
- SP - Read or write the target Stack pointer.
- SYNC - Measure target BDM speed and reinitialize BDM drivers.
- T - Execute a single instruction and return to active background.
- TG - Enable instruction tagging and begin target execution at the current PC.
- WAKE – Wake Target from STOP and Place in Active Background.
- WB - Write a byte to target memory.
- WBB - Write a byte to target memory with BDM ROM & registers in map.
- WBW - Write a word to target memory with BDM ROM & registers in map.
- WW - Write a word to target memory.
- WNX - Pre-increment target X-index register by 2 and write the word pointed to by X.
- X - Read or write the target X index register.
- Y - Read or write the target Y index register.

A.2 ACKDI - Disable ACK Handshake Protocol

Command Line Format

ACKDI

Parameter Description

None.

BDM Command Name/Type

ACK_DISABLE ($D6) / Hardware

Command Description

The ACKDI command disables the ACK handshake protocol if it was previously enabled. An ACK handshake pulse is not issued at the completion of the command.

Note: The ACK_DISABLE command is only recognized by the Enhanced BDM module. The command will be ignored by a Standard BDM module.

Example

```
?ACKDI
?
```

A.3 ACKEN - Enable ACK Handshake Protocol

Command Line Format

ACKEN

Parameter Description

None.

BDM Command Name/Type

ACK_ENABLE ($D5) / Hardware

Command Description

The ACKEN command enables the ACK handshake protocol if it was previously disabled. An ACK handshake pulse is issued at the completion of the command.

Note: The ACK_ENABLE command is only recognized by the Enhanced BDM module. The command will be ignored by a Standard BDM module.

Example

```
?ACKEN
?
```

A.4 BKGD - Place Target in Active Background Mode

Command Line Format

BKGD

Parameter Description

None.

BDM Command Name/Type

BACKGROUND ($90) / Hardware

Command Description

The BKGD command causes the target device to enter active background if firmware is enabled (ENBDM = 1). For enhanced BDM modules, an ACK handshake pulse is issued when the part enters active background mode if the ACK protocol has been previously enabled.

Example

```
?BKGD
?
```

A.5 BSPEED - Reinitialize BDM Drivers to Clock Speed of Target Device

Command Line Format

BSPEED <BDMFreq>

Parameter Description

<BDMFreq> A 16-bit decimal number representing BDM clock frequency in KHz.

BDM Command Name/Type

N/A

Command Description

The BSPEED command does not correspond to one of the BDM hardware or software commands, instead the <BDMFreq> parameter is used to reinitialize the low-level drivers to match the timing of the target's BDM. The <BDMFreq> command line argument specifies the input frequency to the target's BDM module. If the CLKSW bit in the target's BDM status register is clear, the BDM module's input clock frequency is EXTAL/2. If the CLKSW bit is set, the BDM module's input clock frequency is the same as the target's bus clock.

Example

```
?BSPEED 8000
?
```

A.6 D - Read or Write The D-accumulator

Command Line Format

D [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the D-accumulator.

BDM Command Name/Type

READ_D ($64) or WRITE_D ($44) / Firmware

Command Description

The D command is used to read or write the value of the target's D-accumulator. Entering the D command without the <Data16> parameter causes the current value of the D-accumulator to be displayed by sending the READ_D BDM command to the target. Supplying the optional <Data16> parameter results in the value of the target's D-accumulator being modified by sending the WRITE_D BDM command.

Example

```
?D
D: 1234
?D 5678
?D
D: 5678
?
```

A.7 EXIT - Exit The BDM Debugger And (re)enter D-Bug12X

Command Line Format

EXIT

Parameter Description

None.

BDM Command Name/Type

N/A

Command Description

The EXIT command does not correspond to one of the BDM hardware or software commands, instead it is used to leave the BDM debugger and return to D-Bug12. If BDM communications was established with a target in the BDM debugger, D-Bug12 will display the ‘S>’ or ‘R>’ prompt, otherwise the ‘Can't Communicate With Target CPU’ menu will be displayed.

Example

```
?EXIT
S>
```

A.8 G - Exit Active Background, Begin Target Execution at The Current PC

Command Line Format

G

Parameter Description

None.

BDM Command Name/Type

GO ($08) / Firmware

Command Description

The G command causes the target device to leave active background and begin program execution at the instruction pointed to by the current value of the Program Counter. For enhanced BDM modules, an ACK handshake pulse is issued when the part leaves active background mode if the ACK protocol has been previously enabled.

Example

```
?G
?
```

A.9 GU - Exit Active Background, Begin Target Execution at The Current PC Until

Command Line Format

GU

Parameter Description

None.

BDM Command Name/Type

GO_UNTIL ($0C) / Firmware

Command Description

The GU command causes the target device to leave active background and begin program execution at the instruction pointed to by the current value of the Program Counter. For enhanced BDM modules, an ACK handshake pulse is issued when the part reenters active background mode if the ACK protocol has been previously enabled.

Note: The GO_UNTIL command is only recognized by the Enhanced BDM module. The command will be ignored by a Standard BDM module.

Example

```
?GU
?
```

A.10 PC - Read or Write The Program Counter

Command Line Format

PC [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the Program Counter.

BDM Command Name/Type

READ_PC ($63) or WRITE_PC ($43) / Firmware

Command Description

The PC command is used to read or write the value of the target's Program Counter. Entering the PC command without the <Data16> parameter causes the current value of the Program Counter to be displayed by sending the READ_PC BDM command to the target. Supplying the optional <Data16> parameter results in the modification of the target's Program Counter by sending the WRITE_PC BDM command.

Example

```
?PC
PC: C13A
?PC f000
?PC
PC: F000
?
```

A.11 RB - Read a Byte From Target Memory

Command Line Format

RB <Address16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to read a byte of data.

