Hot Connect Debug for S12 and S12X MagniV Mixed-Signal Microcontrollers

AMF-ACC-T1658

Gordon Doughman | Field Applications Engineer
Tom Richardson | Field Applications Engineer

S E P T . 2 0 1 5

External Use

Agenda

- The Problem/Solution
- MagniV Debug Interface
- Target Connection
- The Hardware
- The Software
- Demo

The Problem

- A module in a vehicle is unresponsive to:
  - Network communications.
  - Switch inputs.

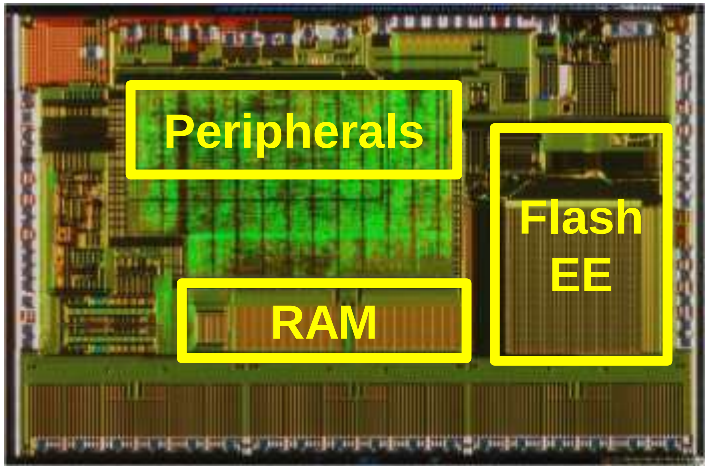
Body

- Observed on multiple vehicles?
- Problem resolves on battery disconnect.
- Issue can't be reproduced on bench.

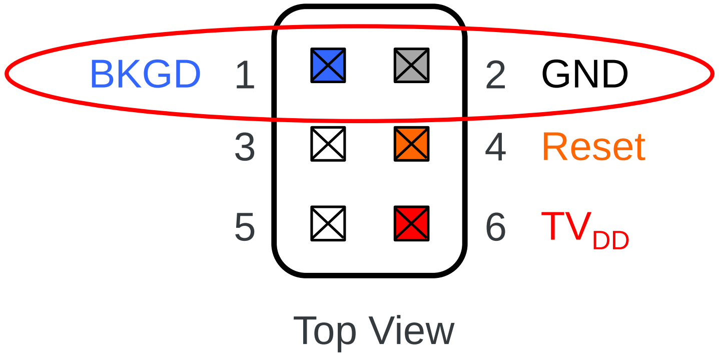
Safety

The Solution

- The BDM/BDC interface can be used to access target:
  - Peripheral Registers
  - Flash
  - EEPROM/DFlash
  - RAM
- Without disturbing the running target.



