AN1817

MMC20xx M•CORE OnCE Port Communication and Control Sequences

By Steve Becker
Freescale Microcontroller Division, Applications Group
Austin, Texas

Introduction

The on-chip emulation (OnCE) port in Freescale's M•CORE M200 core is a JTAG-like (Joint Test Action Group) serial interface. An external device called a command controller communicates with and controls an M•CORE M200xx core core through the core's OnCE port. In addition to other tasks, the command controller can cause the core to stop executing at a predefined instruction or data fetch or even to program a non-volatile memory device that might be connected to the core. Freescale sells an enhanced background debug interface (EBDI), a version of a command controller.

This application note describes the specific serial command sequences a command controller should present to a OnCE port to communicate with and control an M•CORE M200 core. All MMX20xx Family microcontrollers (MCU) contain this core.

These communication and control sequences adhere to the OnCE controller state diagrams in the *MMC2001Reference Manual* and *MMC2003 Reference Manual*, Freescale document order numbers MCORERM/AD and MMC2003RM/D, respecitvely. The input variable shown in the state diagrams, TMS (test or debug mode select), is used

for steering between states during each rising edge of TCK, the debug (test) serial clock.

The communication and control sequences were derived by reviewing the software contained within a particular command controller and by reviewing a OnCE port design specification. No effort has been made to test these sequences verbatim. The sequences are intended as guides only.

M•CORE OnCE Port Register Architecture

Figure 1 and Figure 2 show the 8-bit JTAG (Joint Test Action Group) instruction register, the OnCE command register (OCMR), and the OnCE (JTAG) data registers in the MMC20xx.

Writing 0x3 to the 8-bit JTAG instruction register will ensure that all further attempts to communicate with that register will be redirected to the 8-bit OnCE command (instruction) register until the microcontroller containing the M200 is reset. The MMC20xx's OnCE port is controlled using the OnCE command (instruction) register.

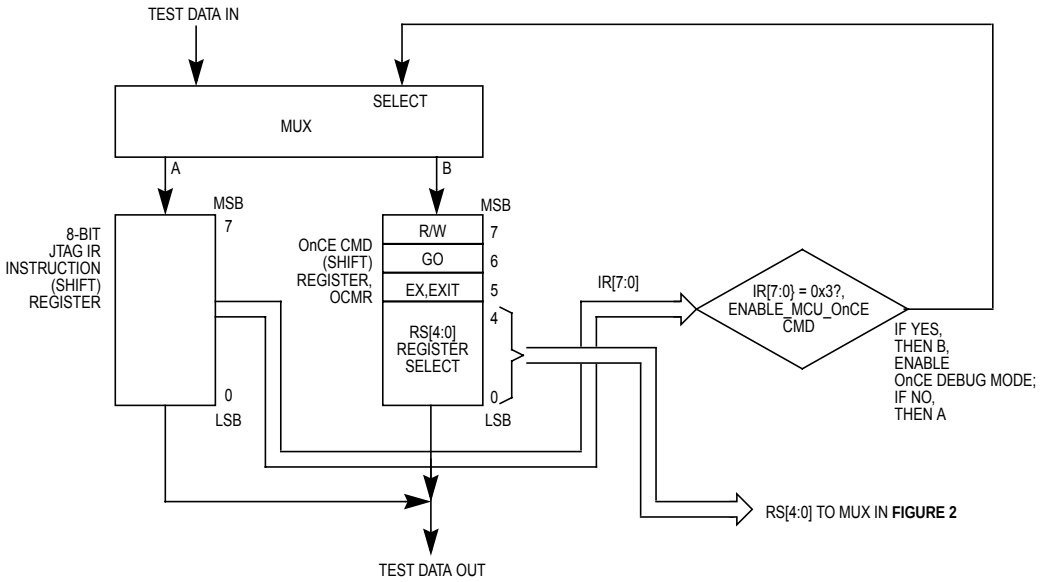

Figure 1. JTAG and OnCE Instruction Registers (IRs)