BDM Command Name/Type

READ_BYTE ($E0) / Hardware

Command Description

The RB command is used to read a byte of data from the target's memory. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been read from memory.

Example

```
?rb c000
C000: CF
?
```

A.12 RBB - Read a Byte From Target Memory With BDM ROM & Registers in Map

Command Line Format

RBB <Address16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to read a byte of data.

BDM Command Name/Type

READ_BD_BYTE ($E4) / Hardware

Command Description

The RBB command is used to read a byte of data from the target's memory with the BDM ROM and registers in the memory map ($FF00 - $FFFF) during the read cycle, thus allowing access to the BDM ROM and status and control registers. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been read from memory.

Example

```
?rbb f020
F020: 1C
?
```

A.13 RBW - Read a Word From Target Memory With BDM ROM & Registers in Map

Command Line Format

RBW <Address16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to read a word of data.

BDM Command Name/Type

READ_BD_WORD ($EC) / Hardware

Command Description

The RBW command is used to read a word of data from the target's memory with the BDM ROM and registers in the memory map ($FF00 - $FFFF) during the read cycle, thus allowing access to the BDM ROM and status and control registers. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been read from memory.

Example

```
?rbw f020
F020: 1CFF
?
```

A.14 RESET - Reset Target in Special Single Chip Mode

Command Line Format

RESET

Parameter Description

None.

BDM Command Name/Type

N/A

Command Description

The RESET command does not correspond to one of the BDM hardware or software commands, instead it is used to reset the target MCU, placing it in Special Single-chip mode.

Example

```
?reset
?
```

A.15 RMB – Read Multiple Bytes From Target Memory

Command Line Format

RMB <Address16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to read eight bytes of data.

BDM Command Name/Type

N/A

Command Description

When performing low-level debugging of a target device, especially a device that may be secured, it is conveient to be able to read multiple bytes of data with a single command rather than issuing multiple RB or RW commands. The RMB command will display eight bytes of target data beginning at <Address16>. Note that the RMB command uses the READ_WORD ($E8) hardware command so <Address16> MUST be an even address.

Example

```
?rmb 30
0030: 00 00 BF 00 00 00 00 78
?
```

A.16 RNX - Pre-increment X-index Register by 2, Read Word Pointed to By X

Command Line Format

RNX

Parameter Description

None.

BDM Command Name/Type

READ_NEXT ($62) / Firmware

Command Description

The RNX command reads a word of data from target memory. Before the read is performed, the target's X index register is incremented by two.

Example

```
?rnx
[++X]: C93F
?
```

A.17 RW - Read a Word From Target Memory

Command Line Format

RW <Address16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to read a word of data.

BDM Command Name/Type

READ_WORD ($E8) / Hardware

Command Description

The RW command is used to read a word of data from the target's memory. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been read from memory.

Example

```
?rw f020
F020: FFFF
?
```

A.18 SP - Read or Write The Stack Pointer

Command Line Format

SP [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the Stack Pointer.

BDM Command Name/Type

READ_SP ($67) or WRITE_SP ($47) / Firmware

Command Description

The SP command is used to read or write the value of the target's Stack Pointer. Entering the SP command without the <Data16> parameter causes the current value of the Stack Pointer to be displayed by sending the READ_SP BDM command to the target. Supplying the optional <Data16> parameter results in the modification of the target's Stack Pointer by sending the WRITE_SP BDM command.

Example

```
?SP
SP: 3FDC
?SP 4000
?SP
SP: 4000
?
```

A.19 SYNC - Measure BDC Clock Speed and Reinitialize The BDM Drivers

Command Line Format

SYNC

Parameter Description

None.

BDM Command Name/Type

N/A / Non-intrusive

Command Description

The SYNC command does not correspond to an actual BDM serial command, instead it uses the SYNC protocol to measure the operating frequency of the BDM module. After obtaining the BDM module operating frequency, the BDM software drivers are initialized using the measured frequency. Note that if the CLKSW bit in the BDM Status/Control register is set, the BDM module's input clock frequency is the same as the target's bus clock. Otherwise, the BDM module will be clocked by the crystal ÷ 2.

Example

```
?rbb ff01
ff01: 84
?sync
BDM Speed: 24000 KHz
?WBB 80
? rbb ff01
ff01: FF
?sync
BDM Speed: 8000 KHz
? rbb ff01
ff01: 80
?
```

A.20 T - Execute a Single Instruction at Current PC, Return to Active Background

Command Line Format

T

Parameter Description

None.

BDM Command Name/Type

TRACE1 ($10) / Firmware

Command Description

Execute one instruction at the current PC and return to active background mode. For enhanced BDM modules, an ACK handshake pulse is issued when the part returns to active background mode if the ACK protocol has been previously enabled.

Example

```
?t
?
```

A.21 TG - Enable Instruction Tagging, Begin Execution at the Current PC

Command Line Format

TG

Parameter Description

None.

BDM Command Name/Type

TAGGO ($18) / Firmware

Command Description

The TG command causes the target device to, enable instruction tagging, leave active background and begin program execution at the instruction pointed to by the current value of the Program Counter. After execution of the TG command, no further BDM communication with the target is possible as the BKGD pin is internally disconnected from the BDM module and becomes the TAGHI pin. For enhanced BDM modules, an ACK handshake pulse is NOT issued when the part leaves active background mode even if the ACK protocol has been enabled.

Example

```
?TG
?
```

A.22 WAKE - Wake Target from STOP and Place in Active Background

Command Line Format

WAKE

Parameter Description

None.

BDM Command Name/Type

N/A

Command Description

The WAKE command can be used to bring an MCU out of STOP mode and place the MCU in active background mode without resetting the target device. This capability is most useful when a module becomes unresponsive to the normal interrupt sources that cause an MCU to exit STOP mode. The WAKE command uses the open-drain relay driver connected to pin three of the of the LFBDMPGMR's target BDM header to assert a target wake-up signal (typically the *XIRQ input).