- Only two connections to the target required:
  - BKGD pin
  - V_{SS}/GND



Real World Issue

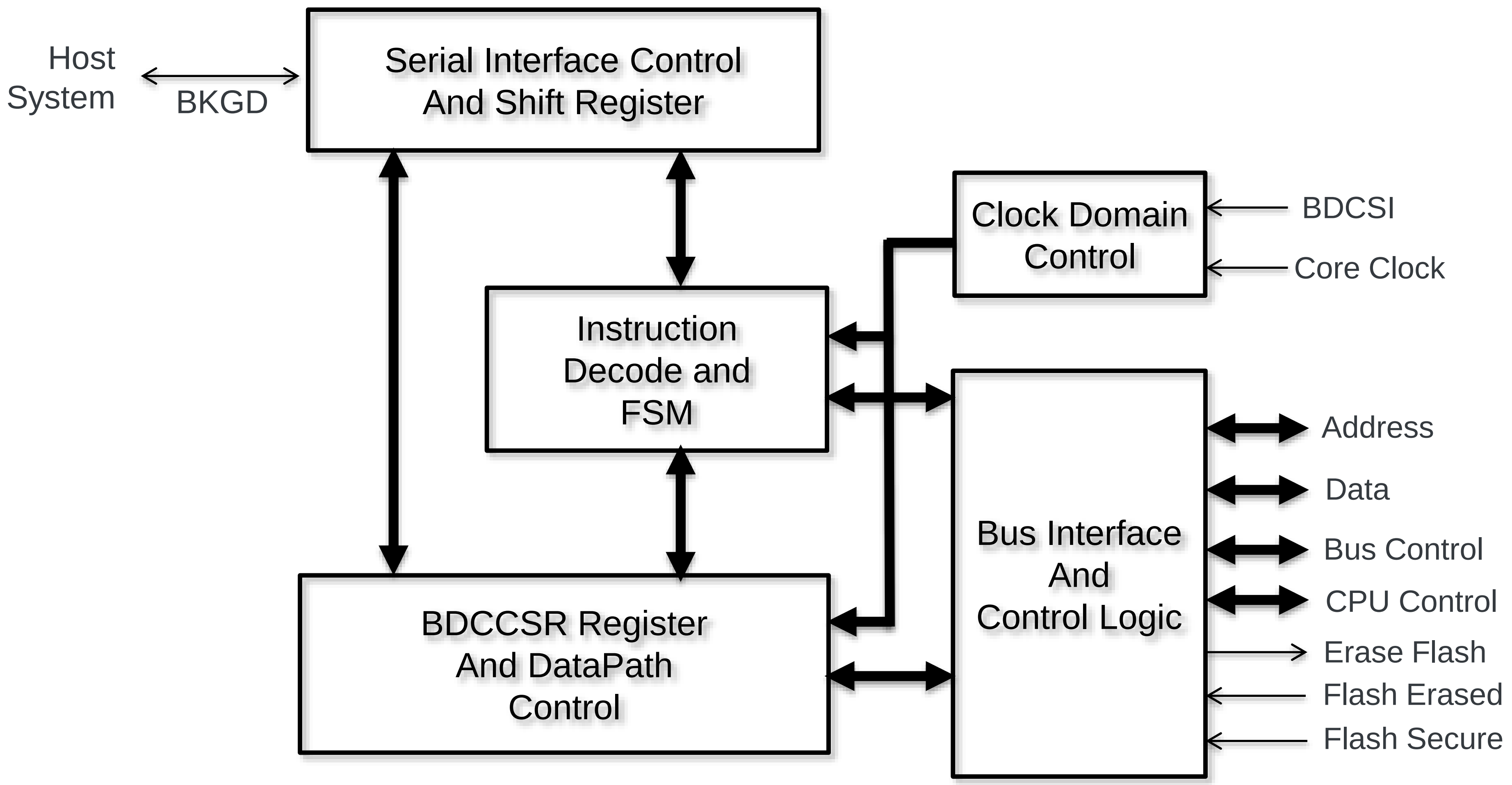
- The Body Controller on vehicles would randomly “lockup”. Recycling power restored operation. Problem was not reproducible.
- The BDM connection was made with an active module in a vehicle exhibiting the fault.
- BDM tool uploaded the Registers, RAM, and EE.
- Tool was used to stop the CPU and determine where the CPU was executing.
- The diagnosis identified exactly where the module was stuck in a loop.
- Within an hour of “Hot connecting” the BDM tool to the module, the root cause was identified as a software error.

MagniV Debug Interface

BDC/BDM Debug Interface

- Single-wire interface originally developed for HC12.
- Propagated to S08, S12, S12X, S12Z.
  - Same communication protocol, different command sets.
- Allows non-intrusive ‘DMA-style’ access to all memory mapped resources.
  - Uses ‘low’ portion of bus clock or cycle steal.
- SYNC command allows determination of communication rate.
- Three operating modes:
  - Disabled, out of reset in ‘Normal Single-chip’.
  - Enabled, only by a BDM tool.
  - Active
    - Out of reset in ‘Special Single-chip’
    - Execution of BGND instruction or breakpoint from DBG, when Enabled.
    - Forced by BDM tool after being enabled.

BDC Block Diagram


Host System
BKGD
Serial Interface Control And Shift Register
Clock Domain Control
BDCSI
Core Clock
Instruction Decode and FSM
Bus Interface And Control Logic
Address
Data
Bus Control
CPU Control
Erase Flash
Flash Erased
Flash Secure
BDCCSR Register And DataPath Control

BDC/BDM and Security

- When target device is secured:
  - BDC operation is restricted to checking BDCCSR register for secure mode.
  - Cannot access ANY on-chip resources.
    - Older BDM interface allowed access to ALL I/O registers.
  - Can perform an ‘Erase and Unsecure’ operation ONLY.
- Backdoor ‘key’ can be used to temporarily disable security.
- HOWEVER:
  - Application software must support reception of backdoor ‘key’ via communications channel.
  - Perform unlock operation
- If module is unresponsive, this is not an option.
- SO, security is a two edged sword.

BDC/BDM and Low Power Modes

- WAIT Mode (execution of WAI instruction)
  - CPU execution is halted, all on-chip clocks continue.
  - All BDC commands not involving CPU resources are allowed.
    - Principally, all memory read/write operations.
  - WAIT flag in BDCCSR is set.
  - Cannot be placed in active background while in WAIT mode.
- STOP mode (execution of STOP instruction)
  - CPU and all on-chip clocks are halted.
  - BDC communication is not possible UNLESS:
    - BDM enabled prior to entering STOP mode – allows access to BDCCSR only!
    - BDM enabled & BDCCIS set – core clocks continue to run.
      - Only useful for debugging.
- Unresponsive module in STOP mode == no data.

BDC/BDM Resources

- Single 16-bit register provides all BDC control and status functions.

