DOCUMENT NUMBER
S12XBDMV1/D

Background Debug Module (BDM) Block Guide

V01.00

Original Release Date: 19 MAI 2003
Revised: 09 SEPTEMBER 2003

Motorola, Inc.

Revision History

Revision Number	Date	Author	Summary of Changes
01.00	5/19/2003	Nils Bossemeyer	Initial release.
00.01	6/10/2003	Nils Bossemeyer	Changed version counter
01.00	6/16/2003	Nils Bossemeyer	Changed version counter. Added information to section 4.8 regarding abort procedure for firmware commands.
01.00	9/02/2003	Nils Bossemeyer	Removed description of tagging feature which is now part of the DBG. Removed previous Section 2 and moved pin information to section 1.4. Added BDM CCR HIGH Byte register to BDM map. Changed content of BDM Status register (removed ENTAG bit and RESET value of CLKSW bit in emulation mode). Added information to deprecated TAGGO command. Removed peripheral mode information and added emulation mode information. Updated cycle information of BDM protocol (Note regarding External Wait function).
01.00	9/04/2003	Nils Bossemeyer	Added information to RESET state of ENBDM bit.
01.00	2/16/2004	Nils Bossemeyer	General updates.
01.00	4/06/2004	Nils Bossemeyer	Update regarding misaligned word accesses.
01.00	27/04/2004	Nils Bossemeyer	Correct information about write accessibility of CLKSW bit. Update of section 4.7 Serial Interface Hardware Handshake Protocol. Added internal only information.

Table of Contents

Section 1 Introduction to Background Debug Module (S12X_BDM)

Section 2 Detailed Description of the Background Interface Pin (BKGD)

Section 3 Memory Map/Register Definition

Section 4 Functional Description

List of Tables

List of Figures

Section 1 Introduction to Background Debug Module (S12X_BDM)

This section describes the functionality of the Background Debug Module (BDM) sub-block of the HCS12X Core Platform.

A block diagram of the BDM is shown in **Figure 1-1**.

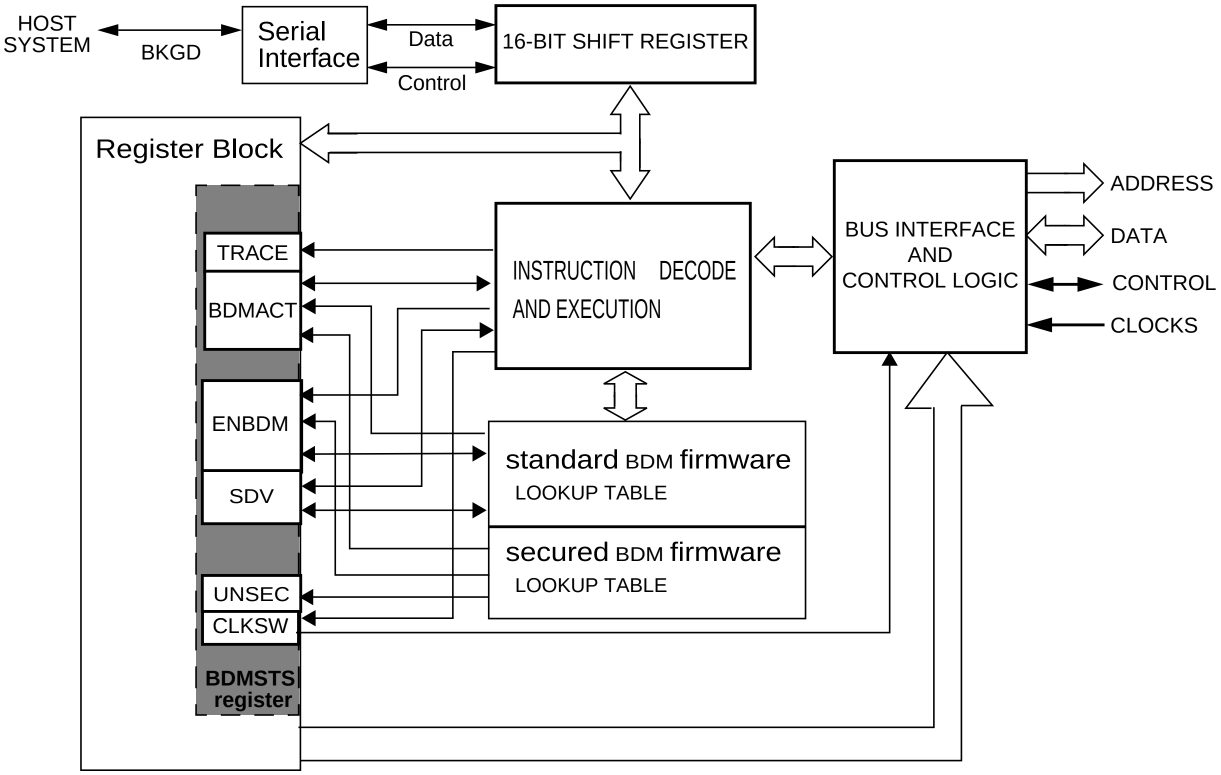

Figure 1-1 BDM Block Diagram

1.1 Overview

The Background Debug Module (BDM) sub-block is a single-wire, background debug system implemented in on-chip hardware for minimal CPU intervention. All interfacing with the BDM is done via the BKGD pin.

The BDM has enhanced capability for maintaining synchronization between the target and host while allowing more flexibility in clock rates. This includes a sync signal to determine the communication rate and a handshake signal to indicate when an operation is complete. The system is backwards compatible to the BDM of the S12 family with the following exceptions:

- TAGGO command no longer supported by BDM
- External instruction tagging feature now part of DBG module

- BDM register map and register content extended/modified
- Global page access functionality
- Enabled but not active out of reset in emulation modes
- CLKSW bit set out of reset in emulation mode.
- Family ID readable from firmware ROM at global address $7FFF0F (value for HCS12X devices is $C1)

1.2 Features

Features of the HCS12X BDM are:

- Single-wire communication with host development system
- Enhanced capability for allowing more flexibility in clock rates
- SYNC command to determine communication rate
- GO_UNTIL command
- Hardware handshake protocol to increase the performance of the serial communication
- Active out of reset in Special-Single-Chip mode
- Nine hardware commands using free cycles, if available, for minimal CPU intervention
- Hardware commands not requiring active BDM
- 14 firmware commands execute from the standard BDM firmware lookup table
- Software control of BDM operation during wait mode
- Software selectable clocks
- Global page access functionality
- Enabled but not active out of reset in emulation modes
- CLKSW bit set out of reset in emulation mode.
- When secured, hardware commands are allowed to access the register space in Special Single-Chip mode, if the FLASH and EEPROM erase tests fail.
- Family ID readable from firmware ROM at global address $7FFF0F (value for HCS12X devices is $C1)

1.3 Modes of Operation

BDM is available in all operating modes but must be enabled before firmware commands are executed. Some system s may have a control bit which allows suspending the function during background debug mode.

1.3.1 Regular Run Modes

All of these operations refer to the part in run mode and being not secured. The BDM does not provide controls to conserve power during run mode.

- Normal Operation

  General operation of the BDM is available and operates the same in all normal modes.

- Special single-chip mode

  In special single-chip mode, background operation is enabled and active out of reset. This allows programming a system with blank memory.

- Emulation Modes

  In emulation mode, background operation is enabled but not active out of reset. This allows debugging and programming a system in this mode more easily.