After the assertion of the target wake-up signal, the WAKE command waits a specified period of time (1 – 65535 mS) for the target's clock to startup and stabilize. It then enables the BDM ACK protocol handshaking and writes to the target's COPCTL register to set the RSBCK bit. Setting the RSBCK bit will disable the COP when the target is in active background, allowing the examination of target resources. The WAKE command then attempts to place the target in active background mode and reports the success or failure of this operation.

Once the target MCU has been successfully placed in active background mode, the EXIT command can be entered at the '`?`' prompt to return to D-Bug12XZ where the '`S>`' prompt should be displayed. At this point the standard D-Bug12XZ commands can be used to display the CPU registers and other on-chip resources.

Restrictions

The WAKE command can only be used with the S12X and 0.18μ S12 family devices.

Example

```
D-Bug12X 5.0.0b3
Copyright 1996 - 2011 Freescale Semiconductor
For Commands type "Help"


1.) Reset Target
2.) Hot Connect to Target
3.) Erase & Unsecure
4.) Enter BDM debugger
? 4

BDM Command Debugger
For Commands type "HELP"

?wake
Enter Target BDM Speed (kHz): 4000
Enter Clock Startup Delay (mS): 50
Wake Target From STOP (y/n)? y

Target Device in Active Background Mode
?rbb ff01
FF01: C0
?exit
S>rd

PP  PC    SP    X     Y    D = A:B   CCR = IPL SXHI NZVC  GP RP EP
FE C019  20F3  C01D  0000     05:01         0  0101 0000  00 FD FE
xx:C019  20FE              BRA   $C019
S>md 3c

0030  00 00 BF 00 - 00 00 00 78 - 00 20 C1 00 - 45 00 00 00  .......x. ..E...
S>
```

A.23 WB - Write a Byte to Target Memory

Command Line Format

WB <Address16> <Data8>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to write a byte of data.

<Data8> An 8-bit hexadecimal number representing the data to be written to <Address16>

BDM Command Name/Type

WRITE_BYTE ($C0) / Hardware

Command Description

The WB command is used to write a byte of data from the target's memory. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been written to memory, indicating that the next BDM command may be sent.

Example

```
?wb 1000 16
?
```

A.24 WBB - Write a Byte to Target Memory with BDM ROM & Registers in Map

Command Line Format

WBB <Address16> <Data8>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to write a byte of data.

<Data8> An 8-bit hexadecimal number representing the data to be written to <Address16>

BDM Command Name/Type

WRITE_BD_BYTE ($C4) / Hardware

Command Description

The WBB command is used to write a byte of data from the target's memory with the BDM ROM and registers in the memory map ($FF00 - $FFFF) during the read cycle, thus allowing the BDM control registers to be altered. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been written to target memory, indicating that the next BDM command may be sent.

Example

```
?wbb FF01 84
?
```

A.25 WBW - Write a Word to Target Memory with BDM ROM & Registers in Map

Command Line Format

WBW <Address16> <Data16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to write a word of data.

<Data16> A 16-bit hexadecimal number representing the data to be written to <Address16>

BDM Command Name/Type

WRITE_BD_WORD ($CC) / Hardware

Command Description

The WBW command is used to write a byte of data from the target's memory with the BDM ROM and registers in the memory map ($FF00 - $FFFF) during the read cycle, thus allowing the BDM control registers to be altered. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been written to memory, indicating that the next BDM command may be sent.

Example

```
?wbw 1000 1617
?
```

A.26 WW - Write a Word to Target Memory

Command Line Format

WW <Address16> <Data16>

Parameter Description

<Address16> A 16-bit hexadecimal number representing the address at which to write a word of data.

<Data16> A 16-bit hexadecimal number representing the data to be written to <Address16>

BDM Command Name/Type

WRITE_BD_WORD ($CC) / Hardware

Command Description

The WW command is used to write a byte of data from the target's memory. Because this is a Hardware command, it may be executed even if the target is not in active background mode. For enhanced BDM modules, if handshaking has been enabled, an ACK handshake pulse is issued when the data has been written to target memory, indicating that the next BDM command may be sent.

Example

```
?ww 1000 1617
?
```

A.27 WNX - Pre-increment X-index Register by 2, Write the Word Pointed to By X

Command Line Format

WNX <Data16>

Parameter Description

<Data16> A 16-bit hexadecimal number representing the data to be written to [++X]

BDM Command Name/Type

WRITE_NEXT ($42) / Firmware

Command Description

The WNX command writes a word of data to target memory. Before the write is performed, the target's X index register is incremented by two.

Example

```
?wnx 1234
?
```

A.28 X - Read or Write the X-index Register

Command Line Format

X [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the X index register.

BDM Command Name/Type

READ_X ($65) or WRITE_X ($45) / Firmware

Command Description

The X command is used to read or write the value of the target's X index register. Entering the command without the <Data16> parameter causes the current value of the X index register to be displayed by sending the READ_X BDM command to the target. Supplying the optional <Data16> parameter results in the value of the target's X index register being modified by sending the WRITE_X BDM command.

Example

```
?X
X: C05A
?X 5678
?X
X: 5678
?
```

A.29 Y - Read or Write the Y-index Register

Command Line Format

Y [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the Y index register.

BDM Command Name/Type

READ_Y ($66) or WRITE_Y ($46) / Firmware

Command Description

The Y command is used to read or write the value of the target's Y index register. Entering the command without the <Data16> parameter causes the current value of the Y index register to be displayed by sending the READ_Y BDM command to the target. Supplying the optional <Data16> parameter results in the value of the target's Y index register being changed by sending the WRITE_Y BDM command.