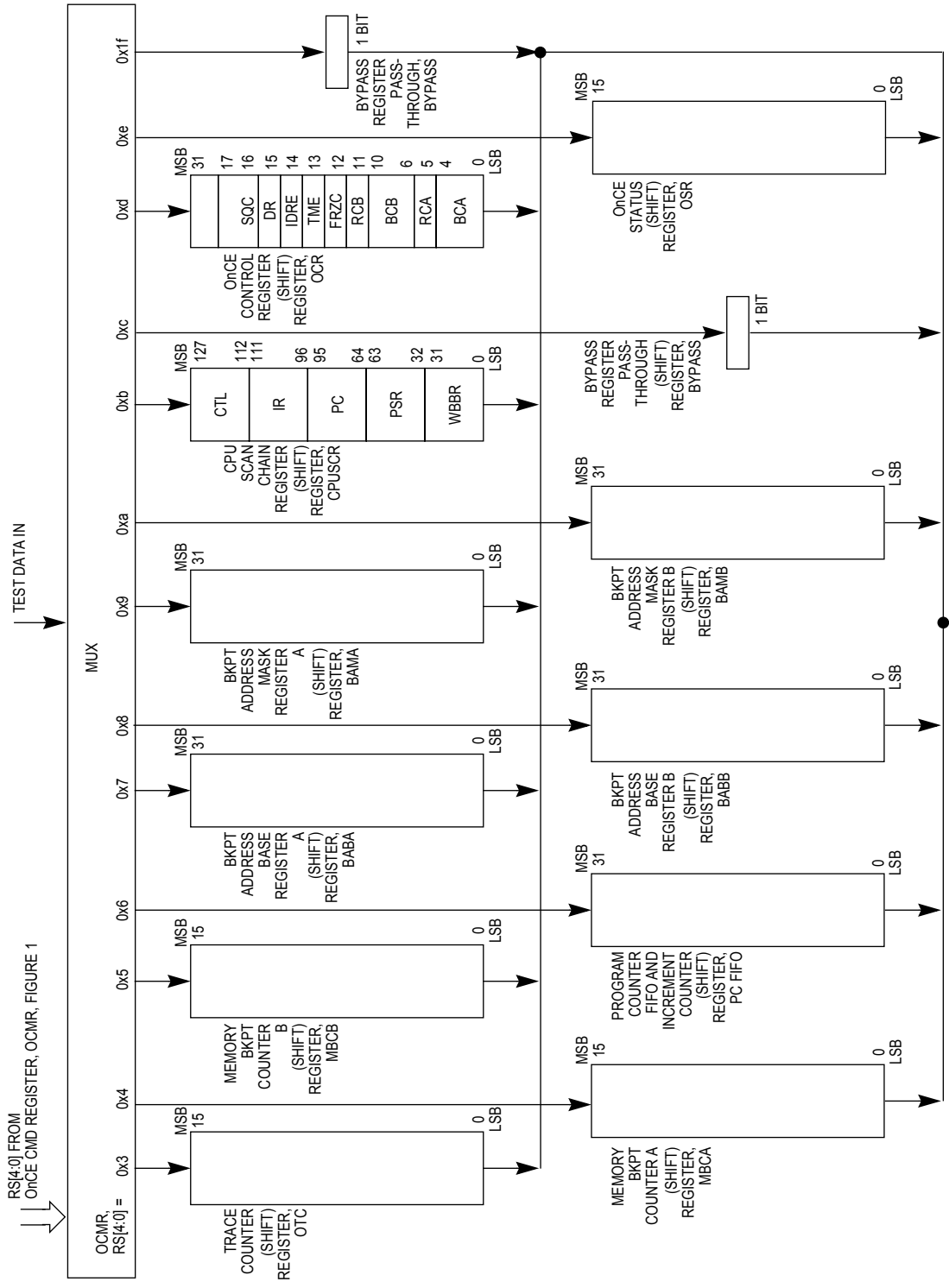

Figure 2. JTAG Data Registers (DRs)

The OnCE debug module's serial interface expects a debug clock, TCK, that is no more than half of the frequency of the MMC20xx's CPU (central processor unit) clock, CLK. Per JTAG protocol, the value of test mode select, TMS, during each rising edge of the debug module's clock, TCK, will determine the next TAP (test access port) controller state to be entered.

An external device called a command controller is used to present TCK, TMS, and data to the OnCE module's serial interface.

The following examples show the sequences required to perform specific tasks with the OnCE port.

OnCE port examples are:

1. Entering debug mode by asserting the debug enable $\overline{DE}$ pin
2. Entering debug mode by setting the debug request (DR) bit
3. Polling the MMC20xx OnCE status register
4. Reading/writing an MMC20xx register while in debug mode
5. Causing the MMC20xx to exit from debug mode to user mode
6. Reading and writing memory using the MMC20xx's JTAG/OnCE port
7. Single-stepping the MMC20xx
8. Setting a breakpoint and exiting to user mode

Entering Debug Mode

When in debug mode:

1. The MMC20xx's vector base register (VBR) always must be set to a valid memory location. The transfer error acknowledge (TEA) exception vector in the VBR should be set to a valid memory location.
2. The watchdog enable bit (WDBG) in the watchdog control register (WCR) should be set. If the MMC20xx attempts to access invalid memory so that an internal bus transfer acknowledge (TA) is not asserted to the CPU within 128 CPU CLK cycles, the watchdog will time out and cause an internal transfer error acknowledge (TEA) to be presented to the CPU to terminate the access. If the watchdog was not enabled, the MMC20xx would wait indefinitely for either TA or TEA to be asserted.

The two ways to place the MMC20xx into debug mode are:

- To assert the debug enable ($\overline{DE}$) signal
- To set the debug request (DR) bit in the MMC20xx's OnCE control register, OCR

Using the Debug Enable ($\overline{DE}$) Pin

Figure 3 shows the timing associated with asserting the debug enable $\overline{DE}$ pin to put the MMC20xx into debug mode.

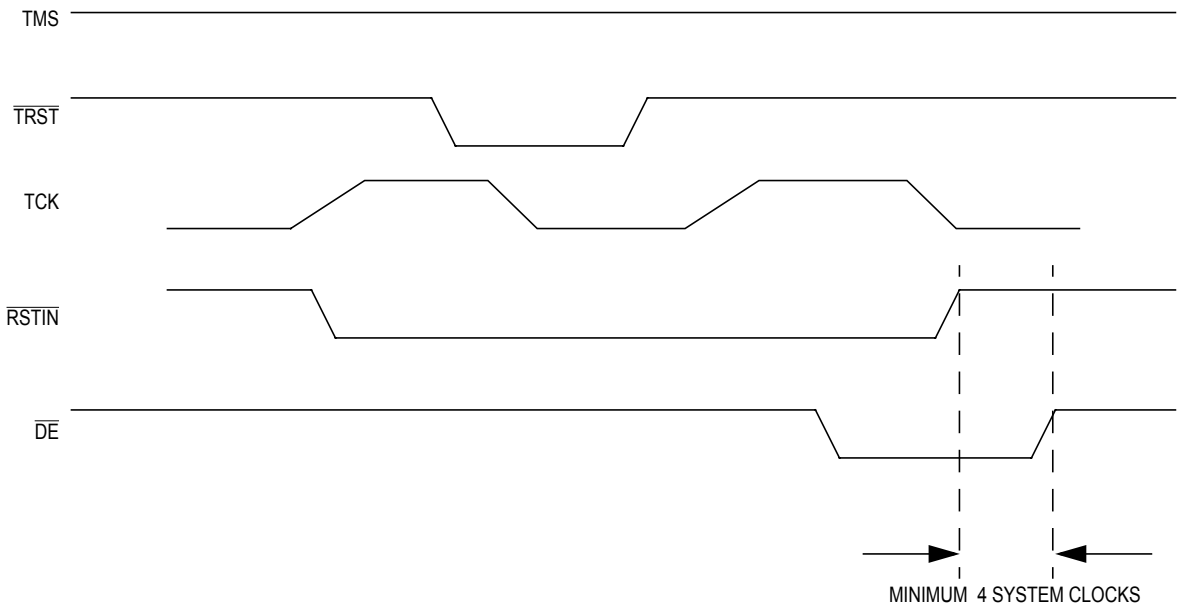

Figure 3. M200 OnCE Port Signal Relationships

After completion, the OnCE status register (OSR) processor mode bits, PM[1:0], must be polled to ensure that M•CORE has entered debug mode before accessing any M•CORE registers or the OnCE CPU scan chain register, CPUSCR.

Entering Debug Mode by Setting the DR Bit

Table 1 shows the method for putting the M200 into debug mode using the DR (debug request) control bit.

Assumptions —

- The MMC20xx is not in debug mode.
- The JTAG state machine is in the test logic reset state.