1.3.2 Secure Mode Operation

If the part is in secure mode, the operation of the BDM is reduced to a small subset of its regular run mode operation. Secure operation prevents access to FLASH or EEPROM other than allowing erasure. For more information please see **4.1 Security**.

1.3.3 Low-Power Modes

- Wait Mode

  The BDM cannot be used in wait mode if the system disables the clocks to the BDM (CWAI bit in CLKSEL register of CRG module is set).

  There is a clearing mechanism associated with the WAIT instruction when the clocks to the BDM are disabled. As the clocks restart from wait mode, the BDM receives a soft reset (clearing any command in progress) and the ACK function will be disabled. This is a change from previous BDM modules.

- Stop Mode

  The BDM is completely shutdown in stop mode.

  There is a clearing mechanism associated with the STOP instruction. STOP must be enabled and the part must go into stop mode for this to occur. As the clocks restart from stop mode, the BDM receives a soft reset (clearing any command in progress) and the ACK function will be disabled. This is a change from previous BDM modules.

Section 2 Detailed Description of the Background Interface Pin (BKGD)

A single-wire interface pin referred to as BKGD pin (Background interface pin) is used to communicate with the BDM system. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the background debug mode.

Section 3 Memory Map/Register Definition

3.1 BDM Memory Map

Table 3-1 show the BDM memory map when BDM is active.

Table 3-1 BDM Memory Map

Global Address	Module	Size (Bytes)
$7FFF00 - $7FFF09	BDM registers	10
$7FFF0A - $7FFF0E	BDM firmware ROM	5
$7FFF0F	Family ID (part of BDM firmware ROM)	1
$7FFF10 - $7FFFFF	BDM firmware ROM	240

3.2 Detailed Register Definition

A summary of the registers associated with the BDM is shown in **Table 3-2**. Registers are accessed by host-driven communications to the BDM hardware using READ_BD and WRITE_BD commands.

Table 3-2 BDM Register Map Summary

Global Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$7FFF00	Reserved	Read:	X	X	X	X	X	X	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
$7FFF01	BDMSTS	Read:	ENBDM	BDMACT	0	SDV	TRACE	CLKSW	UNSEC	0
		Write:		▨	▨	■	■		▨	▨
$7FFF02	Reserved	Read:	X	X	X	X	X	X	X	X
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
$7FFF03	Reserved	Read:	X	X	X	X	X	X	X	X
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
$7FFF04	Reserved	Read:	X	X	X	X	X	X	X	X
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
$7FFF05	Reserved	Read:	X	X	X	X	X	X	X	X
		Write:	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented, Reserved

■ = Implemented (do not alter)

X = Indeterminate

0 = Always read zero

Table 3-2 BDM Register Map Summary

Global Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$7FFF06	BDM - CCRL	Read:	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
		Write:								
$7FFF07	BDM - CCRH	Read:	0	0	0	0	0	CCR10	CCR9	CCR8
		Write:	▨	▨	▨	▨	▨			
$7FFF08	BDMGPR	Read:	BGAE	BGP6	BGP5	BGP4	BGP3	BGP2	BGP1	BGP0
		Write:								
$7FFF09	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨

▨	= Unimplemented, Reserved	■	= Implemented (do not alter)
X	= Indeterminate	0	= Always read zero

3.2.1 BDM Status Register

Register global address $7FFF01

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	0	SDV	TRACE	CLKSW	UNSEC	0
W		▨	▨	■	■		▨	▨
Reset:								
Special single-chip mode:	0[1]	1	0	0	0	0	0[3]	0
Emulation modes:	1	0	0	0	0	1[2]	0	0
All other modes:	0	0	0	0	0	0	0	0

▨	= Unimplemented, Reserved	■	= Implemented (do not alter)
0	= Always read zero		

NOTES:

1. ENBDM is read as "1" by a debugging environment in Special single-chip mode when the device is not secured or secured but fully erased (Flash and EEPROM). This is because the ENBDM bit is set by the standard firmware before a BDM command can be fully transmitted and executed.
2. CLKSW is read as "1" by a debugging environment in Emulation modes when the device is not secured and read as "0" when secured.
3. UNSEC is read as "1" by a debugging environment in Special single-chip mode when the device is secured and fully erased, else it is "0" and can only be read if not secure (see also bit description).

Figure 3-1 BDM Status Register (BDMSTS)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured, but subject to the following:

- ENBDM should only be set via a BDM hardware command if the BDM firmware commands are needed. (This does not apply in Special Single-Chip and Emulation Modes).
- BDMACT can only be set by BDM hardware upon entry into BDM. It can only be cleared by the standard BDM firmware lookup table upon exit from BDM active mode.
- CLKSW can only be written via BDM hardware WRITE_BD commands.
- All other bits, while writable via BDM hardware or standard BDM firmware write commands, should only be altered by the BDM hardware or standard firmware lookup table as part of BDM command execution.

ENBDM — Enable BDM

This bit controls whether the BDM is enabled or disabled. When enabled, BDM can be made active to allow firmware commands to be executed. When disabled, BDM cannot be made active but BDM hardware commands are still allowed.
1 = BDM enabled
0 = BDM disabled

NOTE: *ENBDM is set by the firmware out of reset in Special-Single-Chip mode and by hardware in Emulation modes. In Special-Single-Chip mode with the device secured, this bit will not be set by the firmware until after the EEPROM and FLASH erase verify tests are complete.In Emulation modes with the device secured the BDM operations are blocked.*

BDMACT — BDM active status

This bit becomes set upon entering BDM. The standard BDM firmware lookup table is then enabled and put into the memory map. BDMACT is cleared by a carefully timed store instruction in the standard BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map.
1 = BDM active
0 = BDM not active

SDV — Shift data valid

This bit is set and cleared by the BDM hardware. It is set after data has been transmitted as part of a firmware or hardware read command or after data has been received as part of a firmware or hardware write command. It is cleared when the next BDM command has been received or BDM is exited. SDV is used by the standard BDM firmware to control program flow execution.
1 = Data phase of command is complete
0 = Data phase of command not complete

TRACE — TRACE1 BDM firmware command is being executed

This bit gets set when a BDM TRACE1 firmware command is first recognized. It will stay set until BDM firmware is exited by one of the following BDM commands: GO or GO_UNTIL.
1 = TRACE1 command is being executed

0 = TRACE1 command is not being executed

CLKSW — Clock switch

The CLKSW bit controls which clock the BDM operates with. It is only writable from a hardware BDM command. A minimum delay of 150 cycles at the clock speed that is active during the data portion of the command send to change the clock source should occur before the next command can be send. The delay should be obtained no matter which bit is modified to effectively change the clock source (either PLLSEL bit or CLKSW bit). This guarantees that the start of the next BDM command uses the new clock for timing subsequent BDM communications.

Table 3-3 shows the resulting BDM clock source based on the CLKSW and the PLLSEL (PLL select in the CRG module, the bit is part of the CLKSEL register) bits.