Example

```
?Y
Y: DAB3
?Y 1A8B
?Y
Y: 1A8B
?
```

Appendix B

B.1 S12Z BDC Debugger

The S12Z BDM debugger contained within D-Bug12XZ is intended to be used by tool developers or factory engineers when evaluating new silicon or debugging BDC communication problems, however, it may also be useful to end customers. Using D-Bug12XZ's low level BDC driver routines, the BDC debugger allows individual BDC commands to be sent to a target device directly from the command line. Unlike D-Bug12XZ, which maintains constant communications with a connected target, no BDC communication occurs other than during the execution of a command. Also note that the BDC debugger does not perform any checks to ensure a target is connected before executing a command.

The BDC debugger can be entered either from D-Bug12XZ's command line (see the BDMDB command description) or from the startup menu (option 4) if communication cannot be established with a target. The BDC debugger BSPEED command can be used to (re)initialize the driver routines for an alternate BDC operating frequency if necessary. Note that the command line argument to the BSPEED command is the input frequency to the target's BDC module. Currently available MCUs containing an S12Z CPU utilize the S12 Clock, Reset and Power Management Unit (CPMU). The CPMU contains an internal 1 MHz reference clock (IRCCLK) which is used as the input clock to the BDC module (BDCCLK).

Alternately, the BDC Debugger's SYNC command can be used to measure the target BDC speed and initialize the BDC drivers.

The following list summarizes the BDM debugger command set for the S12Z devices. Each command's function and command line syntax are described in detail.

- ACKDI – Disable ACK Handshake protocol.
- ACKEN – Enable ACK Handshake protocol.
- BDCCSR – Read/Write BDC Status & Control Register.
- BKGD – Place target in active background mode.
- BSPEED – Reinitialize BDM drivers to clock speed of target device.
- CCR – Read/Write Condition Code Register.
- Dn – Read/Write Data Register Dn; n = 0 – 7.
- DM – Dump Memory At Next Sequential Address.
- DMWS – Dump Memory At Next Sequential Address & Display Status.
- EF – Mass Erase ALL of Internal Flash.
- EXIT – Exit BDM debugger.
- FM – Fill Memory At Next Sequential Address.
- FMWS – Fill Memory At Next Sequential Address & Display Status.
- G – Begin Program Execution At Current PC.
- GU – Begin Program Execution At Current PC.
- NOP – No Operation.
- PC – Read/Write Program Counter.
- RESET – Reset target in special single chip mode.
- RM – Read Memory.
- RMWS – Read Memory & Display Status.
- RTB – Read 64-bits of DBG trace buffer.
- RS – Read Same Memory Location As Last RM Command.
- RSWS – Read Same Memory Location As Last RM Command & Display Status.
- SYNC – Measure and display the target BDM clock.
- SP – Read/Write Stack Pointer.
- SYNCPC – Display Current Value of PC.
- WM – Write Memory.
- WMWS – Write Memory & Display Status.
- X – Read/Write X-index Register.
- Y - Read/Write Y-index Register.

B.2 ACKDI - Disable ACK Handshake Protocol

Command Line Format

ACKDI

Parameter Description

None.

BDM Command Name/Type

ACK_DISABLE ($03) / Always Available

Command Description

The ACKDI command disables the ACK handshake protocol if it was previously enabled. An ACK handshake pulse is not issued at the completion of the command.

Example

```
?ACKDI
?
```

B.3 ACKEN - Enable ACK Handshake Protocol

Command Line Format

ACKEN

Parameter Description

None.

BDM Command Name/Type

ACK_ENABLE ($02) / Always Available

Command Description

The ACKEN command enables the ACK handshake protocol if it was previously disabled. An ACK handshake pulse is issued at the completion of the command.

Note: The ACK_ENABLE command is only recognized by the Enhanced BDM module. The command will be ignored by a Standard BDM module.

Example

```
?ACKEN
?
```

B.4 BDCCSR – Read/Write the BDC Control and Status Register

Command Line Format

BDCCSR [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the BDCCSR register.

BDM Command Name/Type

READ_BDCCSR ($2D) or WRITE_BDCCSR ($0D) / Always Available

Command Description

The BDCCSR command is used to read or write the BDC Control and Status Register. Entering the command without the <Data16> parameter causes the current value of the BDC Control and Status Register register to be displayed by sending the READ_BDCCSR BDC command to the target. Supplying the optional <Data16> parameter writes the supplied value to the BDC Control and Status Register by sending the WRITE_BDCCSR BDC command.

Example

```
?bdccsr
BDCCSR: C200
?
```

B.5 BKGD - Place Target in Active Background Mode

Command Line Format

BKGD

Parameter Description

None.

BDM Command Name/Type

BACKGROUND ($04) / Non-intrusive

Command Description

The BKGD command causes the target device to enter active background if firmware is enabled (ENBDM = 1). An ACK handshake pulse is issued when the part enters active background mode if the ACK protocol has been previously enabled.

Example

```
?BKGD
?
```

B.6 BSPEED - Reinitialize BDM Drivers to Clock Speed of Target Device

Command Line Format

BSPEED <BDMFreq>

Parameter Description

<BDMFreq> A 16-bit decimal number representing BDM clock frequency in KHz.

BDM Command Name/Type

N/A

Command Description

The BSPEED command does not correspond to one of the BDM hardware or software commands, instead the <BDMFreq> parameter is used to reinitialize the low-level drivers to match the timing of the target's BDM. The <BDMFreq> command line argument specifies the input frequency to the target's BDM module. If the CLKSW bit in the target's BDM status register is clear, the BDM module's input clock frequency is "BDCCLK". Please refer to the specific target MCU's Reference Manual to determine the source of this clock. If the CLKSW bit is set, the BDM module's input clock frequency is the same as the target's bus clock.

Example

```
?BSPEED 1000
?
```

B.7 CCR – Read/Write the CPU12Z Condition Code Register

Command Line Format

CCR [<Data16>]

Parameter Description

<Data16> A 16-bit hexadecimal number representing the new value of the CCR register.