BDCCSRH	R	ENBDC	BDMACT	BDCCIS	0	STEAL	CLKSW	UNSEC	ERASE
	W		▨		▨			▨	▨
BDCCSRL	R	WAIT	STOP	RAMWF	OVRUN	NORESP	RDINV	ILLACC	ILLCMD
	W								

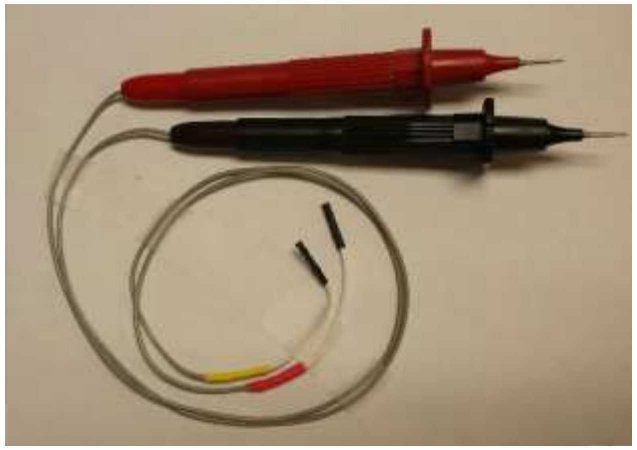
- ENBDC & BDMACT set when device reset in Special Single-chip mode.
  - BDM tools reset target device in Special Single-chip mode.
    - Hold BKGD pin low on rising edge of reset.
- BDCCSRL bits are cleared by writing ‘1’ (except OVRUV).
- Standard BDM tools do not provide direct access to this register.
- D-Bug12XZ’s low-level BDM debugger can directly read & write the BDCCSR (more on this later).

Target Connection

Target Connection

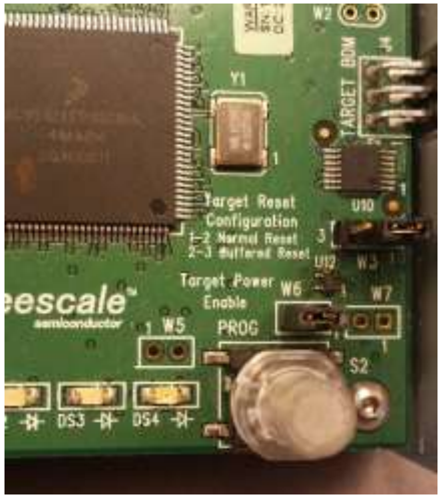
- Typical BDM connection for Programming or Debug is 6 pin ribbon cable.
- For Hot Connect, a 2 wire connection (BKGD, GND) is recommended. This minimizes chance of an accidental reset.
  - For target with BDM header installed:

  - For target with test points only:



Target Connection (cont'd)

- Configuration of FG box:
  - With a 2 wire connection, target Vdd is not present.
    - Jumper W6 (Target Power Enable) needs to be installed, to provide 5V to the internal level shifter.



    - For a target operating at 3V, instead of a jumper on W6, 2 diodes can be connected in series between the pins (1 = anode, 2 = cathode).

Target Connection (cont’d)

- Before connecting to the malfunctioning target:
  - Practice on a known good module first
  - Make sure that a terminal log file is open.
  - Observe proper ESD practices

The Hardware - LFBDMPGMR

LFBDMPGMR – aka Flash Gordon

- Designed as a BDM high speed production Flash programmer.
- Also runs D-Bug12XZ and D-BugS08 firmware for low-level debugging.
- USB & RS-232 interface to host computer.
- $500 – Can be ordered direct or through a distributor.

LFBDMPGMR – The Win7, 64-bit problem

- ‘Customized’ USB drivers only 32-bit, not digitally signed.
  - No problem for 32-bit systems.
- USB interface utilizes Silicon Labs CP2102 USB-to-Serial device.
- 64-bit drivers are available!!!
- But…
- CP2102 VID & PID were programmed with Freescale values.
  - Silicon Labs 64-bit drivers won’t recognize the LFBDMPGMR.
- But we have a solution!!!
- CP2102 VID & PID can be reprogrammed using simple procedure.
  - Instructions & software detailed in **LFBDMPGMR-64-BIT-SW** which can be downloaded from freescale.com.

LFBDMPGMR – Firmware

- The LFBDMPGMR comes with the production programming firmware installed.
- Built in bootloader allows updates or other firmware to be run.
  - Detailed instructions for loading D-Bug12XZ can be found in Appendix D of the D-Bug12XZ Reference Manual.
- Zip file containing S-Record file, Reference Manual, etc. can be found here:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=S08_S12_X_FLASHPGMR&fpsp=1&tab=Design_Tools_Tab

- “Downloads” Tab, under “Software Development Tools”
  - File name: D-BUG12XZV6FW
- Supports all currently available S12, S12X, S12XE & S12Z devices.