Table 3-3 BDM Clock Sources

PLLSEL	CLKSW	BDMCLK
0	0	Bus clock dependent on oscillator
0	1	Bus clock dependent on oscillator
1	0	Alternate clock (refer to the device specification to determine the alternate clock source)
1	1	Bus clock dependent on the PLL

NOTE: *The BDM alternate clock source can only be selected when CLKSW = 0 and PLLSEL = 1. The BDM serial interface is now fully synchronized to the alternate clock source, when enabled. This eliminates frequency restriction on the alternate clock which was required on previous versions. Refer to the device specification to determine which clock connects to the alternate clock source input.*

NOTE: *If the acknowledge function is turned on, changing the CLKSW bit will cause the ACK to be at the new rate for the write command which changes it.*

NOTE: *In Emulation mode the CLKSW bit will be set out of RESET.*

UNSEC — Unsecure

This bit is only writable in special single-chip mode from the BDM secure firmware. It is in a zero state as secure mode is entered so that the secure BDM firmware lookup table is enabled and put into the memory map overlapping the standard BDM firmware lookup table.

The secure BDM firmware lookup table verifies that the on-chip EEPROM and FLASH EEPROM are erased. This being the case, the UNSEC bit is set and the BDM program jumps to the start of the standard BDM firmware lookup table and the secure BDM firmware lookup table is turned off. If the erase test fails, the UNSEC bit will not be asserted.

1 = System is in a unsecured mode.
0 = System is in a secured mode.

NOTE: *When UNSEC is set, security is off and the user can change the state of the secure bits in the on-chip FLASH EEPROM. Note that if the user does not change the state* *of the bits to “unsecured” mode, the system will be secured again when it is next taken out of reset.*

3.2.2 BDM CCR LOW Holding Register

Register global address $7FFF06

	7	6	5	4	3	2	1	0
R W	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
Reset:								
Special single-chip mode:	1	1	0	0	1	0	0	0
All other modes:	0	0	0	0	0	0	0	0

Figure 3-2 BDM CCR LOW Holding Register (BDMCCRL)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

NOTE: *When BDM is made active, the CPU stores the content of its* CCR_L *register in the BDMCCRL register. However, out of special single-chip reset, the BDMCCRL is set to $D8 and not $D0 which is the reset value of the* CCR_L *register in this CPU mode. Out of reset in all other modes the BDMCCRL register is read zero.*

When entering background debug mode, the BDM CCR LOW holding register is used to save the low byte of the condition code register of the user’s program. It is also used for temporary storage in the standard BDM firmware mode. The BDM CCR LOW holding register can be written to modify the CCR value.

3.2.3 BDM CCR HIGH Holding Register

Register global address $7FFF07

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CCR10	CCR9	CCR8
W	▨	▨	▨	▨	▨			
Reset:	0	0	0	0	0	0	0	0

Figure 3-3 BDM CCR HIGH Holding Register (BDMCCRH)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

When entering background debug mode, the BDM CCR HIGH holding register is used to save the high byte of the condition code register of the user’s program. The BDM CCR HIGH holding register can be written to modify the CCR value.

3.2.4 BDM Global Page Index Register

Register global address $7FFF08

	7	6	5	4	3	2	1	0
R W	BGAE	BGP6	BGP5	BGP4	BGP3	BGP2	BGP1	BGP0
Reset:	0	0	0	0	0	0	0	0

Figure 3-4 BDM Global Page Register (BDMGPR)

Read: All modes through BDM operation when not secured

Write: All modes through BDM operation when not secured

BGAE — BDM Global Page Access Enable Bit

BGAE enables global page access for BDM hardware and firmware read/write instructions The BDM hardware commands used to access the BDM registers (READ_BD_ and WRITE_BD_)can not be used for global accesses even when the BGAE bit is set.
1 = BDM Global Access enabled
0 = BDM Global Access disabled

BGP6–BGP0 — BDM Global Page Index Bits 6 - 0

These bits define the extended address bits from 22 to 16. For more detailed information regarding the Global Page Window scheme please refer to the S12X_MMC Block Guide.

3.3 Family ID Assignment

The family ID is a 8-bit value located in the firmware ROM (at global address: $7FFF0F). The read-only value is a unique family ID which is $C1 for S12X devices.

Section 4 Functional Description

The BDM receives and executes commands from a host via a single wire serial interface. There are two types of BDM commands, namely, hardware commands and firmware commands.

Hardware commands are used to read and write target system memory locations and to enter active background debug mode, see **4.3 BDM Hardware Commands**. Target system memory includes all memory that is accessible by the CPU.

Firmware commands are used to read and write CPU resources and to exit from active background debug mode, see **4.4 Standard BDM Firmware Commands**. The CPU resources referred to are the accumulator (D), X index register (X), Y index register (Y), stack pointer (SP), and program counter (PC).

Hardware commands can be executed at any time and in any mode excluding a few exceptions as highlighted (see **4.3 BDM Hardware Commands**) and in secure mode **4.1 Security**. Firmware commands can only be executed when the system is not secure and is in active background debug mode (BDM).

4.1 Security

If the user resets into special single-chip mode with the system secured, a secured mode BDM firmware lookup table is brought into the map overlapping a portion of the standard BDM firmware lookup table. The secure BDM firmware verifies that the on-chip EEPROM and FLASH EEPROM are erased. This being the case, the UNSEC and ENBDM bit will get set. The BDM program jumps to the start of the standard BDM firmware and the secured mode BDM firmware is turned off and all BDM commands are allowed. If the EEPROM or FLASH do not verify as erased, the BDM firmware sets the ENBDM bit, without asserting UNSEC, and the firmware enters a loop. This causes the BDM hardware commands to become enabled, but does not enable the firmware commands. This allows the BDM hardware to be used to erase the EEPROM and FLASH.

BDM operation is not possible in any other mode than special single-chip mode when the device is secured. The device can only be unsecured via BDM serial interface in special single-chip mode.For more information regarding security please see the S12X_9SEC document.

4.2 Enabling and Activating BDM

The system must be in active BDM to execute standard BDM firmware commands. BDM can be activated only after being enabled. BDM is enabled by setting the ENBDM bit in the BDM status (BDMSTS) register. The ENBDM bit is set by writing to the BDM status (BDMSTS) register, via the single-wire interface, using a hardware command such as WRITE_BD_BYTE.

After being enabled, BDM is activated by one of the following(1):

- Hardware BACKGROUND command

NOTES:

1. BDM is enabled and active immediately out of special single-chip reset.

- CPU BGND instruction
- External instruction tagging mechanism(1)
- Breakpoint force or tag mechanism(1)

When BDM is activated, the CPU finishes executing the current instruction and then begins executing the firmware in the standard BDM firmware lookup table. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

> ***NOTE:*** *If an attempt is made to activate BDM before being enabled, the CPU resumes normal instruction execution after a brief delay. If BDM is not enabled, any hardware BACKGROUND commands issued are ignored by the BDM and the CPU is not delayed.*

In active BDM, the BDM registers and standard BDM firmware lookup table are mapped to addresses $7FFF00 to $7FFFFF. BDM registers are mapped to addresses $7FFF00 to $7FFF08. The BDM uses these registers which are readable anytime by the BDM. However, these registers are not readable by user programs.

4.3 BDM Hardware Commands

Hardware commands are used to read and write target system memory locations and to enter active background debug mode. Target system memory includes all memory that is accessible by the CPU such as on-chip RAM, EEPROM, FLASH EEPROM, I/O and control registers, and all external memory.

Hardware commands are executed with minimal or no CPU intervention and do not require the system to be in active BDM for execution, although, they can still be executed in this mode. When executing a hardware command, the BDM sub-block waits for a free CPU bus cycle so that the background access does not disturb the running application program. If a free cycle is not found within 128 clock cycles, the CPU is momentarily frozen so that the BDM can steal a cycle. When the BDM finds a free cycle, the operation does not intrude on normal CPU operation provided that it can be completed in a single cycle. However, if an operation requires multiple cycles the CPU is frozen until the operation is complete, even though the BDM found a free cycle.