BDM Command Name/Type

READ_Rn ($6C) or WRITE_Rn ($4C) / Active Background

Command Description

The CCR command is used to read or write the CPU12Z Condition Code Register (CCR). Entering the command without the <Data16> parameter causes the current value of the CCR to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional <Data16> parameter writes the supplied value to the CCR by sending the WRITE_Rn command.

Example

```
?ccr
CCR: 00D0
?ccr 00d1
?ccr
CCR: 00D1
?
```

B.8 Dn – Read/Write CPU12Z Data Register Dn; n = 0 - 7

Command Line Format

Dn [<Data8> | <Data16> | <Data32>]

Parameter Description

<Data8> A 8-bit hexadecimal number representing the new value of data register D0 or D1.

<Data16> A 16-bit hexadecimal number representing the new value of data register D2, D3, D4 or D5.

<Data32> A 32-bit hexadecimal number representing the new value of data register D6 or D7.

BDM Command Name/Type

READ_Rn ($60 + CRN) or WRITE_Rn ($40 + CRN) / Active Background

Command Description

The Dn command is used to read or write the CPU12Z Data Registers. Entering the command without a parameter causes the current value of the sspecified data register to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional parameter writes the supplied value to the data register by sending the WRITE_Rn command.

Example

```
?D0
D0: 00
?D0 aa
?d0
D0: AA
?d2
D2: 0000
?d2 aa55
?d2
D2: AA55
?d6
D6: 00000000
?d6 12345678
?d6
D6: 12345678
?
```

B.9 DM – Dump Memory At Next Sequential Address

Command Line Format

DM.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

DUMP_MEM.sz ($32 + (4 * sz)) / Non-intrusive

Command Description

The DM command is used to display sequential bytes, words or long words of memory. The DM command **must** be preceeded by an initial RM or RMWS command to set the base address of sequential accesses. Thereafter, a DM command may be followed by additional DM or DMWS commands. The initial address, supplied by the RM or RMWS command, is incremented by the operand size (1, 2 or 4).

Example

```
?rm.w 1000
001000: 1234
?dm.w
Data: 5678
?dm.w
Data: 9ABC
?
```

B.10 DMWS – Dump Memory At Next Sequential Address & Display Status

Command Line Format

DMWS.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

DUMP_MEM.sz_WS ($33 + (4 * sz)) / Non-intrusive

Command Description

The DMWS command is used to display sequential bytes, words or long words of memory. In addition to displaying data at the next sequential address, the low byte of the BDCCSR is displayed. The BDCCSRL byte reflects the success or failure of the memory read operation. Refer to the Background Debug Controller documentation for a description of the various bits in the BDCCSRL. The DMWS command **must** be preceeded by an initial RM or RMWS command to set the base address of sequential accesses. Thereafter, a DMWS command may be followed by additional DM or DMWS commands. The initial address, supplied by the RM or RMWS command, is incremented by the operand size (1, 2 or 4).

Example

```
?rm.w 1000
001000: 1234
?dmws.w
Data: 5678
Status: 00
?dmws.l
Data: 9ABCDEF0
Status: 00
?
```

B.11 EF – Mass Erase ALL of Internal Flash

Command Line Format

EF

Parameter Description

None.

BDM Command Name/Type

ERASE_FLASH ($95) / Always Available

Command Description

The EF command is used to mass erase all internal Flash using the BDC ERASE_FLASH command. As required by the BCD controller, the ERASE_FLASH command is issued twice in succession. However, as shown in the example, prior to issuing the ERASE_FLASH command, a prompt is issued asking for confirmation of the mass erase operation. After issuing the ERASE_FLASH command twice, D-Bug12XZ waits until the erase operation is complete before returning to the '?' prompt. No checking is performed to ensure the mass erase operation completed without error.

Note: The ERASE_FLASH command requires the default device bus clock frequency after reset. Thus the BDM Debugger RESET command should be executed prior to issuing the EF command.

Example

```
?reset
Target CPU Has Been Reset
?ef
Are you sure you want to erase target Flash? (y/n) y
?
```

B.12 EXIT - Exit The BDM Debugger And (re)enter D-Bug12XZ

Command Line Format

EXIT

Parameter Description

None.

BDM Command Name/Type

N/A

Command Description

The EXIT command does not correspond to one of the BDM hardware or software commands, instead it is used to leave the BDM debugger and return to D-Bug12XZ. If BDM communications was established with a target in the BDM debugger, D-Bug12XZ will display the ‘S>’ or ‘R>’ prompt, otherwise the ‘Can't Communicate With Target CPU’ menu will be displayed.

Example

```
?EXIT
S>
```

B.13 FM – Fill Memory At Next Sequential Address

Command Line Format

FM.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

FILL_MEM.sz ($12 + (4 * sz)) / Non-intrusive

Command Description

The FM command is used to fill sequential bytes, words or long words of memory. The FM command **must** be preceeded by an initial WM or WMWS command to set the base address of sequential accesses. Thereafter, a FM command may be followed by additional FM commands. The initial address, supplied by the FM or FMWS command, is incremented by the operand size (1, 2 or 4).

Example

```
?wm.w 1000 1234
?fm.w 5678
?fm.l 9abcdef0
?rm.l 1000
001000: 12345678
?dm.l
Data: 9ABCDEF0
?
```

B.14 FMWS – Fill Memory At Next Sequential Address & Display Status

Command Line Format

FMWS.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

FILL_MEM.sz_WS ($13 + (4 * sz)) / Non-intrusive

Command Description

The FMWS command is used to fill sequential bytes, words or long words of memory. In addition to writing data to the next sequential address, the low byte of the BDCCSR is displayed. The BDCCSRL byte reflects the success or failure of the memory write operation. Refer to the Background Debug Controller documentation for a description of the various bits in the BDCCSRL. The FMWS command **must** be preceeded by an initial WM or WMWS command to set the base address of the next sequential accesses. Thereafter, a FMWS command may be followed by additional FM of FMWS commands. The initial address, supplied by the WM command, is incremented by the operand size (1, 2 or 4).