Table 1. TMS Sequence for Entering Debug Mode

Step	Test Module Select	JTAG State	Note
1	1	Test-logic-reset	Reset state after assertion of TRST or power-on reset
2	0	Run-test/idle	
3	1	Select-DR-scan	
4	1	Select-IR-scan	
5	0	Capture-IR	Capture the MMC20xx instruction register, IR.
6	0	Shift-IR	Load the 8-bit JTAG IR with 0x3, the ENABLE_MCU_OnCE command. See **Figure 1**.
	3 TCLKs		
7	1	Exit1-IR	The rising edge of TCK while entering this state will shift an additional bit into the JTAG IR. At this point, the M200 JTAG IR is ready to be loaded.
8	1	Update-IR	The JTAG IR is updated.
9	1	Select-DR-scan	
10	1	Select-IR-scan	
11	0	Capture-IR	Capture the MMC20xx IR.
12	0	Shift-IR	The 8-bit MMC20xx OnCE command (instruction) register, OCMR, must be loaded with 0x0d to select the OnCE control register, OCR, for **writing**, cause the value in the MMC20xx IR to **not be executed**, and cause the MMC20xx to remain in debug mode. (Since GO is 0, the EX bit is actually ignored.)
	7 TCLKs		
13	1	Exit1-IR	The rising edge of TCK, while entering this state, will shift the last bit into the MMC20xx IR.
14	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated.
15	1	Select-DR-scan	
16	0	Capture-DR	
17	0	Shift-DR	Shift the value 0x8000 to the OnCE control register, OCR, to set its debug request, DR, bit to 1.
	31 TCLKs		
18	1	Exit1-DR	The rising edge of TCK, while entering this state, will shift the last bit into the OnCE control register, OCR.
19	1	Update-DR	The OnCE control register (OCR) is updated, and the MMC20xx will enter debug mode after the current instruction executes.
20	0	Run-test/idle	

The JTAG instruction ENABLE_MCU_ONCE, 0x3, will activate the OnCE state machine and all further communication via the SELECT-IRSCAN path will be with the 8-bit OnCE command (instruction) register rather than the 8-bit JTAG TAP IR until the device containing the MMC20xx is reset.

Polling the MMC20xx OnCE Status Register

Assumption: The instruction ENABLE_MCU_OnCE, 0x3, is in the JTAG instruction register (IR) so that the MMC20xx is in debug mode.

Once the processor mode (PM) status bits in the MMC20xx OnCE status register, OSR, are $(10)_2$ to indicate the MMC20xx is in debug mode, the external command controller may safely access the MMC20xx's OnCE port registers.

Table 2. TMS Sequencing for Polling the MMC20xx OnCE Status Register

Step	Test Module Select	JTAG State	Note
1	0	Run-test/idle	
2	1	Select-DR-scan	
3	1	Select-IR-scan	
4	0	Capture-IR	
5	0	Shift-IR	Load the OnCE command (instruction) register (OCMR) with 0x8E, to **read** OnCE status register (OSR), to **not execute** the instruction, and to **not exit** debug mode.
	7 TCLKs		
6	1	Exit1-IR	Last bit is shifted in during rising edge of TCK while entering state.
7	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated. Their contents appear at their parallel outputs.
8	1	Select-DR-scan	
9	0	Capture-DR	MMC20xx OnCE status register (OSR) is captured. Value at parallel inputs is copied into register.
10	0	Shift-DR	The MMC20xx OnCE status register (OSR) is shifted out, a total DR shift of 16 bits.
	15 TCLKs		
11	1	Exit1-DR	Last bit of data is shifted from data register, DR. At this point, the external JTAG controller has all of the MMC20xx OnCE status register (OSR) status bits and can now check if the processor mode PM[1:0] bits are $(10)_2$.
12	1	Update-DR	
13	0	Run-test/idle	If PM[1:0] is not equal to $(10)_2$, loop back to step 5.

Reading/Writing an MMC20xx Register While in Debug Mode

A **mov reg,reg** opcode must be used to read or write an MMC20xx register while in debug mode. In the case of writes, the value to be written to the register must reside in the write-back bus register (WBBR) of the CPU scan chain register (CPUSCR) when the write begins.

Opcodes Used to Read and Write Registers

Table 3 shows opcodes for reading and writing to each MMC20xx register.

When accessing the alternate register file (r[0:15]'), the PSR register S bit and AF bits must be set. The r0 register is used to read/write to the control registers. Thus, the contents of r0 must be saved and restored when accessing cr[0:12].

Table 3. Opcodes Used to Access MMC20xx Registers

Opcode	Read/Write Mnemonic	Opcode	Read Mnemonic	Opcode	Write Mnemonic
1200	mov r0,r0	PSR accesses are made through the OnCE CPUSCR.			
1211	mov r1,r1	1010	mfcr r0,cr1 (VBR)	1810	mtcr r0,cr1 (VBR)
1222	mov r2,r2	1020	mfcr r0,cr2 (EPSR)	1820	mtcr r0,cr2 (EPSR)
1233	mov r3,r3	1030	mfcr r0,cr3 (FPSR)	1830	mtcr r0,cr3 (FPSR)
1244	mov r4,r4	1040	mfcr r0,cr4 (EPC)	1840	mtcr r0,cr4 (EPC)
1255	mov r5,r5	1050	mfcr r0,cr5 (FPC)	1850	mtcr r0,cr5 (FPC)
1266	mov r6,r6	1060	mfcr r0,cr6 (SS0)	1860	mtcr r0,cr6 (SS0)
1277	mov r7,r7	1070	mfcr r0,cr7 (SS1)	1870	mtcr r0,cr7 (SS1)
1288	mov r8,r8	1080	mfcr r0,cr8 (SS2)	1880	mtcr r0,cr8 (SS2)
1299	mov r9,r9	1090	mfcr r0,cr9 (SS3)	1890	mtcr r0,cr9 (SS3)
12AA	mov r10,r10	10A0	mfcr r0,cr10 (SS4)	18A0	mtcr r0,cr10 (SS4)
12BB	mov r11,r11	10B0	mfcr r0,cr11 (GCR)	18B0	mtcr r0,cr11 (GCR)
12CC	mov r12,r12	10C0	mfcr r0,cr12 (GSR)	18C0	mtcr r0,cr12 (GSR)
12DD	mov r13,r13				
12EE	mov r14,r14				
12FF	mov r15,r15				

Writing to a Register In Debug Mode

This example changes the value of the MMC20xx r15 register to 0xDEADBEEF.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, is in the JTAG instruction register (IR) so that the MMC20xx is in debug mode.
- The contents of the CPU scan chain register (CPUSCR) have been saved.
- The JTAG state machine is in the run-test/idle state.

Initial register values for **Table 4** are:

1. The data value of register Y is the value loaded into the write-back bus register (WBBR). The contents of the WBBR will be written to register X. When the feed forward Y operand (FFY) bit in the control state register (CTL) is not set, the value loaded into the WBBR section of the CPU scan chain (shift) register (CPUSCR) is not important.
2. CTL = 0xFFDB: All internal state bits are set to 1;
   FFY = 1 (use WBBR); FDB = 1 (debug mode); SZ = 0b10 (16-bit);
   TC = 0b110 (supervisor instruction access).
   If the FFY bit is cleared to 0, the access will be considered a read R15 register operation using the same opcode for mov R15,R15, but the WBBR will be loaded with the value of R15 from the write-back bus register (WBBR).
3. PC = 32-bit address. Program counter normally is set to the PC value saved when debug mode is entered.
4. WBBR = the data to be loaded into R15, 0xDEADBEEF.