The Software – D-Bug12XZ

D-Bug12XZ - Overview

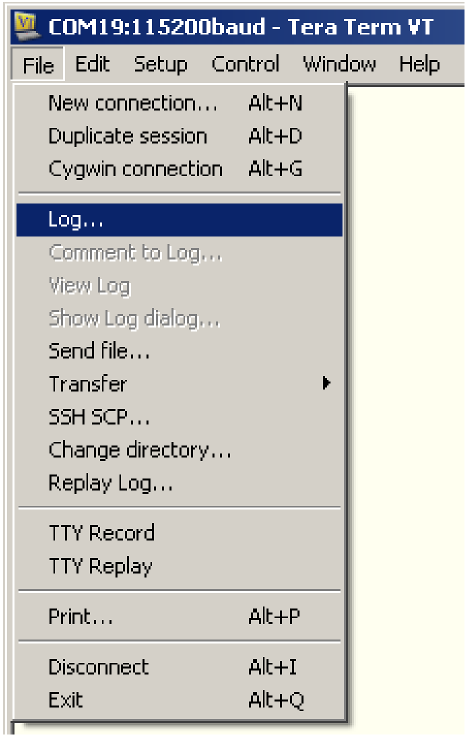
- Simple but powerful command line debugger.
  - No symbolics or source level debug.
- D-Bug12XZ is a useful tool in tracking down obscure bugs that can be masked by high level debuggers.
- Designed to put the least software complexity between developer & device.
- When a system is ‘hung up’, D-Bug12XZ can be used to ***reliably*** ‘hot connect’ to a target system.
- All memory mapped resources can be uploaded to host computer for later analysis.
- After ‘hot connect’ target can be stopped to find where the code is ‘hung’.

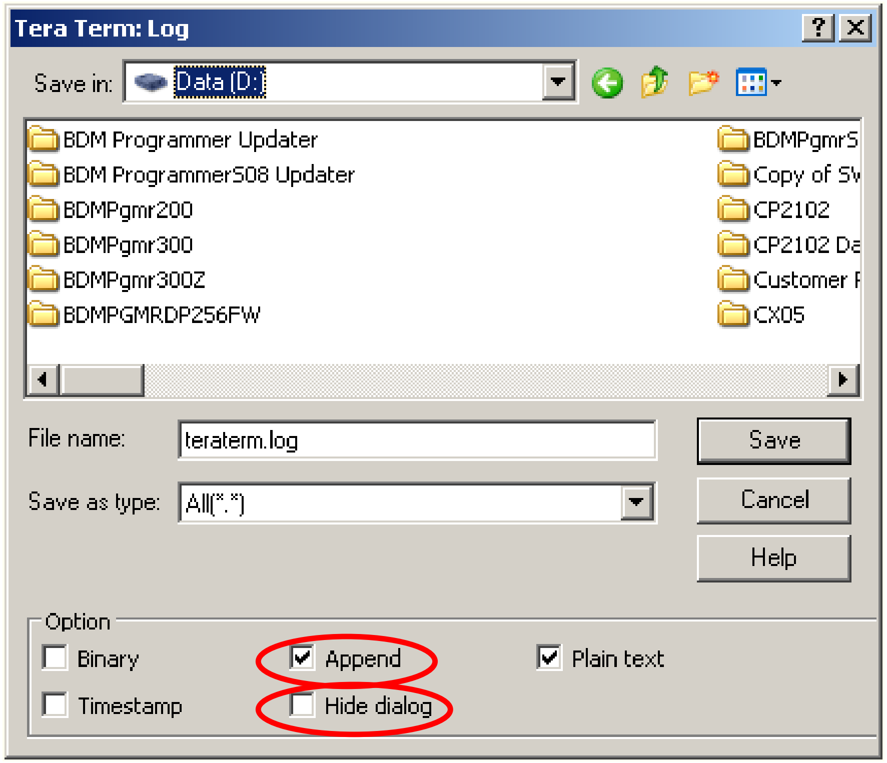
D-Bug12XZ – Host Computer Requirements

- Requires nothing more than a simple terminal emulator.
  - HyperTerm on older WinXP systems.
  - We prefer the free, open source, TeraTerm
    - http://ttssh2.osdn.jp
    - More reliable than HyperTerm (but not perfect!)
    - It's scriptable.
  - Communications settings:
    - Data bits – 8
    - Stop bits – 1
    - Parity – none
    - Handshaking – XOn/Xoff (important!)
    - Baud rate – 600 to 230,400; D-Bug12XZ, auto baud detect.
      - Recommend 115,200
    - VT100 emulation for 'LOG' command (optional).
    - Terminal width of at least 100 characters for S12Z.