Example

```
?wm.w 1000 1234
?fm.w 5678
?fm.l 9abcdef0
?rm.l 1000
001000: 12345678
?dm.l
Data: 9ABCDEF0
?
```

B.15 G - Exit Active Background, Begin Target Execution at The Current PC

Command Line Format

G

Parameter Description

None.

BDM Command Name/Type

GO ($08) / Non-intrusive

Command Description

The G command causes the target device to exit active background and begin program execution at the instruction pointed to by the current value of the Program Counter. The CPU pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC. If any register (such as the PC) is altered by a BDC command while in active background mode, the updated value is used when prefetching resumes. If enabled, an ACK is driven on exiting active background mode.

If the G command is executed when the MCU is not in active background, an illegal command response is returned and the ILLCMD bit in the BDCCSR register is set.

Example

```
?G
?
```

B.16 GU - Exit Active Background, Begin Target Execution at The Current PC Until

Command Line Format

GU

Parameter Description

None.

BDM Command Name/Type

GO_UNTIL ($0C) / Firmware

Command Description

The GU command causes the target device to exit active background and begin program execution at the instruction pointed to by the current value of the Program Counter. The CPU pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC. If any register (such as the PC) is altered by a BDC command while in active background mode, the updated value is used when prefetching resumes.

If enabled, an ACK is driven on the BGND pin when the CPU12Z reenters active background mode. If ACK handshaking is disabled, the GU command is identical to the G command.

Example

```
?GU
?
```

B.17 NOP – No Operation

Command Line Format

NOP

Parameter Description

None.

BDM Command Name/Type

NOP ($00) / Non-Intrusive

Command Description

NOP performs no operation and may be used as a null command where required.

Example

```
?NOP
?
```

B.18 PC – Read/Write CPU12Z Program Counter

Command Line Format

PC [<Address24>]

Parameter Description

<Address24> A 24-bit hexadecimal number representing the new value of the Program Counter.

BDM Command Name/Type

READ_Rn ($6B) or WRITE_Rn ($4B) / Active Background

Command Description

The PC command is used to read or write the CPU12Z Program Counter. Entering the command without a parameter causes the current value of the Program Counter to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional parameter writes the supplied value to the Program Counter by sending the WRITE_Rn command.

Example

```
?pc fe0000
?pc
PC: FE0000
?
```

B.19 RESET - Reset Target in Special Single Chip Mode

Command Line Format

RESET

Parameter Description

None.

BDM Command Name/Type

N/A

Command Description

The RESET command does not correspond to one of the BDM hardware or software commands, instead it is used to reset the target MCU, placing it in Special Single-chip mode.

Example

```
?reset
?
```

B.20 RM – Read Memory

Command Line Format

DM.<sz> <Address24>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.
<Address24> A 24-bit hexadecimal number representing the target address at which to read data.

BDM Command Name/Type

READ_MEM.sz ($30 + (4 * sz)) / Non-intrusive

Command Description

The RM command is used to read bytes, words or long words of memory. The hardware forces low-order address bits to zero for long word accesses to ensure these accesses are on 0- modulo-size alignments.

Example

```
?rm.w 1000
001000: 1234
?rm.w 1002
001002: 5678
?rm.b 1003
001003: 78
?
```

B.21 RMWS – Read Memory & Display Status

Command Line Format

RMWS.<sz> <Address24>

Parameter Description

<sz>	"B" = Byte, "W" = Word, "L" = Long.
<Address24>	A 24-bit hexadecimal number representing the target address at which to read data.

BDM Command Name/Type

READ_MEM.sz_WS ($31 + (4 * sz)) / Non-intrusive

Command Description

The RMWS command is used to display sequential bytes, words or long words of memory. In addition to displaying data at the next sequential address, the low byte of the BDCCSR is displayed. The BDCCSRL byte reflects the success or failure of the memory read operation. Refer to the Background Debug Controller documentation for a description of the various bits in the BDCCSRL.

Example

```
?rmws.w 1000
001000: 1234
Status: 00
?rmws.w 1002
001002: 5678
Status: 00
?rmws.l 1004
Data: 9ABCDEF0
Status: 00
?
```

B.22 RTB – Read 64-bits of DBG trace buffer

Command Line Format

RTB

Parameter Description

None.

BDM Command Name/Type

READ_DBGTB ($07) / Non-intrusive

Command Description

The RMWS command is used to read 64 bits of data from the DBG trace buffer. The first 32-bit long word corresponds to trace buffer line bits[31:0]; the second to trace buffer line bits[63:32]. Refer to the DBG module description for more detailed information.

Example

```
?rtb
EEEEEEEE:EEEEEEEE
?rtb
EEEEEEEE:EEEEEEEE
?
```

B.23 RS – Read Same Memory Location As Last RM Command

Command Line Format

RS.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

READ_SAME.sz ($50 + (4 * sz)) / Non-intrusive

Command Description

The RS command is used to display memory bytes, words or long words using the same address as a previous RM or RMWS command. The RS command **must** be preceeded by an initial RM or RMWS command to set the base address of the memory access. Thereafter, an RS command may be followed by additional RS or RSWS commands. The initial address, supplied by the RM or RMWS command, is incremented by the operand size (1, 2 or 4).

Example

```
?rm.w 1000
001000: 1234
?rs.w
Data: 1234
?rs.w
Data: 1234
?
```

B.24 RSWS – Read Same Memory Location As Last RM Command & Display Status

Command Line Format

RSWS.<sz>

Parameter Description

<sz> "B" = Byte, "W" = Word, "L" = Long.