Table 4. Sequence for Writing to R15

Step	TMS	JTAG	Note
1	0	Run-test/idle	
2	1	Select-DR-scan	
3	1	Select-IR-scan	
4	0	Capture-IR	Capture the MMC20xx's IR.
5	0	Shift-IR	Shift 0x4b into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to not … debug mode.
7 TCLKs			
6	1	Exit1-IR	The last bit is shifted in during the rising edge of TCK while entering this state.
7	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated.
8	1	Select-DR-scan	
9	0	Capture-DR	
10	0	Shift-DR	Write CPU scan chain register (CPUSCR) with IR = 0x12FF, the opcode for mov r15,r15 control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, copy value of origin register Y(R0) to destination register X(R15); force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered; processor status register (PSR) = 0xA000,0000; write-back bus register (WBBR) = new value to be written into R15
127 TCLKs			
11	1	Exit1-DR	The last bit is shifted in during the rising edge of TCK while entering this state.
12	1	Update-DR	Following this update, MMC20xx register R15 is written with the data in the write-back bus register, WBBR.
13	0	Run-test/idle	The MMC20xx is still in debug mode.

Reading from a Register in Debug Mode

This example reads the contents of R0.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, resides in the JTAG instruction register (IR) so that the MMC20xx is in debug mode,
- The contents of the CPU scan chain register (CPUSCR) have been saved.
- The JTAG state machine is in the run-test/idle state.

Table 5. Sequence for Reading from R0 (Sheet 1 of 2)

Step	Test Module Select	JTAG	Note
1	0	Run-test/idle	
2	1	Select-DR-scan	
3	1	Select-IR-scan	
4	0	Capture-IR	
5	0	Shift-IR	Shift 0x4b into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
	7 TCLKs		
6	1	Exit1-IR	The last bit is shifted in during the rising edge of TCK while entering this state.
7	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated.
8	1	Select-DR-scan	
9	0	Capture-DR	
10	0	Shift-DR	Write CPU scan chain register (CPUSCR) with IR = 0x1200, the opcode for **mov r0,r0**; control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, do not copy the value of the write-back bus register (WBBR) to destination register X(R0); force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered; processor status register (PSR) = 0x8000 0000 write-back bus register (WBBR) = 0
	127 TCLKs		
11	1	Exit1-DR	The last bit is shifted in during the rising edge of TCK while entering this state.

Table 5. Sequence for Reading from R0 (Sheet 2 of 2)

Step	Test Module Select	JTAG	Note
12	1	Update-DR	
13	0	Run-test/idle	
14	1	Select-DR-scan	
15	1	Select-IR-scan	
16	0	Capture-IR	
17	0	Shift-IR	Shift 0x8b into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **execute** the instruction in the IR, and to not exit debug mode.
7 TCLKs			
18	1	Exit1-IR	The last bit is shifted in during the rising edge of TCK while entering this state. Now the MMC20xx OnCE command (instruction) register's (OCMRs) parallel outputs are ready to be updated.
19	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated.
20	1	Select-DR-scan	
21	0	Capture-DR	
22	0	Shift-DR	The contents of the 128-bit MMC20xx CPU scan chain register CPUSCR are shifted out. The write-back bus register (WBBR) portion of it will contain the value of R0.
127 TCLKs			
23	1	Exit1-DR	Last bit of data is shifted out of CPUSCR
24	1	Update-DR	
25	0	Run-test/idle	

Causing the MMC20xx to Exit from Debug Mode to User Mode

Assumptions —

- The MMC20xx is in debug mode.
- The user has set the OnCE control register (OCR) properly for hardware breakpoints prior to executing this sequence.
- Tracing was disabled by clearing the trace mode enable (TME) bit in the OnCE control register (OCR).
- The value of the CPU scan chain register (CPUSCR) was saved before entering debug mode.
- The contents of all of the MMC20xx's non-debug mode registers have been restored to their values before debug mode was entered
- The JTAG state machine is in the run-test/idle state, and there are no upcoming changes to program flow.

Table 6. Sequence for Transitioning from Debug Mode to User Mode (Sheet 1 of 2)

Step	Test Module Select	JTAG	Note
1	0	Run-test/idle	
2	1	Select-DR-scan	
3	1	Select-IR-scan	
4	0	Capture-IR	
5	0	Shift-IR	Shift 0x0b into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
7 TCLKs			
6	1	Exit1-IR	Last bit of data is shifted into the OnCE IR.
7	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated. Contents of the OCMR are presented at the OCMR's parallel outputs.
8	1	Select-DR-scan	
9	0	Capture-DR	

Table 6. Sequence for Transitioning from Debug Mode to User Mode (Sheet 2 of 2)

Step	Test Module Select	JTAG	Note
10	0	Shift-DR	Write CPU scan chain register (CPUSCR) with IR=0x0001 (sync), the opcode for **sync**; set control state register (CTL) = 0xFE4F for feed forward Y (FFY) operand = 0, do not copy the value of the write-back bus register (WBBR) to destination register X(R0); set force PSR debug enable mode (FDB) bit = 0 for not enabling debug enable mode; set PC = the PC saved when debug mode was entered; set processor status register (PSR) = (value of PSR saved before entering debug mode) ORed with 0xA000 0000, which sets the supervisor bit and a bit which was formerly used. set write-back bus register (WBBR) = 0
127 TCLKs			
11	1	Exit1-DR	The last bit of data is shifted in during the rising edge of TCK while entering this state.
12	1	Update-DR	The parallel outputs of the MMC20xx's CPU scan chain register (CPUSCR) are updated.
13	0	Run-test/idle	
14	1	Select-DR-scan	
15	1	Select-IR-scan	
16	0	Capture-IR	
17	0	Shift-IR	Shift 0xEC into the OnCE command (instruction) register (OCMR) to select the 1-bit bypass (pass-through) register for **writing**, to **execute** the instruction in the IR, and to **exit** debug mode.
7 TCLKs			
18	1	Exit1-IR	The last bit of data is shifted into the IR during the rising edge of TCK while entering this state.
19	1	Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated.
20	0	Run-test/idle	
21	1	Select-DR-scan	
23	0	Capture-DR	
24	1	Shift-DR	The contents of the 1-bit bypass (pass-through) register will be shifted out 8 times to the external command controller.
7 TCLKs			
25	1	Exit1-DR	
26	1	Update-DR	The MMC20xx will **exit** debug mode and enter user mode.
27	0	Run-test/idle	

Reading from Memory Using the MMC20xx's JTAG/OnCE Port

This example will read an 8-, 16-, or 32-bit value from memory using register R0 as a pointer:

1. Using the write-back bus register (WBBR), save the value presently in R0 so that it can be restored to R0 at the end of this example.
2. Write the address of the memory location to be read to R0 using the WBBR.
3. Using R0 as a pointer and the WBBR, read the value at the memory location.
4. Using the WBBR, restore the original value to R0 that was saved in step 1.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, resides in the JTAG instruction register (IR) so that the MMC20xx is in debug mode.
- The original value of the CPU scan chain register (CPUSCR) has been saved.
- The JTAG state machine is in the run-test/idle state.