D-Bug12XZ – Terminal Emulator Data Logging

- Prior to connecting the LFBDMPGMR to the target:
  - Set up the terminal emulator to log all data to a disk file.
  - For TeraTerm:





File name: teraterm.log

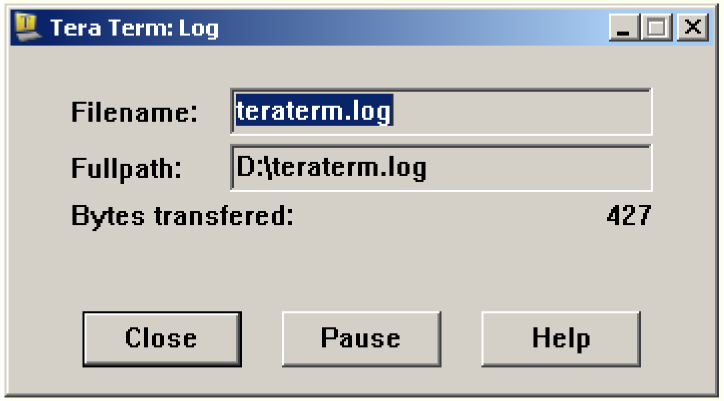
Save as type: All(*.*)

Option

- [ ] Binary
- [✓] Append
- [✓] Plain text
- [ ] Timestamp
- [ ] Hide dialog

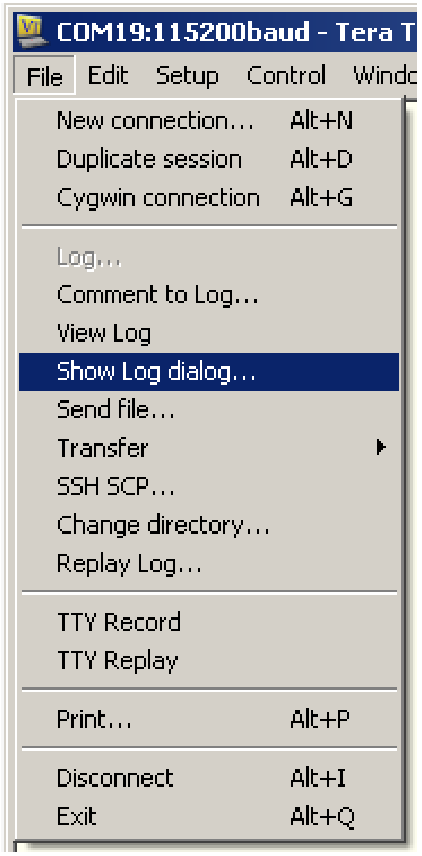
D-Bug12XZ – Terminal Emulator Data Logging

- After debug session is complete, close the log file:



Filename: teraterm.log
Fullpath: D:\teraterm.log
Bytes transfered: 427

- If the log file dialog is hidden:



D-Bug12XZ - Prompts

- After power up or resetting hardware press ‘Return/Enter’ key.
- The following prompt should be displayed:

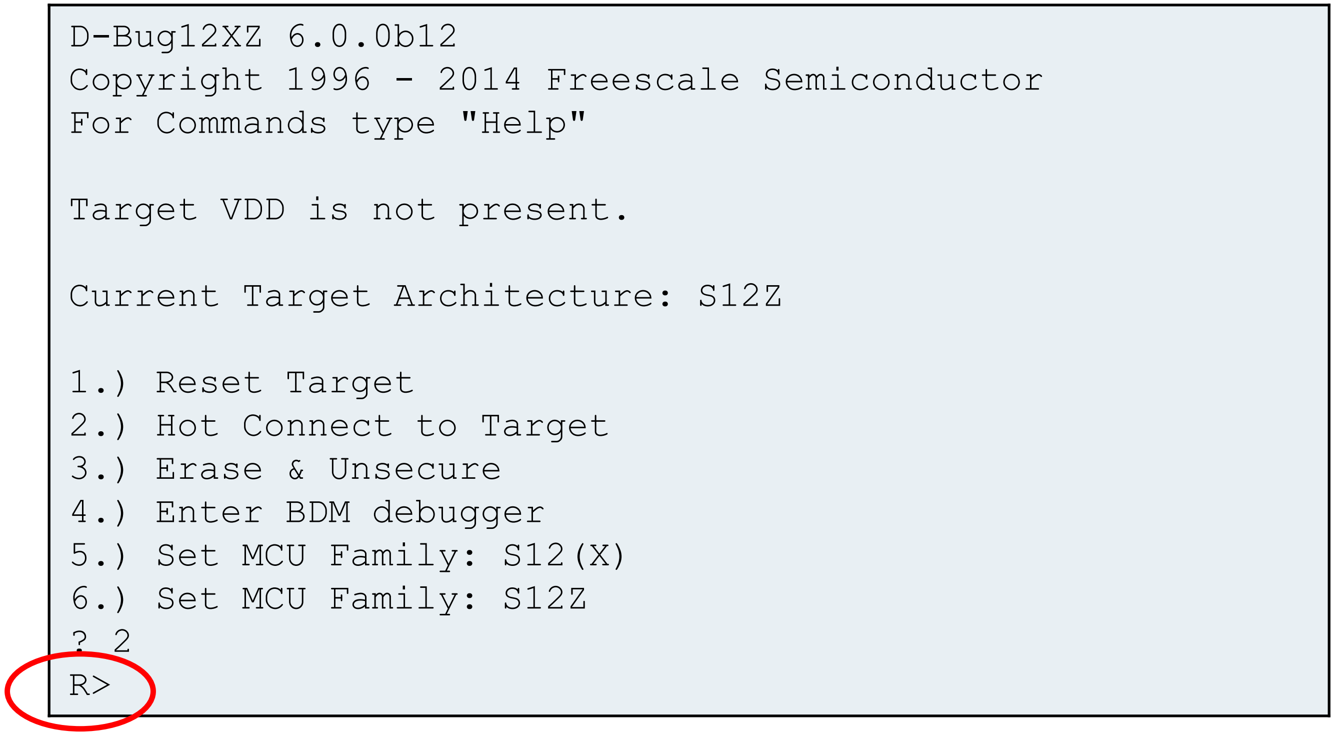
```
D-Bug12XZ 6.0.0b12
Copyright 1996 - 2015 Freescale Semiconductor
For Commands type "Help"


Target VDD is not present.

Current Target Architecture: S12Z

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
?
```