BDM Command Name/Type

DUMP_MEM.sz_WS ($51 + (4 * sz)) / Non-intrusive

Command Description

The RSWS command is used to display memory bytes, words or long words using the same address as a previous RM or RMWS command. In addition to displaying data at the same address, the low byte of the BDCCSR is displayed. The BDCCSRL byte reflects the success or failure of the memory read operation. Refer to the Background Debug Controller documentation for a description of the various bits in the BDCCSRL. The RSWS command **must** be preceeded by an initial RM or RMWS command to set the base address of memory access. Thereafter, an RSWS command may be followed by additional RS or RSWS commands. The initial address, supplied by the RM or RMWS command, is incremented by the operand size (1, 2 or 4).

Example

```
?
```

B.25 SP – Read/Write CPU12Z Stack Pointer

Command Line Format

SP [<Address24>]

Parameter Description

<Address24> A 24-bit hexadecimal number representing the new value of the Stack Pointer.

BDM Command Name/Type

READ_Rn ($6A) or WRITE_Rn ($4A) / Active Background

Command Description

The SP command is used to read or write the CPU12Z Stack Pointer. Entering the command without a parameter causes the current value of the Stack Pointer to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional parameter writes the supplied value to the Stack Pointer by sending the WRITE_Rn command.

Example

```
?sp 3000
?sp
SP: 003000
?
```

B.26 SYNC - Measure BDC Clock Speed and Reinitialize The BDC Drivers

Command Line Format

SYNC

Parameter Description

None.

BDM Command Name/Type

N/A / Non-intrusive

Command Description

The SYNC command does not correspond to an actual BDC serial command, instead it uses the SYNC protocol to measure the operating frequency of the BDC module. After obtaining the BDC module operating frequency, the BDC software drivers are initialized using the measured frequency. Note that if the CLKSW bit in the BDC Status/Control register is set, the BDC module's input clock frequency is the same as the target's bus clock. Otherwise, the BDC module will be clocked by IRCCLK.

Example

```
?sync
BDC Clock Frequency: 998 KHz
?BDCCSR
BDCCSR: C200
?bdccsr c600
?bdccsr
BDCCSR: FEFE
?sync
BDC Clock Frequency: 6232 KHz
?bdccsr
BDCCSR: C600
?
```

B.27 SYNCPC - Display Current Value of Program Counter

Command Line Format

SYNCPC

Parameter Description

None.

BDM Command Name/Type

SYNC_PC ($01) / Non-intrusive

Command Description

The SYNCPC command is used to display the current contents of the 24-bit Program Counter. This command can be used to dynamically access the PC for performance monitoring as the execution of this command is considerably less intrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence.

While the BDC is not in active BDM (i.e. it is executing the target application), SYNC_PC returns the PC address of the instruction currently being executed by the CPU. In active BDM, SYNC_PC returns the address of the next instruction to be executed on returning from active BDM. Thus following a write to the PC in active BDM, a SYNC_PC returns that written value.

Example

```
?syncpc
PC: 001502
?syncpc
PC: 002003
?syncpc
PC: 002002
?syncpc
PC: 002002
?syncpc
PC: 001002
?
```

A.28 T - Execute a Single Instruction at Current PC, Return to Active Background

Command Line Format

T

Parameter Description

None.

BDM Command Name/Type

STEP1 ($09) / Firmware

Command Description

The T command is used to execute a single instruction at the current PC location. In active background mode this command executes the next instruction. If enabled an ACK is driven. If the T command is issued and the CPU is not halted, the command is ignored.

Example

```
?t
?
```

B.29 WM – Write Memory

Command Line Format

WM.<sz> <Address24> <DataSz>

Parameter Description

<sz>	"B" = Byte, "W" = Word, "L" = Long.
<Address24>	A 24-bit hexadecimal number representing the target address at which to read data.
<DataSz>	An 8-, 16- or 32-bit value depending on <sz>.

BDM Command Name/Type

WRITE_MEM.sz ($10 + (4 * sz)) / Non-intrusive

Command Description

The WM command is used to write bytes, words or long words to memory. The hardware forces low-order address bits to zero for long word accesses to ensure these accesses are on 0- modulo-size alignments.

Example

```
?wm.b 1000 01
?wm.w 1001 2345
?wm.b 1003 67
?wm.l 1004 89abcdef
?rm.l 1000
001000: 01234567
?rm.l 1004
001004: 89ABCDEF
?
```

B.30 WMWS – Write Memory & Display Status

Command Line Format

WMWS.<sz> <Address24> <DataSz>

Parameter Description

<sz>	"B" = Byte, "W" = Word, "L" = Long.
<Address24>	A 24-bit hexadecimal number representing the target address at which to read data.
<DataSz>	An 8-, 16- or 32-bit hexadecimal value depending on <sz>.

BDM Command Name/Type

WRITE_MEM.sz_WS ($11 + (4 * sz)) / Non-intrusive

Command Description

The WMWS command is used to write bytes, words or long words to memory. In addition to displaying data at the next sequential address, the low byte of the BDCCSR is displayed. The BDCCSRL byte reflects the success or failure of the memory read operation. Refer to the Background Debug Controller documentation for a description of the various bits in the BDCCSRL.

Example

```
?wmws.b 1000 01
Status: 00
?wmws.w 1001 2345
Status: 00
?wm.b 1003 67
Status: 00
?wm.l 1004 89abcdef
Status: 00
?rmws.w 1000
001000: 0123
Status: 00
?rmws.w 1002
001002: 4567
Status: 00
?rmws.l 1004
Data: 89ABCSEF
Status: 00
?
```

B.31 X – Read/Write CPU12Z X Index Register

Command Line Format

X [<Address24>]

Parameter Description

<Address24> A 24-bit hexadecimal number representing the new value of the X Index Register.