The BDM hardware commands are listed in **Table 4-1**.

NOTES:

1. This method is provided by the S12X_DBG module.

Table 4-1 Hardware Commands

Command	Opcode (hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware is enabled. If enabled, an ACK will be issued when the part enters active background mode.
ACK_ENABLE	D5	None	Enable Handshake. Issues an ACK pulse after the command is executed.
ACK_DISABLE	D6	None	Disable Handshake. This command does not issue an ACK pulse.
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Must be aligned access.

NOTE:
If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

The READ_BD and WRITE_BD commands allow access to the BDM register locations. These locations are not normally in the system memory map but share addresses with the application in memory. To distinguish between physical memory locations that share the same address, BDM memory resources are enabled just for the READ_BD and WRITE_BD access cycle. This allows the BDM to access BDM locations unobtrusively, even if the addresses conflict with the application memory map.

4.4 Standard BDM Firmware Commands

Firmware commands are used to access and manipulate CPU resources. The system must be in active BDM to execute standard BDM firmware commands, see **4.2 Enabling and Activating BDM**. Normal instruction execution is suspended while the CPU executes the firmware located in the standard BDM firmware lookup table. The hardware command BACKGROUND is the usual way to activate BDM.

As the system enters active BDM, the standard BDM firmware lookup table and BDM registers become visible in the on-chip memory map at \$7FFF00–\$7FFFFF, and the CPU begins executing the standard BDM firmware. The standard BDM firmware watches for serial commands and executes them as they are received.

The firmware commands are shown in **Table 4-2**.

Table 4-2 Firmware Commands

Command[1]	Opcode (hex)	Data	Description
READ_NEXT[2]	62	16-bit data out	Increment X by 2 (X = X + 2), then read word X points to.
READ_PC	63	16-bit data out	Read program counter.
READ_D	64	16-bit data out	Read D accumulator.
READ_X	65	16-bit data out	Read X index register.
READ_Y	66	16-bit data out	Read Y index register.
READ_SP	67	16-bit data out	Read stack pointer.
WRITE_NEXT[2]	42	16-bit data in	Increment X by 2 (X = X + 2), then write word to location pointed to by X.
WRITE_PC	43	16-bit data in	Write program counter.
WRITE_D	44	16-bit data in	Write D accumulator.
WRITE_X	45	16-bit data in	Write X index register.
WRITE_Y	46	16-bit data in	Write Y index register.
WRITE_SP	47	16-bit data in	Write stack pointer.
GO	08	none	Go to user program. If enabled, ACK will occur when leaving active background mode.
GO_UNTIL[3]	0C	none	Go to user program. If enabled, ACK will occur upon returning to active background mode.
TRACE1	10	none	Execute one user instruction then return to active BDM. If enabled, ACK will occur upon returning to active background mode.
TAGGO -> GO	18	none	(Previous enable tagging and go to user program.) **This command will be deprecated and should not be used anymore. Opcode will be executed as a GO command.**

NOTES:

1. If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

2. When the firmware command READ_NEXT or WRITE_NEXT is used to access the BDM address space the BDM resources are accessed rather than user code. Writing BDM firmware is not possible.
3. Both WAIT (with clocks to the S12 CPU core disabled) and STOP disable the ACK function. The GO_UNTIL command will not get an Acknowledge if one of these two CPU instructions occurs before the "UNTIL" condition (BDM active, see **4.7 Serial Interface Hardware Handshake Protocol** last Note).

4.5 BDM Command Structure

Hardware and firmware BDM commands start with an 8-bit opcode followed by a 16-bit address and/or a 16-bit data word depending on the command. All the read commands return 16 bits of data despite the byte or word implication in the command name.

NOTE: *8-bit reads return 16-bits of data, of which, only one byte will contain valid data. If reading an even address, the valid data will appear in the MSB. If reading an odd address, the valid data will appear in the LSB.*

NOTE: *16-bit misaligned reads and writes are generally not allowed. If attempted by BDM hardware command, the BDM will ignore the least significant bit of the address and will assume an even address from the remaining bits.*

NOTE: *The following cycle count information is only valid when the External Wait function is not used (see EWAIT bit of EBI sub-block). During an external wait the BDM can not steal a cycle. Hence be careful with the External Wait function when the BDM serial interface is much faster than the bus, because of the BDM soft-reset after time-out (see* ***4.11 Serial Communication Time-out****).*

For hardware data read commands, the external host must wait at least 150 bus clock cycles after sending the address before attempting to obtain the read data. This is to be certain that valid data is available in the BDM shift register, ready to be shifted out. For hardware write commands, the external host must wait 150 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed. The 150 bus clock cycle delay in both cases includes the maximum 128 cycle delay that can be incurred as the BDM waits for a free cycle before stealing a cycle.

For firmware read commands, the external host should wait at least 44 bus clock cycles after sending the command opcode and before attempting to obtain the read data. This includes the potential of extra cycles when the access is external and stretched (+1, 2, or 3 cycles) or to registers of the PRU (Port Replacement Unit) in emulation mode. The 44 cycle wait allows enough time for the requested data to be made available in the BDM shift register, ready to be shifted out.

NOTE: *This timing has increased from previous BDM modules due to the new capability in which the BDM serial interface can potentially run faster than the bus. On previous BDM modules this extra time could be hidden within the serial time.*

For firmware write commands, the external host must wait 32 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed.

The external host should wait at least for 64 bus clock cycles after a TRACE1 or GO command before starting any new serial command. This is to allow the CPU to exit gracefully from the standard BDM firmware lookup table and resume execution of the user code. Disturbing the BDM shift register prematurely may adversely affect the exit from the standard BDM firmware lookup table.

NOTE: *If the bus rate of the target processor is unknown or could be changing or the External Wait function is used, it is recommended that the ACK (acknowledge function) be used to indicate when an operation is complete. When using ACK, the delay times are automated.*

Figure 4-1 represents the BDM command structure. The command blocks illustrate a series of eight bit times starting with a falling edge. The bar across the top of the blocks indicates that the BKGD line idles in the high state. The time for an 8-bit command is 8 × 16 target clock cycles.(1)