D-Bug12XZ – “Can’t Communicate…” Options

- Option #1, Reset Target
  - Resets a connected target into Special Single-chip mode.
    - **DO NOT SELECT for HOT connect!!!!!**
- Option #2, Hot Connect to Target
  - Sends SYNC command & attempts to communicate with target.
- Option #3, Erase & Unsecure
  - The only way to recover a secured target. ALL Flash & EE is lost!
- Option #4, Enter BDM Debugger
  - Enter Low-level BDM debugger (again, more later).
- Option #5 & #6, selects target architecture
  - S12(X) for ALL S12/S12X devices.
  - S12Z option for all MagniV devices with S12Z core.
    - S12VR & MM912_6xx (dual die analog) utilize S12S CPU core.

D-Bug12XZ – Hot Connect To Target

- Selecting option ‘2’ should cause the ‘`R>`’ prompt to be displayed:



```
D-Bug12XZ 6.0.0b12
Copyright 1996 - 2014 Freescale Semiconductor
For Commands type "Help"

Target VDD is not present.

Current Target Architecture: S12Z

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
R>
```

- ‘`R>`’ prompt indicates target is running application code.

D-Bug12XZ – Hot Connect To Target

- If an '`S>`' prompt is displayed:
  - MCU is in active background mode.
  - MCU was off in the 'weeds' and a BGND op-code was executed after the BDC was enabled.
  - Watchdog (internal or external) asserted during SYNC or normal BDC communications.
    - BKGD pin held low on rising edge of Reset causes entry into active background.
- If a repeated '`R>`' is displayed:
  - i.e. "`R>R>R>R>R>R>R>R>R>R>`"
  - Unstable clock
    - D-Bug12XZ temporarily loses communication and then immediately regains.
  - Watchdog is resetting target.

D-Bug12XZ – Gathering Data From The Target

- At the ‘R>’ prompt, type “device”:

```
R>device

Device: MC9S12ZVL32, MC9S12ZVL16, MC9S12ZVL8, MC9S12ZVLS32, MC9S12ZVLS16
Target PartID: 04150000
PFlash: $FF8000 - $FFFFFF (32768 bytes)
EEPROM: $100000 - $10007F (128 bytes)
RAM: $001000 - $0013FF (1024 bytes)
I/O Regs: $0000 - $0FFF
Target BDM Speed: 1000 KHz

R>
```

- Lists all derivatives (phantoms) based on the target die.
- Target PartID contains mask set info.
  - Important if can’t physically be read from device.
- All on-chip target resources, address ranges and BDC speed.

D-Bug12XZ – Memory Display Command

- MD - Display memory in hexadecimal bytes and ASCII format.
  - Displays the contents of memory in both hexadecimal bytes and ASCII.
  - 16 bytes per line
  - Syntax: MD <StartAddress> [<EndAddress>]
    - "FLASH", "DFLASH", "EE", "EEE", "IO", "RAM" can be used in place of <StartAddress> and <EndAddress>.
  - Allows easy viewing of target memory contents.
- MDW - Display memory in hexadecimal words and ASCII format.
  - Displays the contents of memory in both hexadecimal words and ASCII.
  - Same command line syntax as MD.
- For 0.25µ S12 devices, only 64K addresses may be used for <StartAddress> and <EndAddress> when '`R>`' prompt is displayed.