Table 7. Steps for Reading from a Memory Location (Sheet 1 of 3)

Step	Task	Actions
These steps will: 1. Save the present value in MMC20xx register R0 before R0 is used for transferring data 2. Write instruction that will get value of R0 to CPU scan chain register (CPUSCR)		
1	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
2	Load the CPU scan chain register (CPUSCR).	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) = 0
	Read value of R0 from write-back bus register (WBBR).	
3	Select CPU scan chain register (CPUSCR) for **reading**.	Shift **0xcb** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **execute** the instruction in the IR, and to **not exit** debug mode.
4	Get value of R0 and save it for later use.	Shift out the contents of the CPU scan chain register (CPUSCR) which contains the write-back bus register (WBBR). The WBBR contains the value of R0. The MMC20xx's R0 should be restored to this value at the end of this example.

Table 7. Steps for Reading from a Memory Location (Sheet 2 of 3)

Step	Task	Actions
These steps will: 1. Copy the address of the memory location to be read from the external command controller to the MMC20xx's register R0		
5	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
6	Load the CPU scan chain register (CPUSCR).	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, enabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) is set to address of memory location to be read
	Address of memory location to be read is now in R0.	
These steps will: 1. Read the memory location 2. Write an **ld** instruction to the IR in the CPU scan chain register (CPUSCR)		
7	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
8	Shift values into the CPU scan chain register (CPUSCR). State: Shift-DR	Shift the following into the CPU scan chain register (CPUSCR): for a 32-bit memory location, IR = 0x8000 for **ld.w R0,(R0,0);** for a 16-bit memory location, IR = 0xC000 for **ld.h r0,(r0,0)**; for an 8-bit memory location, IR = 0xA000 for **ld.b r0,(r0,0);** control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register = 0xA000 0000; write-back bus register (WBBR) = 0

Table 7. Steps for Reading from a Memory Location (Sheet 3 of 3)

Step	Task	Actions
9	State: Exit1-DR State: Update-DR After entering state Update-DR, the microcontroller will now go temporarily into user mode and then return to debug mode. Poll the OnCE status register (OSR) per the procedure "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note) to ensure the microcontroller has returned to debug mode before proceding to the next step.	Perform "Poling the MMC20xx OnCE Status Register" (elsewhere in this application note).
10	Select CPU scan chain register (CPUSCR) for **reading**.	Shift **0x8b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **execute** the instruction in the IR, and to **not exit** debug mode.
11	Shift out WBBR value, which contains the memory value to be read, out of the CPU scan chain register (CPUSCR).	Shift out contents of the CPU scan chain register (CPUSCR): write-back bus register (WBBR) = memory value to be read
	Restore the values of R0 and PC that were present when starting this example. Value in WBBR is loaded into R0.	
12	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
13		Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, enabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC must be set to PC saved when debug mode was entered. processor status register = 0xA000 0000; write-back bus register (WBBR) must be set to the value that was in R0 before starting this example

Writing to Memory Using the MMC20xx's JTAG/OnCE Port

This example will write an 8-, 16-, or 32-bit value to memory.

1. Using the write-back bus register (WBBR), save the original values in R0 and R1 so that they can be restored to their respective registers at the end of this example.
2. Using the write-back bus register (WBBR), write the address of the memory location to be written to R0.
3. Using the write-back bus register (WBBR), write the data to be written to R1.
4. Write an instruction for transferring the desired data to memory to the CPU scan chain register's (CPUSCR's) instruction register (IR). The MMC20xx will temporarily exit debug mode and execute the instruction.
5. Poll the OnCE status register to see if the MMC20xx has returned to debug mode from transferring the data (outside debug mode).
6. Using the WBBR, restore the original values to R0 and R1 that were saved in step 1.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, resides in the JTAG instruction register (IR) so that the MMC20xx is in debug mode.
- The original contents of the CPU scan chain register (CPUSCR) have been saved.
- The JTAG state machine is in the run-test/idle state.

Table 8. Steps for Writing to a Memory Location (Sheet 1 of 4)

Step	Task	Actions
These steps will: 1. Save the present value in MMC20xx register R0 before R0 is used for transferring data 2. Write instruction to the CPU scan chain register (CPUSCR) that will get value of R0		
1	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
2	Load the CPU scan chain register (CPUSCR).	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) = 0
	Read value of R0 from write-back bus register (WBBR).	
3	Select CPU scan chain register (CPUSCR) for **reading**.	Shift **0xcb** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **execute** the instruction in the IR, and to **not exit** debug mode.
4	Get value of R0 and save it for later use.	Shift out the contents of the CPU scan chain register (CPUSCR) which contains the write-back bus register (WBBR). The WBBR contains the value of R0. The MMC20xx's R0 should be restored to this value at the end of this example.
These steps will: 1 Save the present value in MMC20xx register R1 before R1 is used for transferring data 2. Write instruction to the CPU scan chain register (CPUSCR) that will get value of R1		
5	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.

Table 8. Steps for Writing to a Memory Location (Sheet 2 of 4)

Step	Task	Actions
6	Load the CPU scan chain register (CPUSCR).	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1211 for instruction **mov R1,R1** control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) = 0
	Read value of R1 from write-back bus register (WBBR).	
7	Select CPU scan chain register (CPUSCR) for **reading**.	Shift **0xcb** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **execute** the instruction in the IR, and to **not exit** debug mode.
8	Get value of R1 and save it for later use.	Shift out the contents of the CPU scan chain register (CPUSCR) which contains the write-back bus register (WBBR). The WBBR contains the value of R1. The MMC20xx's R1 should be restored to this value at the end of this example.
These steps will: 1. Copy the address of the memory location to be read from the external command controller to the MMC20xx's register R0		
9	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
10	Load the CPU scan chain register (CPUSCR).	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, enabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) is set to address of memory location to be read;
	Address of memory location to be read is now in R0	

Table 8. Steps for Writing to a Memory Location (Sheet 3 of 4)