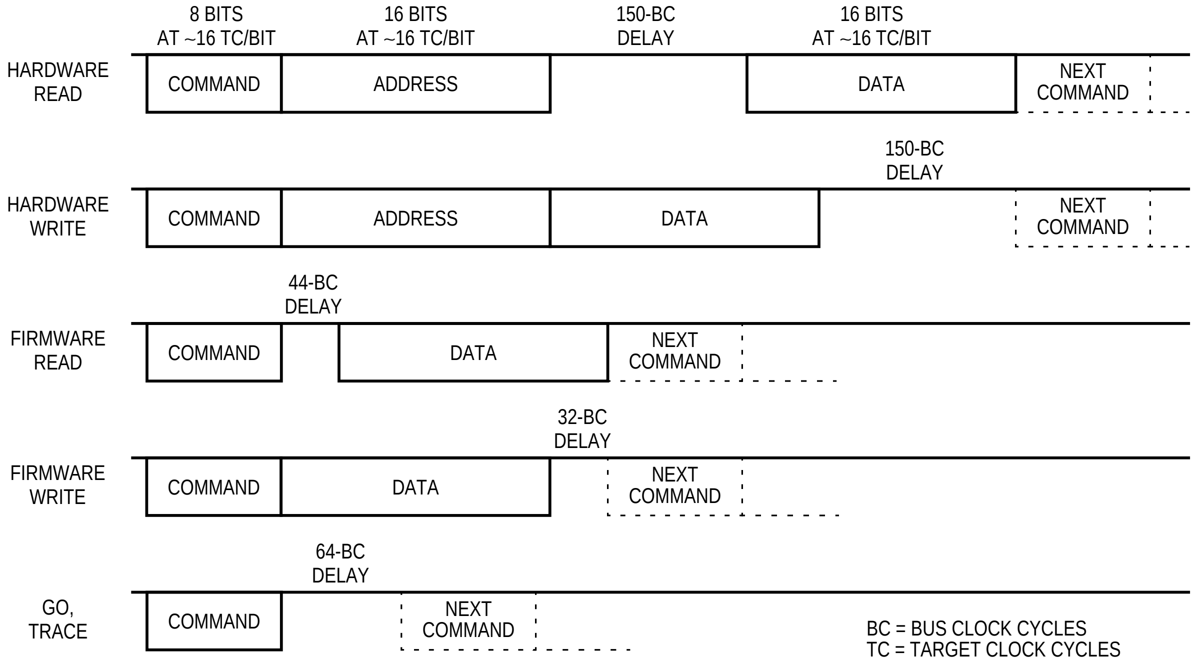

Figure 4-1 BDM Command Structure

4.6 BDM Serial Interface

The BDM communicates with external devices serially via the BKGD pin. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the BDM.

The BDM serial interface is timed using the clock selected by the CLKSW bit in the status register see **3.2.1 BDM Status Register**. This clock will be referred to as the target clock in the following explanation.

NOTES:

1. Target clock cycles are cycles measured using the target MCU’s serial clock rate. See **4.6 BDM Serial Interface** and **3.2.1 BDM Status Register** for information on how serial clock rate is selected.

The BDM serial interface uses a clocking scheme in which the external host generates a falling edge on the BKGD pin to indicate the start of each bit time. This falling edge is sent for every bit whether data is transmitted or received. Data is transferred most significant bit (MSB) first at 16 target clock cycles per bit. The interface times out if 512 clock cycles occur between falling edges from the host.

The BKGD pin is a pseudo open-drain pin and has an weak on-chip active pull-up that is enabled at all times. It is assumed that there is an external pull-up and that drivers connected to BKGD do not typically drive the high level. Since R-C rise time could be unacceptably long, the target system and host provide brief driven-high (speedup) pulses to drive BKGD to a logic 1. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

The timing for host-to-target is shown in **Figure 4-2** and that of target-to-host in **Figure 4-3** and **Figure 4-4**. All four cases begin when the host drives the BKGD pin low to generate a falling edge. Since the host and target are operating from separate clocks, it can take the target system up to one full clock cycle to recognize this edge. The target measures delays from this perceived start of the bit time while the host measures delays from the point it actually drove BKGD low to start the bit up to one target clock cycle earlier. Synchronization between the host and target is established in this manner at the start of every bit time.

Figure 4-2 shows an external host transmitting a logic 1 and transmitting a logic 0 to the BKGD pin of a target system. The host is asynchronous to the target, so there is up to a one clock-cycle delay from the host-generated falling edge to where the target recognizes this edge as the beginning of the bit time. Ten target clock cycles later, the target senses the bit level on the BKGD pin. Internal glitch detect logic requires the pin be driven high no later that eight target clock cycles after the falling edge for a logic 1 transmission.

Since the host drives the high speedup pulses in these two cases, the rising edges look like digitally driven signals.

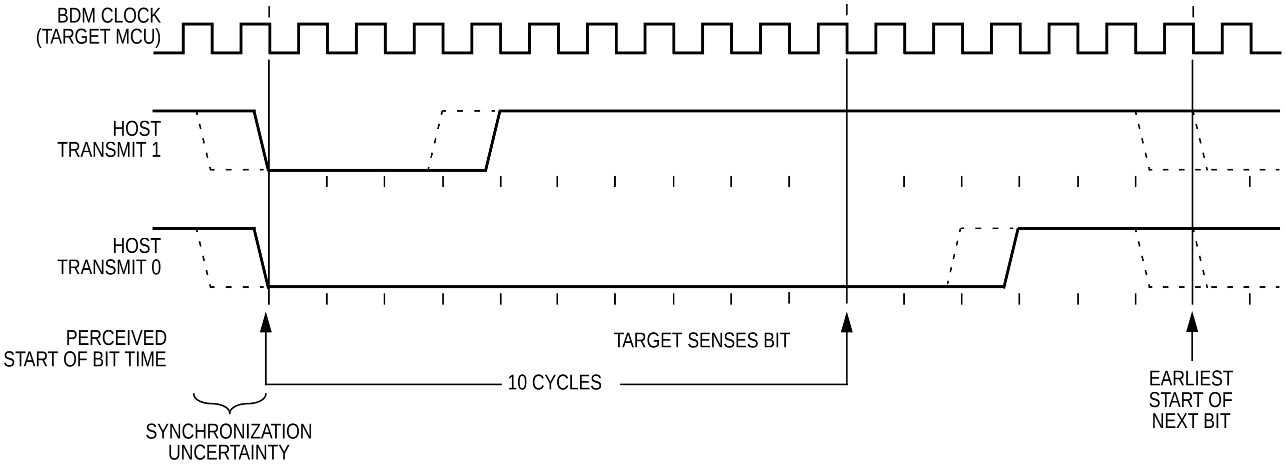

Figure 4-2 BDM Host-to-Target Serial Bit Timing

The receive cases are more complicated. **Figure 4-3** shows the host receiving a logic 1 from the target system. Since the host is asynchronous to the target, there is up to one clock-cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target. The host holds the BKGD pin low long enough for the target to recognize it (at least two target clock cycles). The host must release the low drive before the target drives a brief high speedup pulse seven target clock cycles after the perceived start of the bit time. The host should sample the bit level about 10 target clock cycles after it started the bit time.