D-Bug12XZ – Memory Display Command Example

```
R>md io

000000  04 15 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000010  FF FE 00 00 - 00 00 00 08 - 00 00 00 00 - 00 00 00 00  ................
000020  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000030  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000040  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000050  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000060  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000070  80 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000080  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000090  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
.
.
.
000F90  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FA0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FB0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FC0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FD0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FE0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
000FF0  00 00 00 00 - 00 00 00 00 - 00 00 00 00 - 00 00 00 00  ................
R>
```

D-Bug12XZ – Upload Command

- UPLOAD – Display memory in S-Record Format.
  - Especially useful for extracting Flash, EE, EEE for comparison against known good S-Record contents.
  - Syntax: MD <StartAddress> [<EndAddress>] [;<SRecSize>]
    - Optional <SRecSize> specifies data field length of 16 – 64 bytes.
  - “FLASH”, “DFLASH”, “EE”, “EEE”, “IO”, “RAM” can be used in place of <StartAddress> and <EndAddress>.
  - ‘S9’ end-of-file record displayed after last S-Record.
- For 0.25µ S12 devices, only 64K addresses may be used for <StartAddress> and <EndAddress> when ‘`R>`’ prompt is displayed.

D-Bug12XZ – Upload Command Example

```
R>upload flash
S224FF8000B6FF803E2714B8FF8042A940A743BCF70B877E18070B867505B9FF80452710A61B
S224FF8020F7270CA8F71CF7E70B867D2074051B030010FF21FFCD21FFE3BBFF81F005000085
S224FF80400001FF8052FF86F30000FF80590000FF80590011000000013ED8002F2ECC002EF
S224FF8060C2ECD002C2ECE002C2ED9002C2EDA002C2EDB002C2051AFEBD6020049D6090014B
S224FF808090F06024790A62051AFA1D71641D72621D7360BBFF80C1C064BBFF80C1C062BB10
S224FF80A0FF80C1C060A062F0642611A060F062260B1D64F81100A0642006A0F811000A667F
S224FF80C0051AFE1C7202C021FFAFA402C0E41226101D73F81100A0F81100C0602080D2A456
S224FF80E002C0E42226101D76F81100A0F81100C0602080BDA402C0E44226101D79F811005B
S224FF8100A0F81100C0602080A81C7402C021FF69A402C0E41426101D72F81100A0F811009A
S224FF8120C06020808CA402C0E42426101D75F81100A0F81100C060208077A402C0E444261C
S224FF8140101D78F81100A0F81100C0602080621C7802C021FF23A402C0E41826101D71F8EB
.
.
.
S224FFFF60FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9D
S224FFFF80FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7D
S224FFFFA0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D
S224FFFFC0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3D
S224FFFFE0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF802E6D
S9030000FC
R>
```

D-Bug12XZ – Stop Command

- STOP – Stop Execution of application code in the target MCU.
  - Use after uploading all target resources.
  - Places target MCU in active background, displays CPU registers & disassembles instruction at current PC.
  - Entire MCU state is preserved.
  - Can look at compiler ‘map’ file to see where CPU was executing.
  - If using internal watchdog, MCU will be reset ***UNLESS*** the RSBCK bit in COPCTRL is written to ‘1’.
    - This can be done using the Memory Modify (MM) command.
  - **If using an external watchdog, disable it!**

```
R>stop
Target Processor Has Been Stopped

  PC     SP     IX     IY    D0 D1  D2    D3    D4    D5     D6       D7     CCR = U IPL SX-I NZVC
FF807C 0010E9 0004A1 FF8707 04 00 0190 0000 0000 0000 00000000 00000000       0  0  1101 0000
FF807C  9D60                    INC.W    (0,SP)
S>
```

D-Bug12XZ – Memory Modify Commands

- MM – Memory Modify bytes of RAM or I/O.
  - Can be used to set RSBCK bit!
  - Syntax: MM <TAddress> [<data>..<data>]
    - Optional <data> is written to memory & returns to command prompt.
    - Interactive mode displays data at <TAddress> & allows modification.
      - Subcommands following displayed or new data:
        - <CR> - Display next location.
        - ‘=‘ – Display same location.
        - ‘-’ – Display previous location.
        - ‘.’ – Return to command prompt.
      - If subcommands follow new data, current location will be updated.
    - Verifies all writes and reports any errors.
- MMW - Memory Modify words of RAM or I/O.
  - Does not have to be aligned to a word boundary.

D-Bug12XZ – Memory Modify Example

- Stopping the target ***without*** and ***with*** the RSBCK bit set:

```
R>mm 6cc
0006CC 01 .
R>stop
Target Processor Has Been Stopped
Target CPU Running, Can't Access Registers

Target CPU Has Been Reset
R>mm 6cc
0006CC 01 40 =
Can't Write Target Memory
0006CC 41 .
R>stop
Target Processor Has Been Stopped

  PC      SP      IX      IY     D0  D1   D2    D3    D4    D5      D6        D7      CCR = U IPL SX-I
NZVC
FF807E 0010E9 00049F FF8716 04 00 0190 0000 0000 0000 00000000 00000000       0  0  1101
0000
FF807E  900190                 LD        D2,#$0190
S>
```

D-Bug12XZ – Other Useful Commands

- ‘HELP’ displays a list of commands with brief syntax.
  - Detailed command description can be found in D-Bug12XZ Reference Manual.
- ASM – single line assembler/disassembler.
- BF – Block Fill target memory (RAM or EE) with data.
- BR – Set/Display breakpoints.
- BS – Block Search, search memory for a specified data pattern.
- BULK – Bulk erase on-chip EEPROM or DFlash.
- FBULK – Erase the target MCU’s on-chip Flash.
- FLOAD – Program the MCU’s on-chip Flash from S-Records.
- G – Begin execution of application program.
- GT - Go Till, Set temporary breakpoint and begin execution of user program.

D-Bug12XZ – Other Useful Commands

- LOAD – Load program into target RAM in S-Record format.
- NOBR – Remove one/all breakpoints.
- RD – Register Display, display the CPU register contents.
- RESET – Reset the target MCU.
- RM – Register Modify, interactively examine/change CPU register contents.
- SECURE – Secure target device.
- T – Trace, execute an instruction, disassemble it, and display the CPU registers.
- TO – Trace Over subroutine calls (JSR, BSR, CALL).
- VERF – Verify memory contents against S-Record file.
- <RegisterName> <RegisterValue> – Set CPU <RegisterName> to <RegisterValue>.

D-Bug12XZ – LOG Command

- Used to display target memory locations at selected time intervals.
- Syntax: LOG [<LogAddress> <Bytes> <mSInterval>
  - Adds entry to log data table; 16 entries max.
  - <LogAddress> – Any valid target address.
  - <Bytes> – 1 to 250 bytes of data.
  - <mSInterval> – 1 to 65535 mS.
    - PC communication speed & BDM speed will ultimately limit display interval.
- Syntax: LOG START [;VT] [;BC]
  - Begins the display of data on the terminal.
  - “;VT” – Uses VT100 commands to display <LogAddress> data on its own line.
  - “;BC” – Enable use of target bus clock for BDM.
- Syntax: LOG CLEAR

D-Bug12XZ – LOG Command Example

- “LOG” used with no parameters displays current table entries.

```
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>
```

freescale™

D-Bug12XZ – Adding Comments To Log Files

- Comments can be added using the C++ comment delimiter “//”.
- Any characters following the comment delimiter are ignored.

```
R>mm 6cc         //examine the CPMUCOP register
0006CC 01 .
R>stop           // show what happens when RSBCK is not set
Target Processor Has Been Stopped
Target CPU Running, Can't Access Registers

Target CPU Has Been Reset
R>mm 6cc
0006CC 01 41 =   // write the RSBCK bit to ‘1’
0006CC 41 .      // exit interactive mm mode
R>stop           // stop the target MCU
Target Processor Has Been Stopped

  PC     SP     IX     IY    D0 D1  D2    D3    D4    D5      D6        D7      CCR = U IPL SX-I
NZVC
FF807E 0010E9 00049F FF8716 04 00 0190 0000 0000 0000 00000000 00000000       0  0  1101
0000
FF807E  900190                 LD      D2,#$0190
S>
```

D-Bug12XZ – Low-level BDM Debugger

- Normally used by tool developers or factory engineers when evaluating new silicon or debugging BDC communication problems.
- Issues individual BDM commands to the target.
- If hot connect cannot be achieved, the BDM debugger can be used to ‘probe’ the BDM interface.
- Unlike D-Bug12XZ, no check is made to ensure a target is connected prior to issuing a BDM command.
- The only BDM communication that occurs is as a result of an entered command.
- Can be entered from the “Can’t Communicate…” menu (#4) or from D-Bug12XZ’s command line (BDMDB).

D-Bug12XZ – Low-level BDM Debugger

- Debugger SYNC command issues BDM SYNC command.
  - Requests timed reference pulse to determine communication speed.
  - Initializes BDM drivers & enables BDM ACK protocol handshaking.
  - Sync pulse not detected generally means hardware connection problem.

```
S12Z BDM Command Debugger
For Commands type "HELP"

?sync
BDC Clock Frequency: 1000 KHz
?bdccsr
BDCCSR: 8200
?

?sync
Sync Pulse Not Detected
?
```

- BDCCSR displays/writes the control & status register.
  - 0x8200 – ENBDC = 1 & UNSEC = 1

D-Bug12XZ – Low-level BDM Debugger

- When S12Z device is secure, only BDCCSR can be read.
- When connected to S12/S12X and device is secure, all I/O registers can be read/written.
  - This is how “Erase & Unsecure” works for these devices.
  - Can read location 0x001a, target PARTID to verify BDM connection.
  - Location 0x0101, FSEC register, is a copy of Security byte at 0xff0f.
- Documentation for S12Z BDC debugger found in Appendix B.
- Documentation for S12/S12X BDM debugger found in Appendix A.

www.Freescale.com