Step	Task	Actions
These steps will: 1. Write to the memory location 2. Write an **st** instruction to the IR in the CPU scan chain register (CPUSCR)		
11	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to execute the instruction in the IR, and to not exit debug mode.
12	Shift values into the CPU scan chain register (CPUSCR). State: Shift-DR	Shift the following into the CPU scan chain register (CPUSCR): for a 32-bit memory location, IR = 0x9100 for **st.w R1,(R0,0)**; for a 16-bit memory location, IR = 0xD100 for **st.h R1,(R0,0)**; for an 8-bit memory location, IR = 0xB100 for **st.b R1,(R0,0)**; control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC = PC saved when debug mode is entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) = 0
13	State: Exit1-DR State: Update-DR After entering state Update-DR, the microcontroller will now go temporarily into user mode and then return to debug mode. Poll the OnCE status register (OSR) per the procedure "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note) to ensure the microcontroller has returned to debug mode before proceding to the next step.	Perform "Poling the MMC20xx OnCE Status Register" (elsewhere in this application note).
14	Select CPU scan chain register (CPUSCR) for **reading**.	Shift **0x8b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to execute the instruction in the IR, and to **not exit** debug mode.
	Value in R1 has now been written to the address that is in R0	

Table 8. Steps for Writing to a Memory Location (Sheet 4 of 4)

Step	Task	Actions
These steps will: 1. Restore the value of R0 that was present when this example was started and write the beginning value to the WBBR to then be copied to R0		
15	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
16		Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1200 for instruction **mov r0,r0** control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, enabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC must be set to PC saved when debug mode was entered. processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) must be set to value that was in R0 before starting this example
These steps will: 1. Restore the values of R1 and PC that were present when this example was started and write both values to the WBBR for later copying to R1 and to PC		
17	Select CPU scan chain register (CPUSCR) for **writing**.	Shift **0x4b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the IR, and to **not exit** debug mode.
18		Shift the following into the CPU scan chain register (CPUSCR): IR = 0x1211 for instruction **mov R1,R1** control state register (CTL) = 0xFFDB for feed forward Y (FFY) operand = 1, enabled, force PSR debug enable mode (FDB) bit = 1 for enabling debug mode; PC must be set to PC saved when debug mode was entered; processor status register (PSR) = 0xA000 0000; write-back bus register (WBBR) must be set to value that was in R1 before starting this example

Single-Stepping the MMC20xx

Two methods are used for single-stepping the MMC20xx:

- The first method uses the OnCE trace counter (OTC) and will trace two or more instructions.
- The second method uses the OTC in a special manner and will trace only one instruction.

The single-instruction method is discussed in **Table 9**.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, resides in the JTAG instruction register (IR) so that the MMC20xx is in debug mode
- The JTAG state machine is in the run-test/idle state.

Table 9. Steps for Single-Stepping One Instruction (Sheet 1 of 8)

Task No.	Task	Actions
1	**These steps will:** Get the contents of the CPU scan chain register (CPUSCR) and save them. Note that because of instruction pre-fetching, the PC (the PC value in the CPU scan chain register) will be 2 greater than the address of the instruction presently in the instruction register (IR). Save PC minus 2.	This step should be taken whenever entering debug mode.
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select CPU scan chain register (CPUSCR) for **reading**. State: Shift-IR	Shift **0x8b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **not execute** the instruction in the IR and to **not exit** debug mode.
	State: Exit1-IR	

Table 9. Steps for Single-Stepping One Instruction (Sheet 2 of 8)

Task No.	Task	Actions
1	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Shift out contents of the CPU scan chain register and save. Save (PC-2) for PC. State: Shift-DR	Shift out the contents of the CPU scan chain register (CPUSCR). Save all contents as is, except for PC. The value of PC minus 2 must be saved to compensate for instruction pre-fetching.
	State: Exit1-DR	
	State: Update-DR	
2	Clear the OnCE control register (OCR).	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select OnCE control register for **writing**. State: Shift-IR	Shift **0x4D** into the OnCE command (instruction) register (OCMR) to select the OnCE control register (OCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	
	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Clear OnCE control register (OCR). State: Shift-DR	Shift 0s into the 32-bit OnCE control register (OCR). The debug request (DR) and trace mode enable (TME) bits will be cleared.
	State: Exit1-DR	
	State: Update-DR	
3	**This step will:** Write the OnCE command register (OCMR) to select the CPU scan chain register (CPUSCR) for **writing**, to **not execute** the instruction in the instruction register (IR), and to **not exit** from debug mode, and shift new contents into the CPU scan chain register (CPUSCR).	
	State: Run-test/idle	

Table 9. Steps for Single-Stepping One Instruction (Sheet 3 of 8)

Task No.	Task	Actions
3	State: Select-DR scan	
	State: Select-IR scan	
	State: Capture-IR	
	Select the CPU scan chain register (CPUSCR) for **writing**. State: Shift-IR	Shift **0x0B** into the OnCE command (instruction) register (OCMR) for 7 TCKs to select the CPU scan chain register (CPUSCR) for **writing**, to **not execute** the instruction in the IR, and to **exit** debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command (instruction) register (OCMR) during the rising TCK edge upon entering this state.
	State: Update-IR	The value shifted into the OnCE command (instruction) register (OCMR) will appear at its parallel outputs.
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture -DR	
	State: Shift - DR	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x0001 for instruction **sync**; control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled; Force PSR debug enable mode (FDB) bit = 1 for debug enable mode; Set PC to the PC that was saved (PC minus 2) in task 1. Set processor status register (PSR) to 0xA000 0100 ORed with the PSR that was saved in task 1. Set write-back bus register (WBBR) to 0.
	State: Exit1-DR	
	State: Update-DR	
These steps will: 1. Select the bypass register, **execute** the instruction in the IR, and **not exit** from debug mode		
4	State: Run-test/idle	
	State: Select DR-scan	
	State: Select -IR scan	
	State: Capture-IR	

Table 9. Steps for Single-Stepping One Instruction (Sheet 4 of 8)

Task No.	Task	Actions
4	Select the bypass register (no register) for writing. State: Shift-IR	Shift **0x4c** into the OnCE command (instruction) register (OCMR) for 7 TCKs to select the bypass register (no register) for **writing**, to **execute** the instruction in the IR, and to **not exit** from debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command (instruction) register (OCMR) during the rising TCK edge upon entering this state.
	State: Update-IR	The value shifted into the OnCE command (instruction) register (OCMR) will appear at its parallel outputs.
	State: Run-Test/Idle	
	State: Select DR-Scan	
	State: Capture-DR	
	State: Exit1-DR (note that we skipped state Shift-DR)	
	State: Update-DR	
	The microcontroller will now go temporarily into user mode and then return to debug mode. Poll the OnCE status register (OSR) per the procedure "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note) to ensure the microcontroller has returned to debug mode before proceding to the next task.	Perform "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note).
5	These steps will read and save the contents of the CPU scan chain register (CPUSCR). PC and IP will respectively point to and contain the instruction to be single-stepped.	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select CPU scan chain register (CPUSCR) for reading, to **not execute** the instruction in the instruction register (IR), and to **not exit** debug mode. State: Shift-IR	Shift **0x0B** into the OnCE command register (OCMR) for 7 TCKs to select the CPU scan chain register (CPUSCR) for **reading**, to **not execute** the instruction in the IR, and to **not exit** debug mode.