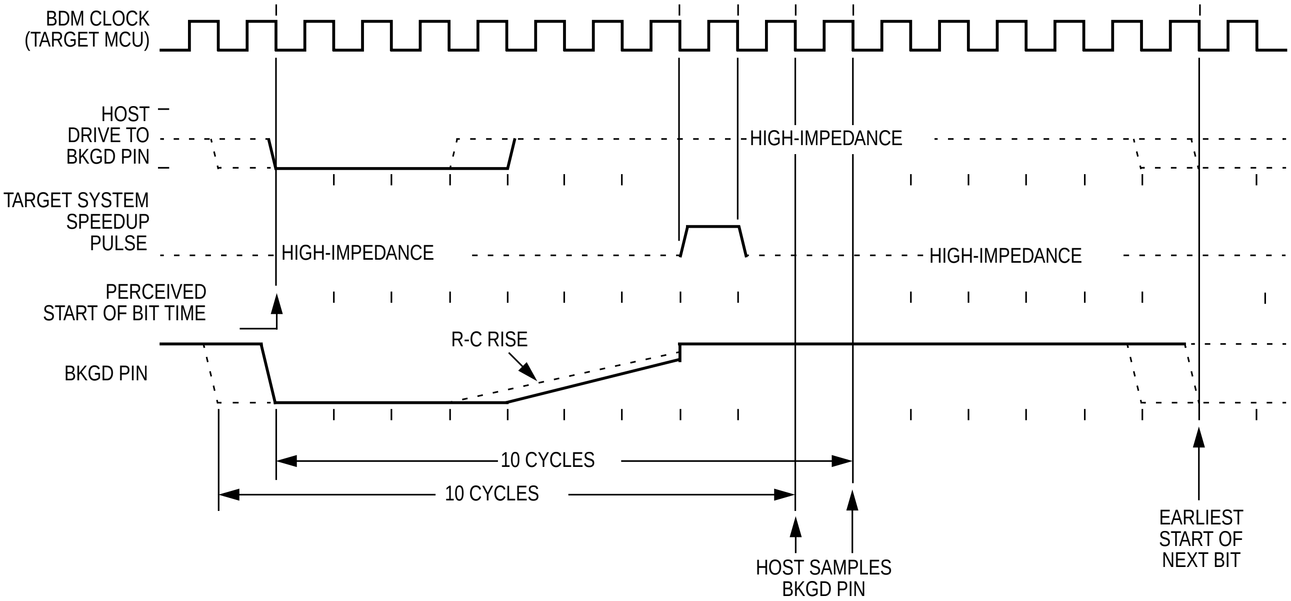

Figure 4-3 BDM Target-to-Host Serial Bit Timing (Logic 1)

Figure 4-4 shows the host receiving a logic 0 from the target. Since the host is asynchronous to the target, there is up to a one clock-cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target. The host initiates the bit time but the target finishes it. Since the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 target clock cycles then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 target clock cycles after starting the bit time.

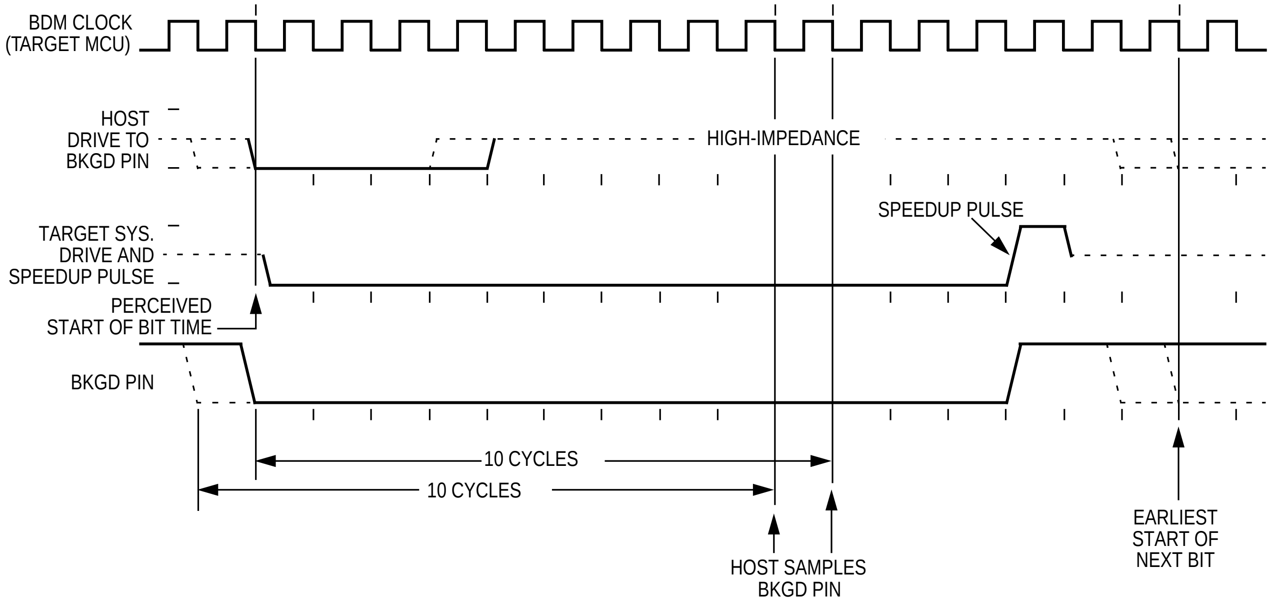

Figure 4-4 BDM Target-to-Host Serial Bit Timing (Logic 0)

4.7 Serial Interface Hardware Handshake Protocol

BDM commands that require CPU execution are ultimately treated at the MCU bus rate. Since the BDM clock source can be asynchronously related to the bus frequency, when CLKSW = 0, it is very helpful to provide a handshake protocol in which the host could determine when an issued command is executed by the CPU. The alternative is to always wait the amount of time equal to the appropriate number of cycles at the slowest possible rate the clock could be running. This sub-section will describe the hardware handshake protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a 16 serial clock cycle low pulse followed by a brief speedup pulse in the BKGD pin. This pulse is generated by the target MCU when a command, issued by the host, has been successfully executed (see **Figure 4-5**). This pulse is referred to as the ACK pulse. After the ACK pulse has finished: the host can start the bit retrieval if the last issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, GO_UNTIL or TRACE1). The ACK pulse is not issued earlier than 32 serial clock cycles after the BDM command was issued. The end of the BDM command is assumed to be the 16th tick of the last bit. This minimum delay assures enough time for the host to perceive the ACK pulse. Note also that, there is no upper limit for the delay between the command and the related ACK pulse, since the command execution depends upon the CPU bus frequency, which in some cases could be very slow compared to the serial communication rate. This protocol allows a great flexibility for the POD designers, since it does not rely on any accurate time measurement or short response time to any event in the serial communication.

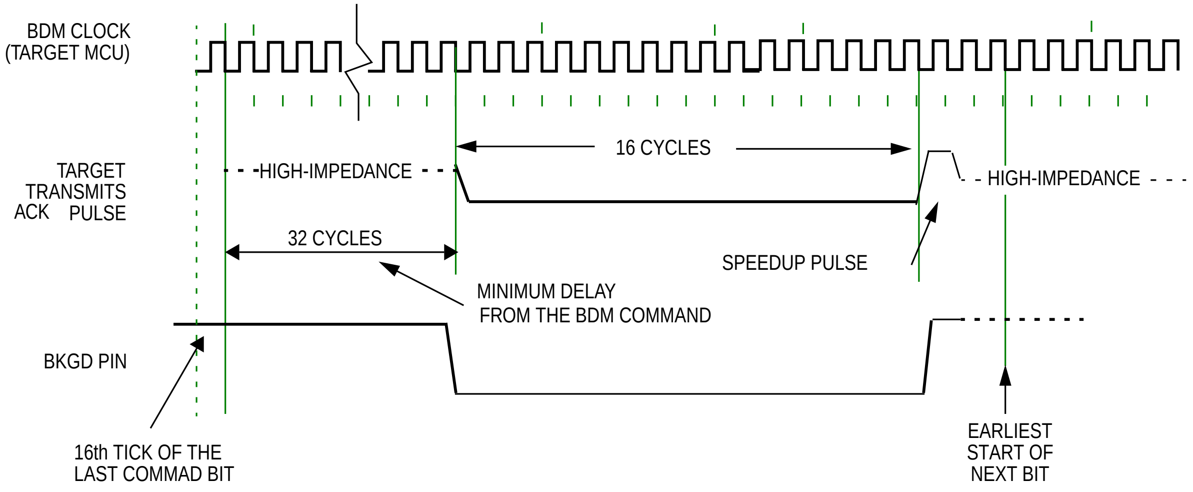

Figure 4-5 Target Acknowledge Pulse (ACK)

NOTE: *If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters WAIT or STOP prior to executing a hardware command, the ACK pulse will not be issued meaning that the BDM command was not executed. After entering wait or stop mode, the BDM command is no longer pending.*

Figure 4-6 shows the ACK handshake protocol in a command level timing diagram. The READ_BYTE instruction is used as an example. First, the 8-bit instruction opcode is sent by the host, followed by the address of the memory location to be read. The target BDM decodes the instruction. A bus cycle is grabbed (free or stolen) by the BDM and it executes the READ_BYTE operation. Having retrieved the data, the BDM issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the byte retrieval process. Note that data is sent in the form of a word and the host needs to determine which is the appropriate byte based on whether the address was odd or even.