BDM Command Name/Type

READ_Rn ($68) or WRITE_Rn ($48) / Active Background

Command Description

The X command is used to read or write the CPU12Z X Index Register. Entering the command without a parameter causes the current value of the X Index Register to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional parameter writes the supplied value to the X Index Register by sending the WRITE_Rn command.

Example

```
?x fe7000
?x
X: FE7000
?
```

B.32 Y – Read/Write CPU12Z Y Index Register

Command Line Format

Y [<Address24>]

Parameter Description

<Address24> A 24-bit hexadecimal number representing the new value of the Y Index Register.

BDM Command Name/Type

READ_Rn ($69) or WRITE_Rn ($49) / Active Background

Command Description

The Y command is used to read or write the CPU12Z Y Index Register. Entering the command without a parameter causes the current value of the Y Index Register to be displayed by sending the READ_Rn BDC command to the target. Supplying the optional parameter writes the supplied value to the Y Index Register by sending the WRITE_Rn command.

Example

```
?y fe8000
?y
Y: FE8000
?
```

Appendix C

C.1 D-Bug12XZ Hardware Requirements

The D-Bug12XZ firmware was designed to run on the LFBDMPGMR hardware, however, the firmware can be run on other hardware. The LFBDMPGMR design uses the S12XEP100, however, D-Bug12XZ can also run on the S12XEQ512 or S12XEP768. D-Bug12XZ uses SCI0 for communication to a host terminal program. The RxD pin of SCI0 must be connected to port pin PT0 in order to support D-Bug12X's auto-baud feature.

Pin PAD11/AN11 must be connected to the center of a 2:1 voltage divider network (the LFBDMPGMR uses 2 – 10K resistors). The top of the network should be connected to pin 6 of the target BDM connector (target V_{DD}). The other end of the network connected to V_{SS}. The D-Bug12 Host CPU V_{RH} pin should be connected to 5 volts and V_{RL} should be connected to V_{SSA}.

The LFBDMPGMR uses a Maxim MAX3378E level translator connected to port pins PT6 and PT7 to allow the hardware to be compatible with both 3.3 and 5 volt systems. However, this is not required by D-Bug12X. Port pin PT7 can be connected directly to pin 1 of the target BDM connector (BGND) and port pin PT6 can be connected directly to pin 4 of the target BDM connector (Reset).

The LFBDMPGMR utilizes a 10 MHz external oscillator as its reference source. Any other hardware must provide an external 10MHz (crystal or oscillator) reference.

C.2 D-Bug12XZ Software Requirements

The LFBDMPGMR contains an 8K bootloader used to update the Programmer's firmware. The bootloader resides from $E000 - $FFFF in the local memory map ($7FE000 – $7FFFFF in the Global map). D-Bug12X's reset and interrupt vector table resides in the local map just below the bootloader from $DF10 – $DFFF. D-Bug12X's 'reset' vector at $DFFE points to the beginning of D-Bug12X's startup code.

Appendix D

D.1 Loading D-Bug12XZ Into the LFBDMPGMR Hardware

The LFBDMPGMR hardware comes preprogrammed with firmware to perform high-speed production programming of M68HC12, MC9S12 and MC9S12X family devices. However, the LFBDMPGMR hardware may also be used to run v6.x.x of the D-Bug12XZ firmware. This version of D-Bug12, developed for the MC9S12XEP100, provides an economical yet powerful debugging tool that can be utilized to debug S12, S12X, or S12Z applications through a BDM connection to a target device. While D-Bug12XZ only provides a command line interface to the user, it provides many powerful debugging features including the ability to 'hot connect' to a running target system without resetting the target device.

The following procedure should be used to load the D-Bug12XZ firmware into the BDM programmer hardware:

1. On the host computer, launch HyperTerminal, TeraTerm or other terminal emulator program and configure its communications settings for 9600 baud, eight data bits, one stop bit, XOn/XOff handshaking and no parity.

2. Connect the BDM programmer hardware to the host computer's serial port and apply power to the programmer. While pressing and holding down the 'Program Start' button, press and release the System Reset/Power button. The boot loader prompt, described in Section 3 should appear on the screen.

3. Type a lower case 'a' on the keyboard to erase the programmer's Flash memory. Erasing the Flash will take several seconds, if the Flash was successfully erased, the prompt will be redisplayed without any error messages.

4. If desired, the baud rate may be changed to reduce the programming time. See Section 2.2.4, Set Baud Rate Command, for a description of the boot loader's Set Baud Rate command.

5. Typing a lower case 'b' on the keyboard causes the boot loader to enter the programming mode, waiting for S Records to be sent from the host. Using the 'Send Text File' capability of the terminal emulator program, locate and send the file "D-Bug12XZ600bx.S19" to the programmer. Note that the 'x' in the file name represents a one or two digit revision number. As each S-Record is received and programmed into the MC9S12XEP100's Flash, an asterisk character ('*') is displayed. If the Flash was successfully programmed, the prompt will be redisplayed without any error messages.

6. Before pressing the reset button to exit the boot loader, enter a lower case 'e' on the keyboard to erase the MC9S12XEP100's on-chip EEPROM. This action will cause D-Bug12XZ to initialize the EEPROM with its data.

7. After pressing the Programmer's System Reset/Power button, press the 'Return' key on the keyboard. The D-Bug12XZ firmware will auto baud to the terminal's baud rate setting. If the programmer is connected to a target system and D-Bug12XZ is able to establish communications with the target device, the D-Bug12XZ prompt will be displayed. If communications cannot be established, D-Bug12XZ will display a menu allowing several options. Refer to Section 3.0, Initial Communication With D-Bug12XZ, for help.

8. To reload the BDM programmer firmware, follow steps 1 through 7 sending the file "BDMPgmrEP100.s19" to the programmer in step 5.