Table 9. Steps for Single-Stepping One Instruction (Sheet 5 of 8)

Task No.	Task	Actions
5	State: Exit1-IR	Last data bit will be shifted into the OnCE command register (OCMR) during rising TCK edge upon entering this state.
	State: Update-IR	The value shifted into the OnCE command register (OCMR) will appear at its parallel outputs.
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Shift out the contents of the CPU scan chain register (CPUSCR), and save its contents. State: Shift-DR	Save the contents of the CPU scan chain register (CPUSCR) for later use.
	State: Exit1-DR	
	State: Update-DR	
6	**These steps will:** Select the CPU scan chain register (CPUSCR) for **writing**, to **execute** the instruction in the instruction register (IR), and to **not exit** from debug mode. We will load the CPUSCR with values and modi ed v alues from step 4.	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	State: Shift-IR	Shift **0x0B** into the OnCE command register (OCMR) for 7 TCKS to select the CPU scan chain register (CPUSCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command register (OCMR) during rising TCK edge upon entering this state.
	State: Update-IR	The value shifted into the OnCE command (instruction) register (OCMR) will appear at its parallel outputs.
	To shift data into the CPU scan chain register (CPUSCR).	
	State: Run-test/idle	

Table 9. Steps for Single-Stepping One Instruction (Sheet 6 of 8)

Task No.	Task	Actions
6	State: Select DR-scan	
	State: Capture-DR	
	State: Shift-DR	Shift the following into the CPU scan chain register (CPUSCR): IR = value of IR read and saved in task 4. control state register (CTL) = 0xFEDB for: feed forward Y (FFY) operand = 0, disabled; forcing PSR debug enable mode (FDB) bit = 1 to place processor in debug enable mode; Set PC to PC value that was saved in task 4. Set the processor status register (PSR) to 0xA000 0100 ORed with the PSR that was saved in task 1. Set the write-back bus register (WBBR) to 0.
	State: Exit1-DR	
	State: Update-DR	
7	**These steps will:** 1.Single-step the intended instruction. Write the OnCE command register (OCMR) to select the bypass register to **execute** the instruction in the instruction register (IR) and to **not exit** from debug mode.	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture IR	
	State: Shift-IR	Shift **0x4c** into the OnCE command (instruction) register (OCMR) for 7 TCKs to select the bypass register (no register) for **writing**, to **execute** the instruction in the IR (the instruction to be single-stepped), and to **not exit** debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command (instruction) register (OCMR) during the rising TCK edge upon entering this state.
	State: Update-IR	The value shifted into the OnCE command (instruction) register (OCMR) will appear at its parallel outputs.

Table 9. Steps for Single-Stepping One Instruction (Sheet 7 of 8)

Task No.	Task	Actions
7	The microcontroller will now go temporarily into user mode and then return to debug mode. Poll the OnCE status register (OSR) per the procedure "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note) to ensure the microcontroller has returned to debug mode before proceding to the next task.	Perform "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note).
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	State: Exit1-DR (note that we skipped state Shift-DR)	
	State: Update-DR	
8	Clear the OnCE trace counter (OTC).	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select the CPU scan chain register (CPUSCR) for **writing**. State: Shift-IR	Shift **0x03** into the OnCE command (instruction) register (OCMR) for 7 TCKs to select the OnCE trace counter (OTC) for **writing**, to **not execute** the instruction in the IR, and to **not exit** from debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command (instruction) register (OCMR) during the rising TCK edge upon entering this state.
	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	State: Shift-DR	Shift 0s into 16-bit OnCE trace counter (OTC).
	State: Exit1-DR	
	State: Update-DR	

Table 9. Steps for Single-Stepping One Instruction (Sheet 8 of 8)

Task No.	Task	Actions
9	Clear the OnCE control register (OCR).	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select OnCE control register for **writing**. State: Shift-IR	Shift **0x4D** into the OnCE command (instruction) register (OCMR) to select the OnCE control register (OCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	
	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-Scan	
	State: Capture-DR	
	Clear the OnCE control register (OCR). State: Shift-DR	Shift 0s into the 32-bit OnCE control register (OCR). The debug request (DR) and trace mode enable (TME) bits will be cleared.
	State: Exit1-DR	
	State: Update-DR	

Setting a Breakpoint and Exiting to User Mode

The two methods for setting/clearing breakpoints use:

- Hardware breakpoint logic in the OnCE controller
- Software breakpoint instruction (**bkpt**)

Each method has its advantages. For example, the hardware breakpoint has a wider range of addresses and access types and cannot be altered by an application program, but it is limited to one breakpoint. On the other hand, software breakpoints have no count limitation, but must be set on op-code fetch addresses and can be masked (when in supervisor mode).

This example shows how to set hardware breakpoints.

Assumptions —

- The instruction ENABLE_MCU_OnCE, 0x3, resides in the JTAG instruction register (IR) so that the MMC20xx is in debug mode.
- The contents of the CPU scan chain register (CPUSCR) have been saved.
- The JTAG state machine is in the run-test/idle state.

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 1 of 7)

Task No.	Task	Note
1	These steps will: Get the contents of the CPU scan chain register (CPUSCR) and save them. Note that because of instruction pre-fetching, the PC (the PC value in the CPU scan chain register) will be 2 greater than the address of the instruction presently in the instruction register (IR). Save PC minus 2.	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select CPU scan chain register (CPUSCR) for **reading**. State: Shift-IR	Shift **0x8b** into the OnCE command (instruction) register (OCMR) to select the CPU scan chain register (CPUSCR) for **reading**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	
	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Shift out contents of the CPU scan chain register and save. Save (PC-2) for PC. State: Shift-DR	Shift out the contents of the CPU scan chain register (CPUSCR). Save all contents as is, except for PC. The value of PC minus 2 must be saved to compensate for instruction pre-fetching.