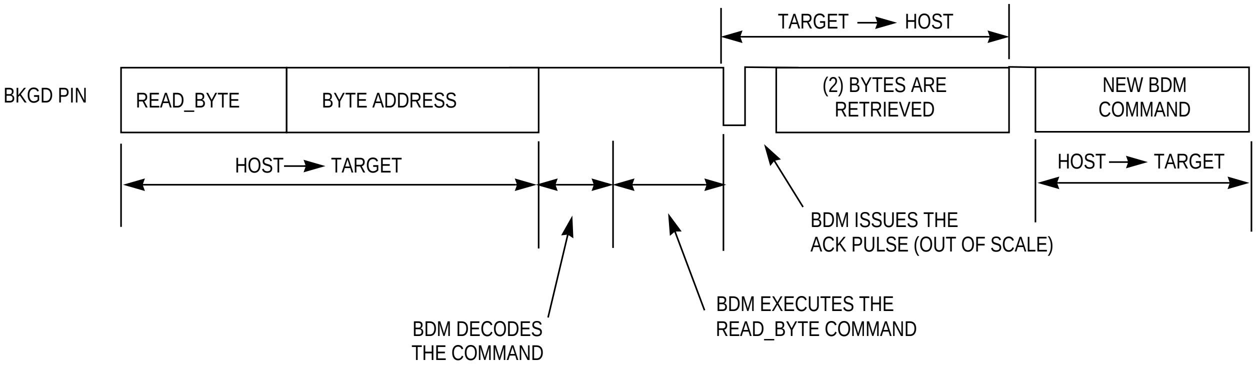

Figure 4-6 Handshake Protocol at Command Level

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negedge in the BKGD pin. The hardware handshake protocol in **Figure 4-5** specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

NOTE: *The only place the BKGD pin can have an electrical conflict is when one side is driving low and the other side is issuing a speedup pulse (high). Other "highs" are pulled rather than driven. However, at low rates the time of the speedup pulse can become lengthy and so the potential conflict time becomes longer as well.*

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters WAIT or STOP while the host issues a hardware command (e.g., WRITE_BYTE), the target discards the incoming command due to the WAIT or STOP being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host (not aware of STOP or WAIT) should decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

NOTE: *The ACK pulse does not provide a time out. This means for the GO_UNTIL command that it can not be distinguished if a STOP or WAIT has been executed (command discarded and ACK not issued) or if the "UNTIL" condition (BDM active) is just not reached yet. Hence in any case where the ACK pulse of a command is not issued the possible pending command should be aborted before issuing a new command. See the handshake abort procedure described in* ***4.8 Hardware Handshake Abort Procedure****.*

4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command, which had not issued the corresponding ACK pulse, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle, providing a speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see **4.9 SYNC — Request Timed Reference Pulse**, and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the SYNC protocol has been completed the host is free to issue new BDM commands. For Firmware READ or WRITE commands it can not be guaranteed that the pending command is aborted when issuing a SYNC before the corresponding ACK pulse. There is a short latency time from the time the READ or WRITE access begins until it is finished and the corresponding ACK pulse is issued. The latency time depends on the firmware READ or WRITE command that is issued and if the serial interface is running on a different clock rate than the bus. When the SYNC command starts during this latency time the READ or WRITE command will not be aborted, but the corresponding ACK pulse will be aborted. A pending GO, TRACE1 or GO_UNTIL command can not be aborted. Only the corresponding ACK pulse can be aborted by the SYNC command.

Although it is not recommended, the host could abort a pending BDM command by issuing a low pulse in the BKGD pin shorter than 128 serial clock cycles, which will not be interpreted as the SYNC command. The ACK is actually aborted when a negedge is perceived by the target in the BKGD pin. The short abort pulse should have at least 4 clock cycles keeping the BKGD pin low, in order to allow the negedge to be detected by the target. In this case, the target will not execute the SYNC protocol but the pending command will be aborted along with the ACK pulse. The potential problem with this abort procedure is when there is a conflict between the ACK pulse and the short abort pulse. In this case, the target may not perceive the abort pulse. The worst case is when the pending command is a read command (i.e., READ_BYTE). If the abort pulse is not perceived by the target the host will attempt to send a new command after the abort pulse was issued, while the target expects the host to retrieve the accessed memory byte. In this case, host and target will run out of synchronism. However, if the command to be aborted is not a read command the short abort pulse could be used. After a command is aborted the target assumes the next negedge, after the abort pulse, is the first bit of a new BDM command.

NOTE: *The details about the short abort pulse are being provided only as a reference for the reader to better understand the BDM internal behavior. It is not recommended that this procedure be used in a real application.*

Since the host knows the target serial clock frequency, the SYNC command (used to abort a command) does not need to consider the lower possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length. Providing a small overhead on the pulse length in order to assure the SYNC pulse will not be misinterpreted by the target. See **4.9 SYNC — Request Timed Reference Pulse**.

Figure 4-7 shows a SYNC command being issued after a READ_BYTE, which aborts the READ_BYTE command. Note that, after the command is aborted a new command could be issued by the host computer.

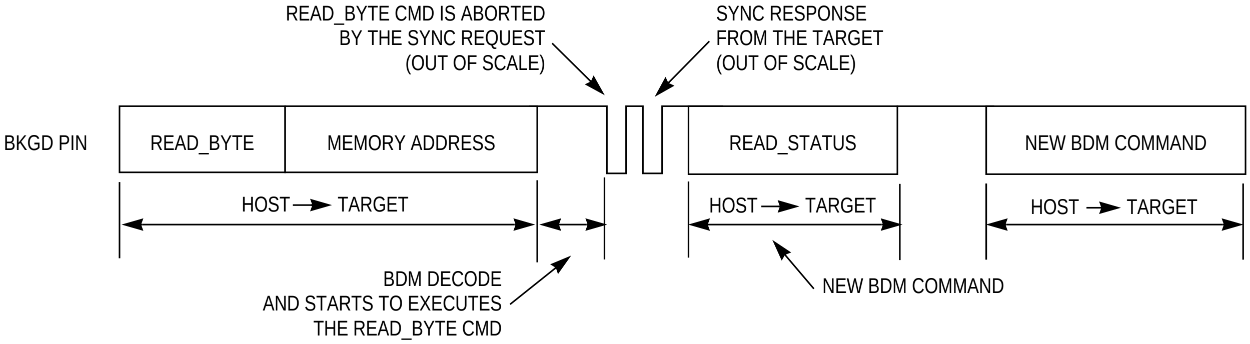

Figure 4-7 ACK Abort Procedure at the Command Level

NOTE: Figure 4-7 *does not represent the signals in a true timing scale*

Figure 4-8 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug active mode. Consider that the target CPU is executing a pending BDM command at the exact moment the POD is being connected to the BKGD pin. In this case, an ACK pulse is issued along with the SYNC command. In this case, there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Since this is not a probable situation, the protocol does not prevent this conflict from happening.

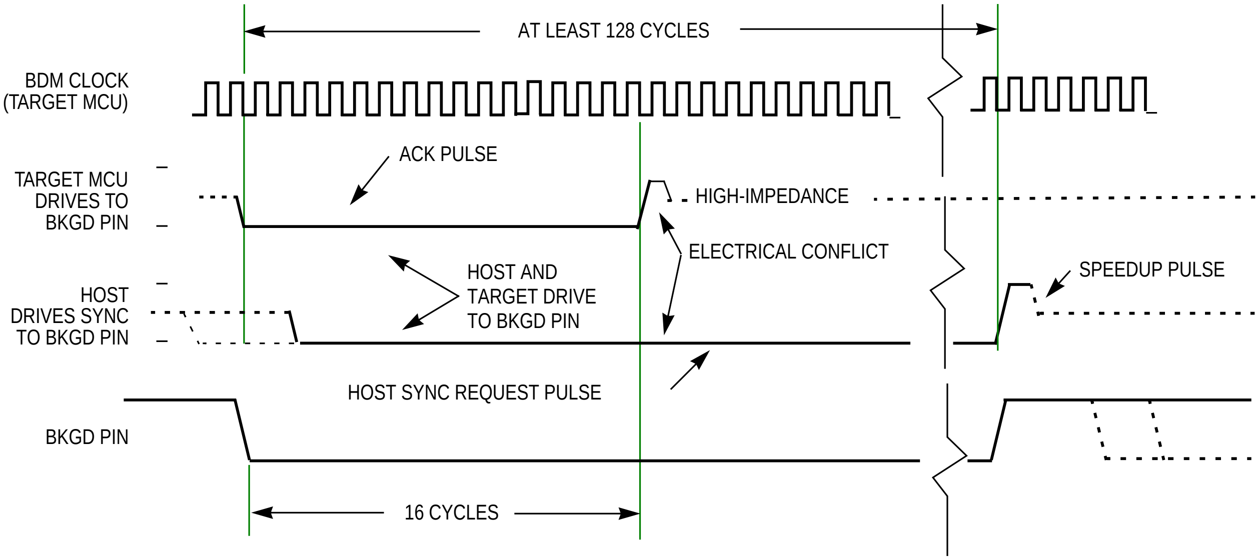

Figure 4-8 ACK Pulse and SYNC Request Conflict

NOTE: *This information is being provided so that the MCU integrator will be aware that such a conflict could eventually occur.*

The hardware handshake protocol is enabled by the ACK_ENABLE and disabled by the ACK_DISABLE BDM commands. This provides backwards compatibility with the existing POD devices which are not able to execute the hardware handshake protocol. It also allows for new POD devices, that support the hardware handshake protocol, to freely communicate with the target device. If desired, without the need for waiting for the ACK pulse.

The commands are described as follows:

- ACK_ENABLE — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See **4.3 BDM Hardware Commands** and **4.4 Standard BDM Firmware Commands** for more information on the BDM commands.

The ACK_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target since it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a Breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

4.9 SYNC — Request Timed Reference Pulse

The SYNC command is unlike other BDM commands because the host does not necessarily know the correct communication speed to use for BDM communications until after it has analyzed the response to the SYNC command. To issue a SYNC command, the host should perform the following steps:

1. Drive the BKGD pin low for at least 128 cycles at the lowest possible BDM serial communication frequency (the lowest serial communication frequency is determined by the crystal oscillator or the clock chosen by CLKSW.)
2. Drive BKGD high for a brief speedup pulse to get a fast rise time (this speedup pulse is typically one cycle of the host clock.)
3. Remove all drive to the BKGD pin so it reverts to high impedance.
4. Listen to the BKGD pin for the sync response pulse.

Upon detecting the SYNC request from the host, the target performs the following steps:

1. Discards any incomplete command received or bit retrieved.
2. Waits for BKGD to return to a logic one.
3. Delays 16 cycles to allow the host to stop driving the high speedup pulse.
4. Drives BKGD low for 128 cycles at the current BDM serial communication frequency.
5. Drives a one-cycle high speedup pulse to force a fast rise time on BKGD.

6. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128 cycle SYNC response pulse and determines the correct speed for subsequent BDM communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

As soon as the SYNC request is detected by the target, any partially received command or bit retrieved is discarded. This is referred to as a soft-reset, equivalent to a time-out in the serial communication. After the SYNC response, the target will consider the next negedge (issued by the host) as the start of a new BDM command or the start of new SYNC request.

Another use of the SYNC command pulse is to abort a pending ACK pulse. The behavior is exactly the same as in a regular SYNC command. Note that one of the possible causes for a command to not be acknowledged by the target is a host-target synchronization problem. In this case, the command may not have been understood by the target and so an ACK response pulse will not be issued.

4.10 Instruction Tracing

When a TRACE1 command is issued to the BDM in active BDM, the CPU exits the standard BDM firmware and executes a single instruction in the user code. Once this has occurred, the CPU is forced to return to the standard BDM firmware and the BDM is active and ready to receive a new command. If the TRACE1 command is issued again, the next user instruction will be executed. This facilitates stepping or tracing through the user code one instruction at a time.

If an interrupt is pending when a TRACE1 command is issued, the interrupt stacking operation occurs but no user instruction is executed. Once back in standard BDM firmware execution, the program counter points to the first instruction in the interrupt service routine.

Be aware when tracing through the user code that the execution of the user code is done step by step but all peripherals are free running. Hence possible timing relations between CPU code execution and occurrence of events of other peripherals no longer exist.

4.11 Serial Communication Time-out

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD in order to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting forever without any time-out limit.

Consider now the case where the host returns BKGD to logic one before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a time-out occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset.

If a read command is issued but the data is not retrieved within 512 serial clock cycles, a soft-reset will occur causing the command to be disregarded. The data is not available for retrieval after the time-out has occurred. This is the expected behavior if the handshake protocol is not enabled. However, consider the behavior where the BDC is running in a frequency much greater than the CPU frequency. In this case, the command could time out before the data is ready to be retrieved. In order to allow the data to be retrieved even with a large clock frequency mismatch (between BDC and CPU) when the hardware handshake protocol is enabled, the time out between a read command and the data retrieval is disabled. Therefore, the host could wait for more then 512 serial clock cycles and still be able to retrieve the data from an issued read command. However, once the handshake pulse (ACK pulse) is issued, the time-out feature is re-activated, meaning that the target will time out after 512 clock cycles. Therefore, the host needs to retrieve the data within a 512 serial clock cycles time frame after the ACK pulse had been issued. After that period, the read command is discarded and the data is no longer available for retrieval. Any negedge in the BKGD pin after the time-out period is considered to be a new command or a SYNC request.

Note that whenever a partially issued command, or partially retrieved data, has occurred the time out in the serial communication is active. This means that if a time frame higher than 512 serial clock cycles is observed between two consecutive negative edges and the command being issued or data being retrieved is not complete, a soft-reset will occur causing the partially received command or data retrieved to be disregarded. The next negedge in the BKGD pin, after a soft-reset has occurred, is considered by the target as the start of a new BDC command, or the start of a SYNC request pulse.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

MOTOROLA

S12X_BDMV1/D
Rev. 4.02
2/2003