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 2 of 7)

Task No.	Task	Note
1	State: Exit1-DR	
	State: Update-DR	
2	Clear the OnCE trace counter (OTC).	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select the CPU scan chain register (CPUSCR) for **writing**. State: Shift-IR	Shift **0x03** into the OnCE command register (OCMR) for 7 TCKs to select the OnCE trace counter (OTC) for **writing**, to **not execute** the instruction in the IR, and to **not exit** from debug mode.
	State: Exit1-IR	Last data bit will shift into the OnCE command (instruction) register (OCMR) during rising TCK edge upon entering this state.
	State: Update-IR	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	State: Shift-DR	Shift 0s into the 16-bit OnCE trace counter (OTC).
	State: Exit1-DR	
	State: Update-DR	
3	**These steps will:** 1. Clear the OnCE control register (OCR)	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IR-scan	
	State: Capture-IR	
	Select OnCE control register for **writing**. State: Shift-IR	Shift **0x4D** into the OnCE command (instruction) register (OCMR) to select the OnCE control register (OCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	
	State: Update-IR	

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 3 of 7)

Task No.	Task	Note
3	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Clear the OnCE control register (OCR). State: Shift-DR	Shift 0s into the 32-bit OnCE control register (OCR). The debug request (DR) and trace mode enable (TME) bits will be cleared.
	State: Exit1-DR	
	State: Update-DR	
4	Now, will load the CPU scan chain register	
	State: Run-test/idle	
	State: Select-DR scan	
	State: Select-IR scan	
	State: Capture-IR	
	Select the CPU scan chain register (CPUSCR) for **writing**. State: Shift-IR	Shift **0x0B** into the OnCE command register (OCMR) for 7 TCKs to select the CPU scan chain register (CPUSCR) for **writing**, to **not execute** the instruction in the IR, and to **not exit** debug mode.
	State: Exit1-IR	Last data bit will be shifted into the OnCE command (instruction) register (OCMR) during the rising TCK edge upon entering this state.
	State: Update - IR	The value shifted into the OnCE command (instruction) register (OCMR) will appear at its parallel outputs.
	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 4 of 7)

Task No.	Task	Note
4	State: Shift-DR	Shift the following into the CPU scan chain register (CPUSCR): IR = 0x0001 for instruction **sync**; control state register (CTL) = 0xFEDB for feed forward Y (FFY) operand = 0, disabled; Force PSR debug enable mode (FDB) bit = 1 for debug enable mode; Set PC to the PC that was saved (PC minus 2) in task 1. Set processor status register (PSR) to 0xA000 0100 ORed with the PSR that was saved in task 1. Set write-back bus register (WBBR) to 0.
	State: Exit1-DR	
	State: Update-DR	
5	Now, repeat this task until each of these registers is loaded: Breakpoint address base register A (BABA) Breakpoint address base register B (BABB) Breakpoint address base Breakpoint address mask register A (BAMA) Breakpoint address mask register B (BAMB) Memory breakpoint counter A (MBCA) Memory breakpoint counter B (MBCB) OnCE trace counter (OTC) OnCE control register (OCR)	
	Write instruction register	
	State: Run-test/idle	
	State: Select DR-scan	
	State: Select IRscan	
	State: Capture-IR	

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 5 of 7)

Task No.	Task	Note
5	Select the OnCE control (instruction) register for **writing**. State: Shift-IR	If setting up: breakpoint address base register A (BABA), shift **0x07** into the OnCE command (instruction) register, If setting up: breakpoint address base register B (BABB), shift **0x08** into the OnCE command (instruction) register, If setting up: breakpoint address mask register A (BAMA), shift **0x09** into the OnCE command (instruction) register, If setting up: breakpoint address mask register B(BAMB), shift **0x0a** into the OnCE command (instruction) register, If setting up: memory breakpoint counter A (MBCA), shift **0x04** into the OnCE command (instruction) register, If setting up: memory breakpoint counter B (MBCB), shift **0x05** into the OnCE command (instruction) register, If setting up: OnCE trace counter (OTC), shift **0x03** into the OnCE command (instruction) register, If setting up: OnCE control register (OCR), shift **0x0d** into the OnCE command (instruction) register, To select that register for **writing**, to **not execute** the instruction in the instruction register, and to **not exit** debug mode
	State: Exit1-IR	
	State: Update-IR	
	Write data register.	

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 6 of 7)

Task No.	Task	Note
5	State: Run-test/idle	
	State: Select DR-scan	
	State: Capture-DR	
	Load a user-supplied value into register being set up. State: Shift-DR	If setting up the BABA, shift a user-supplied value into 32-bit BABA register, LSB rst. If setting up the BABB, shift a user-supplied value into the 32-bit BABB register, LSB rst. If setting up the BAMA, shift a user-supplied value into 32-bit BAMA register, LSB rst. If setting up the BAMB, shift a user-supplied value into 32-bit BAMB register, LSB rst. If setting up the MBCA, shift a user-supplied value into 16-bit MBCA register, LSB rst. If setting up the MBCB, shift a user-supplied value into 16-bit MBCB register, LSB rst. If setting up the OTC, shift a user-supplied value into the 16-bit OTC register, LSB rst. If setting up the OCR, shift a user-supplied value into the 32-bit OCR register, LSB rst.
	State: Exit1-DR	
	State: Update-DR	
6	**These steps will:** 1. Exit out of debug mode.	
	State: Shift-IR	Shift **0xec** into the OnCE command (instruction) register (OCMR) to select the bypass register (no register selected) for **reading**, to **execute** the instruction in the instruction register (IR), and to **exit** debug mode. Since no instructions have been executed since a **sync** instruction was loaded into the IR (via the CPU scan chain register (CPUSCR), execute that **sync** instruction and then **exit** debug mode with these steps.
	State: Exit1-IR	Last bit of data is shifted into the OnCE IR.

Table 10. Arming Breakpoint Logic and Exiting to User Mode (Sheet 7 of 7)

Task No.	Task	Note
6	State: Update-IR	The MMC20xx OnCE command (instruction) register (OCMR) (and the JTAG TAP IR) is (are) updated. Contents of the OCMR are presented at the OCMR's parallel outputs.
	State: Select DR scan	
	State: Capture-DR	
	State: Shift-DR	**Read** the bypass (no register selected) register with 8 TCK cycles, with test mode select (TMS) set to 0 when clocking for the rst 7 TCK cycles, and TMS set to 1 before and while clocking out the eighth 0.
	State: Exit1-DR	Last (eighth) bit of data is shifted out during rising edge of TCK while entering this state.
	State: Update-DR	Parallel outputs of the MMC20xx's CPU scan chain register (CPUSCR) are updated.
	State: Run-test/idle	
	State: Update-DR	MMC20xx will exit debug mode and enter user mode.
	State: Run-test/idle	
7	Now in user mode, the program to be breakpointed can be started. When the breakpoint conditions are met, the MMC20xx will return to debug mode. MMC20xx will enter debug mode immediately after executing a breakpointed op-code or after fetching data at a breakpointed address.	
8	**This step will:** Check for entry into debug mode.	
	Poll the OnCE status register (OSR) per the procedure "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note) to ensure the microcontroller has returned to debug mode before proceding.	Perform "Polling the MMC20xx OnCE Status Register" (elsewhere in this application note).

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor