Intel™ Quark® Microcontroller D2000 Debug Operations

User Guide

November 2016

Document Number: 333241-002EN

Contents

1.0 Introduction 7

1.1 Terminology 8

2.0 Chip Level Tap Controller 9

2.1 CLTAPC Instruction Table 9

2.2 CLTAPC Data Register Table 9

2.3 CLIDCODE 10

2.4 CLBYPASS 10

2.4.1 CLTAPC_SELECT 10

2.4.2 CLTAPC_CPU_VPREQ 11

2.4.3 CLTAPC_CPU_TAPSTATUS 11

2.4.4 CLTAPC_CPU_VPRDY 12

2.4.5 CLTAPC_TAPNW_STATUS 12

3.0 Putting It All Together 13

3.1 Initial JTAG Discovery 13

3.2 Check Host Powergood 13

3.3 Add IA Core TAP to the JTAG Chain 13

3.4 Verify IA Core IDCODE 14

3.5 Flash One-Time-Programmable (OTP) 14

4.0 IA Core TAP 16

4.1 IA Core TAP Instruction Table 16

5.0 Run Control 17

5.1 Introduction to Probe Mode 17

5.2 Probe Mode Entry 17

5.3 Probe Mode Exit 18

5.4 Reset Break 18

5.5 TAPSTATUS Register 18

5.6 Accessing Architectural Registers 19

5.6.1 Submitting Instructions to the Core 19

5.6.2 EIP Management 20

5.6.3 DR7 Management 20

5.6.4 WRITEPIR Register Format 21

5.6.5 Register Read 21

5.6.6 Register Write 21

5.6.7 Special Cases for Register Access 22

5.6.8 Checking for HALT State 22

5.6.9 Pseudo Opcodes for Architectural Register Access 22

5.6.10 Probe Mode Control Register 23

5.6.11 Accessing Model Specific Registers (MSR) 24

5.7 Reading and Writing Memory 24
5.7.1 Management of Architectural Registers for Memory Access 24
5.7.2 CSLIMIT Considerations 25
5.7.3 Memory Read 26
5.7.4 Memory Write 26
5.8 Reading and Writing I/O Ports 26
5.8.1 I/O Read Example in Probe Mode 26
5.8.2 I/O Write Example in Probe Mode 26
5.9 Hardware Breakpoints 27
5.10 Software Breakpoints 27
5.11 Single Step 27
5.12 Redirections into Probe Mode 27
5.12.1 Shutdown Break 27

6.0 Flash Programming Options 28
6.1 Flash Overview 28
6.2 Flash Controller 28
6.3 Supported Operations 28
6.3.1 Mass Erase 28
6.3.2 Page Erase 29
6.3.3 Data Program 29

7.0 Memory Mapped Registers 30
7.1 Register Field Access Types 30
7.2 Flash Controller 0 31
7.2.1 Register Summary 31
7.2.2 TMG_CTRL (TMG_CTRL) 31
7.2.3 ROM_WR_CTRL (ROM_WR_CTRL) 32
7.2.4 ROM_WR_DATA (ROM_WR_DATA) 34
7.2.5 FLASH_WR_CTRL (FLASH_WR_CTRL) 34
7.2.6 FLASH_WR_DATA (FLASH_WR_DATA) 35
7.2.7 FLASH_STTS (FLASH_STTS) 36
7.2.8 CTRL (CTRL) 36
7.3 System Control/Status 37
7.3.1 Register Summary 37
7.3.2 Hybrid Oscillator Configuration 1 (OSC0_CFG1) 37
7.3.3 External Clock Control (CCU_EXT_CLOCK_CTL) 39
7.3.4 System Clock Control (CCU_SYS_CLK_CTL) 40
7.3.5 Reset Control (RSTC) 40
7.3.6 Reset Status (RSTS) 41

Figures

Figure 1. Intel® Quark™ Microcontroller D2000 7
Figure 2. OTP Bit sTAP Unlock Flow 15

Tables

Table 1. Terminology 8
Table 2. CLTAPC TAP Instructions 9
Table 3. CLTAPC Data Register Table 9
Table 4. CLTAPC_SELECT 10
Table 5. CLTAPC_CPU_VPREQ 11
Table 6. CLTAPC_CPU_VPREQ 11
Table 7. CLTAPC_TAPNW_STATUS 12
Table 8. TAP Instructions 16
Table 9. TAPStatus Data Register 18
Table 10. Register Access PIR Values 22
Table 11. PMCR Description 23
Table 12. DS Selector Values for Memory Access 25
Table 13. Register Field Access Types 30
Table 14. Flash Controller 0 Registers 31
Table 15. Flash Timing Control Register 31
Table 16. ROM Write Control Register 32
Table 17. ROM Write Data Register 34
Table 18. DCCM Base Address Register 34
Table 19. Flash Write Data Register 35
Table 20. Flash Status Register 36
Table 21. Control Register 36
Table 22. System Control/Status Registers 37
Table 23. Hybrid Oscillator Configuration Register 37
Table 24. External Clock Control Register 39
Table 25. System Clock Control Register 40
Table 26. Reset Control Registers 40
Table 27. Reset Status Register 41

Revision History

Date	Revision	Description
November 2016	002	System diagram updated.
October 2015	001	Initial release.

§

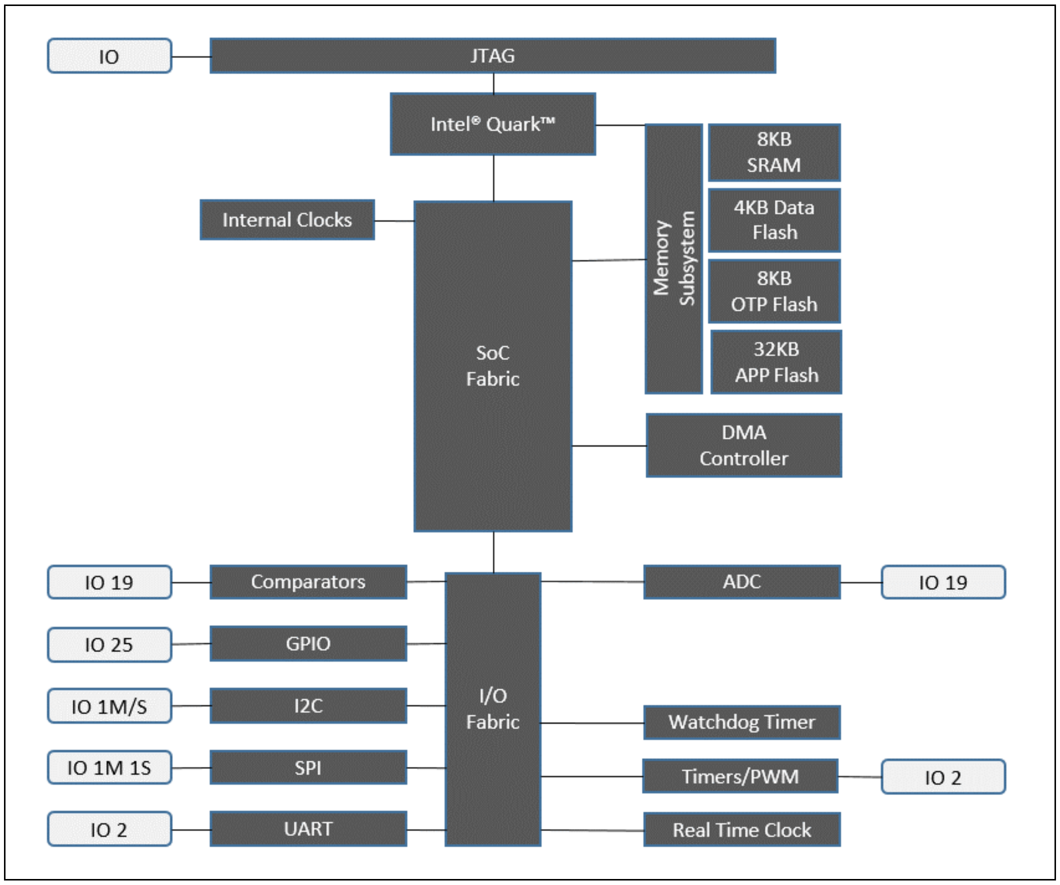
1.0 Introduction

The Intel® Quark™ Microcontroller D2000 is an ultra-low power Intel Architecture (IA) SoC that integrates a Quark™ SE processor core, Sensor Subsystem, Memory Subsystem with on-die volatile and non-volatile storage, Pattern Matching Accelerator and I/O interfaces into a single system-on-chip solution.

This document assumes that the reader has some familiarity with JTAG based debug tools and the use of JTAG for run control of an execution core.

This document provides details on JTAG based debug for any product based on the Intel® Quark™ Microcontroller D2000.

Figure 1. Intel® Quark™ Microcontroller D2000



1.1 Terminology

Table 1. Terminology

Term	Description
CLTAPC	Chip level TAP Controller. This is the top level standard compliant TAP controller for the SoC.
Debug Software	Generic term for software that controls a hardware probe connected to the JTAG pins.
JTAG	“Joint Test Action Group” of the IEEE. This is now a generic term to refer to the TAP and the pins used for communication with TAPs.
PIR	Probe Mode Instruction Register
SoC	System on chip
TAP	Test Access Port as defined by the IEEE 1149.1-1990 (including IEEE 1149.1a-1993), “IEEE Standard Test Access Port and Boundary-Scan Architecture”
TDI	TAP Data In; the serial data input pin for the TAP chain
TDO	TAP Data Out; the serial data output pin for the TAP chain
TDR	TAP Data Register; a serial TAP data register selected by a TAP instruction

2.0 Chip Level Tap Controller

The Intel® Quark™ Microcontroller D2000 has the standard set of JTAG pins, TCLK, TDI, TDO, TMS, and TRST# on the package. These are routed to a debug header on the system board.

The SoC exposes a single IEEE compliant TAP by default, called the 'Chip-Level TAP Controller' (CLTAPC). This TAP provides some basic system status and has the ability to 'add' child TAP controllers to the serial JTAG chain.

Debug Software tools that wish to provide run control for the Intel® Quark™ Microcontroller D2000 must interact with the CLTAPC to add the CPU core. The Debug Software may also use TAP data registers (TDRs) in the CLTAPC to monitor system status. The CLTAPC provides some features related to run control of the core which are described in detail in Chapter 5.0, "Run Control".

2.1 CLTAPC Instruction Table

Table 2. CLTAPC TAP Instructions

Function/Category	Instruction Mnemonic	Opcode (8 bits)	TDR Name
SOC IDCODE	CLIDCODE	0x02	CLIDCODE
BYPASS	CLBYPASS	0xFF	CLBYPASS
TAP NETWORK	CLTAPC_SELECT	0x11	CLTAPC_SELECT
IA Run Control	CLTAPC_CPU_VPREQ	0x70	CLTAPC_CPU_VPREQ
	CLTAPC_CPU_TAPSTATU S	0x72	CLTAPC_CPU_TAPSTATUS
	CLTAPC_CPU_VPRDY	0x71	CLTAPC_CPU_VPRDY
TapNW Status	CLTAPC_TAPNW_STATUS	0x6B	CLTAPC_TAPNW_STATUS
RESERVED	-	All other values	-

2.2 CLTAPC Data Register Table

Table 3. CLTAPC Data Register Table

TDR Name	TDR Length (Bits)	Reset Mechanism	Access
CLIDCODE	32	TRST/TLRS/RST_N_PAD	Read-Only
CLBYPASS	1	TRST/TLRS/RST_N_PAD	Read-Write
CLTAPC_SELECT	12	TRST/TLRS/RST_N_PAD	Read-Write

TDR Name	TDR Length (Bits)	Reset Mechanism	Access
CLTAPC_CPU_VPREQ	8	RST_N_PAD	Read-Write
CLTAPC_CPU_TAPSTATUS	8	RST_N_PAD	Read-Only
CLTAPC_CPU_VPRDY	1	RST_N_PAD	Read-Write
CLTAPC_TAPNW_STATUS	32	RST_N_PAD	Read-Only

NOTE: Test-Logic-Reset state = TLRS

2.3 CLIDCODE

The CLTAPC IDCODE register is 32 bits in size. This value may be used by Debug Software to confirm there is a working JTAG connection to the board and to confirm the identity of the SoC.

The CLTAPC IDCODE is 0x0E786013.

2.4 CLBYPASS

This is the IEEE standard BYPASS data register; it is one bit in size.

2.4.1 CLTAPC_SELECT

This data register contains bits that control the presence of the children TAPs in the SoC on the JTAG chain.

Table 4. CLTAPC_SELECT

CLTAPC_SELECT			
Bit Nr	**Name**	**Reset Value**	**Comments**
1:0	CPUCORE_TAP_SEL	2'b00	2'b00 = Isolated 2'b01 = Normal 2'b10 = Excluded 2'b11 = Shadow
11:2	RESERVED	8'h000	RESERVED

When bits 1:0 are set to the value 01 by Debug Software, the CPU Core TAP is added to the JTAG chain immediately after the CLTAPC.

The chain order becomes TDI →CLTAPC →CPU_CORE_TAP →TDO, when CLTAPC_SELECT=12’b0000_0000_0001.

All other bits in this register are reserved.

2.4.2 CLTAPC_CPU_VPREQ

The CLTAPC has control over an internal PREQ wire connected to the CPU core. This provides Debug Software the ability to use some PREQ based debug features when connected using JTAG pins alone.

This data register contains bits that control the assertion of the internal 'PREQ' signal to the SoC and reset break behavior.

Table 5. CLTAPC_CPU_VPREQ

CLTAPC_CPU_VPREQ			
Bit Nr	**Name**	**Reset Value**	**Comments**
0	assert_vpreq	1'b0	Assert PREQ to Core
1	enable_preq_on_early_LMT_reset	1'b0	Enable PREQ assertion on Early LMT Reset
7:2	RESERVED	4'h0	RESERVED

When bit 0 is set to 1 by Debug Software, the PREQ signal connected to the core is asserted. This signal is an external request to the core that causes the core to break and enter Probe Mode.

Enable resetbreak for the IA core by setting bit 'enable_preq_on_early_LMT_reset' (bit 1). When this bit is set and the IA core receives a RESET event, the IA Core immediately breaks and enters Probe Mode instead of executing the first instruction. The core enters Probe Mode with the CS register set to 0xF000 and the EIP register set to 0x0000FFF0. The instruction located at this address is not executed when a resetbreak triggers the entry to Probe Mode.

All other bits in this register are reserved.

2.4.3 CLTAPC_CPU_TAPSTATUS

This data register contains status bits related to the Run Control features in the CLTAPC_CPU_VPREQ register.

Table 6. CLTAPC_CPU_VPREQ

CLTAPC_CPU_TAPSTAT			
Bit Nr	**Name**	**Reset Value**	**Comments**
0	vpreq_asserted	1'b0	PREQ was asserted via one of the sources in CPU_VPREQ

CLTAPC_CPU_TAPSTAT			
1	prdy_asserted	1'b0	PRDY was asser ted by the Core
2	preq_asserted_via_early_LMT_reset	1'b0	PREQ was asserted via early LMT reset.
6:3	RESERVED	4'b0000	RESERVED
7	pm_debugon_asserted	1'b0	pm_debugon was asserted by LMT

2.4.4 CLTAPC_CPU_VPRDY

This data register is used to reset the status bits in the CLTAPC_CPU_TAPSTATUS register when the register is written to. It is a single bit in size, but the value written does not matter.

2.4.5 CLTAPC_TAPNW_STATUS

This data register contains a mixture of status bits.

The IA Core is awake and running only when the HOST_power_ok, (bit 7) is 1.

Before adding IA Core TAP to the network (by using CLTAPC_SELECT), bit 7 of CLTAPC_TAPNW_STATUS must be HIGH.

Table 7. CLTAPC_TAPNW_STATUS

CLTAPC_TAPNW_STATUS			
Bit Nr	**Name**	**Reset Value**	**Comments**
6:0	0	0	0
7	HOST_power_ok	0	When 1 the HOST power rail is on
9	OTP_bit_status	0	Value of the OTP bit
31:10	Reserved	0	

§

3.0 Putting It All Together

This section uses the reference information from Chapter 2.0, "Chip Level Tap Controller" to put together all of the steps required for Debug Software to prepare the target for IA Core run control.

3.1 Initial JTAG Discovery

Debug Software can use two methods to confirm that it has a good connection to the Chip-Level TAP Controller before moving on to debug the SoC:

- Using a TAP Reset
  - Issue a JTAG reset by asserting and then de-asserting the TRST# pin, or by holding TMS to 1 for five TCLK cycles. Both mechanisms are part of the IEEE 1149.1 standard for a TAP reset.
  - Use a 32 bit or DR Shift using the TAP finite state machine and examine the data captured on TDO. It should match the SoC CLTAPC IDCODE (0x0E765013).
- Without a TAP Reset
  - Shift in the IR Opcode for the CLTAPC IDCODE data register (0x2) and then a 32 bit DR Shift. The data collected on TDO should match the CLTAPC IDCODE.

3.2 Check Host Powergood

Once the Debug Software has verified that it has a working connection to the SoC, it can check to see if the Host is powered.

Using a JTAG DR Shift to read the CLTAPC_TAPNW_STATUS data register, check that bit 7, `HOST_power_ok' is set to 1. When this bit is a 1, it is safe to access the Core's TAP.

3.3 Add IA Core TAP to the JTAG Chain

The IA Core TAP is added to the JTAG chain by setting the 'select' bits in the CLTAPC_SELECT data register. The data value used to add the IA core is as follows: 12'b0000_0000_0001. The CPU TAP is added to the JTAG chain when the TAP finite state machine (FSM) enters Update-DR. The Debug Software should move the TAP FSM to Run-Test/Idle immediately after writing any bits in CLTAPC_SELECT.

The FSM should stay in Run-Test/Idle for at least two TCK clock cycles before accessing any other TDR.

Until the select bits are cleared, Debug Software must be prepared to account for the fact that there are two TAPs on the chain.

3.4 Verify IA Core IDCODE

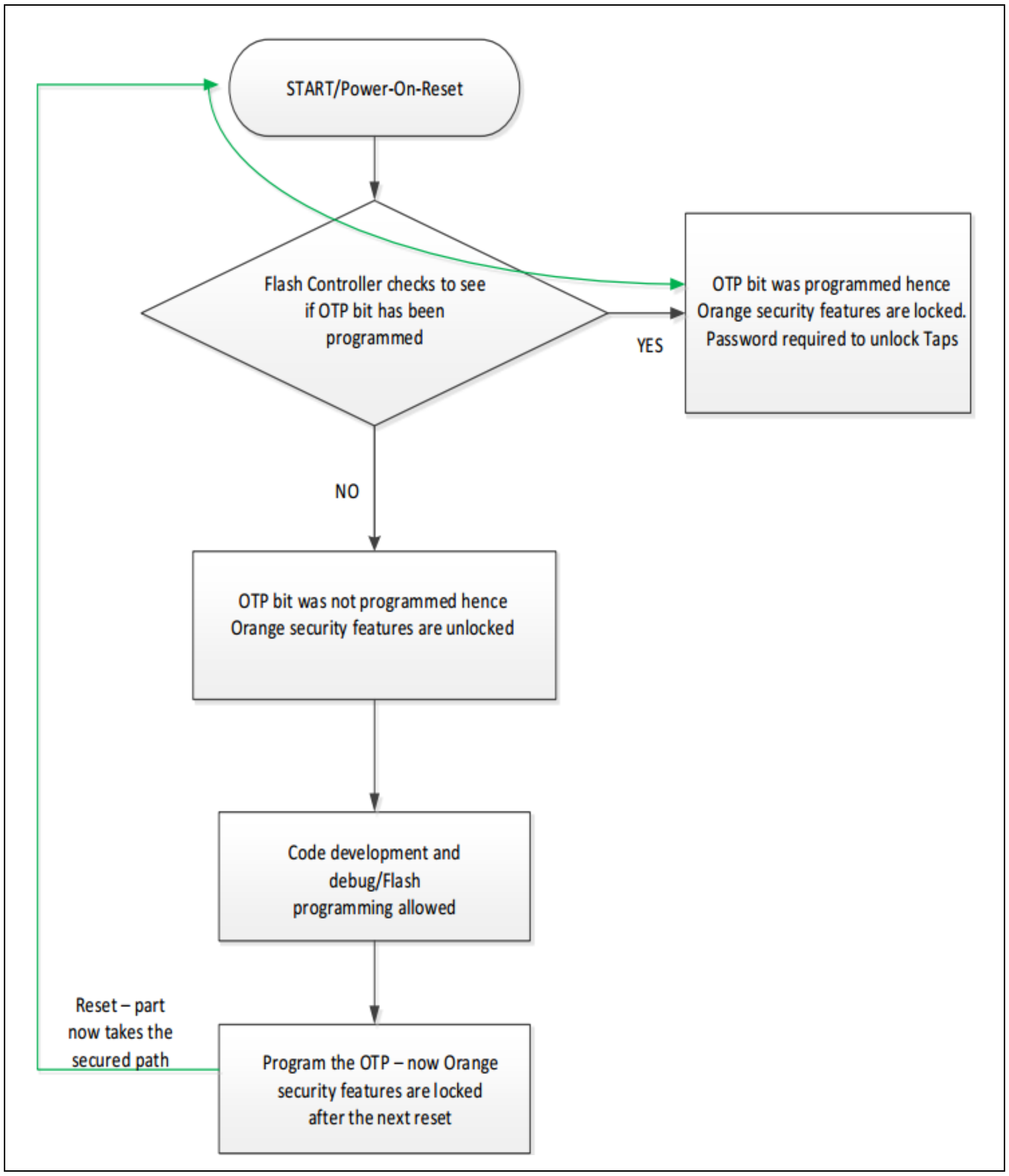
After completing the steps above, the Debug Software can then verify that the IA Core TAP and is in the JTAG chain.

The 8 bit IR Opcode for the IA Core IDCODE data register is 0x2. The IA Core IDCODE value is 0x38289013.

3.5 Flash One-Time-Programmable (OTP)

Security is managed via a combination of a JTAG Password and a Flash memory One-Time-Programmable (OTP) bit. The customer profile is controlled from an OTP bit output by the flash controller. The intent here is that a customer writes the OTP bit after development and debug of its flash images.

Figure 2. OTP Bit sTAP Unlock Flow


START/Power-On-Reset
Flash Controller checks to see if OTP bit has been programmed
YES
OTP bit was programmed hence Orange security features are locked. Password required to unlock Taps
NO
OTP bit was not programmed hence Orange security features are unlocked
Code development and debug/Flash programming allowed
Reset – part now takes the secured path
Program the OTP – now Orange security features are locked after the next reset

§

4.0 IA Core TAP

4.1 IA Core TAP Instruction Table

The IA Core TAP uses an 8-bit instruction register. The following table describes all IR opcode encodings.

Table 8. TAP Instructions

IR Opcode	TAP Command	Description	DR Size (Bits)	Values
0x02	IDCODE	The IEEE 1149 compliant IDCODE	32	The idcode value is 0x38289013
0x03	SUBMITPIR	Submit the contents of the PIR to the core	0	
0x04	PM_ACCESS	Enter and Exit Probe Mode (Details in Section 5)	1	1: Enter Probe Mode 0: Exit Probe Mode
0x06	WRPIR	Write instruction opcodes to the PIR	64	
0x08	PDR_ACCESS	Select the PDR chain for read or write	32	
0x0B	TAPSTATUS	TAP Status register	32	See Table 10.
0x0C	SLVIDCODE	Slave IDCODE	32	The slvidcode value is 0x08000003
0x35	PM_TAP_RB	Setup TAP Reset Break Request into Probe Mode	1	
0xFF	BYPASS	IEEE 1149 BYPASS	1	
All other values	BYPASS	IEEE 1149 BYPASS	1	

§

5.0 Run Control

5.1 Introduction to Probe Mode

Probe Mode is a debug mode of the processor in which the normal execution sequence is interrupted. The processor enters a dormant state where the architectural state can be viewed and modified. The state extraction/modification is performed with the help of the TAP. Special hardware/software products are developed to make use of Probe Mode.

The processor can be made to enter Probe Mode in various ways. The PREQ (Probe Mode Request) pin, called PREQ#, is pulsed to cause Probe Mode entry. The processor can also be configured to enter Probe Mode when certain events occur (redirecting to Probe Mode).

Once the processor enters Probe Mode and is ready to accept commands, it pulses the PRDY (Probe Mode Ready) pin, called PRDY#. Debug Software can then issue Probe Mode instructions (to be described later) to access architectural state. When an instruction is submitted while the processor is in Probe Mode, PRDY# is pulsed upon completion of the instruction.

While in Probe Mode, all communication between the Debug Software and the processor is done through the TAP. The Probe Mode instructions are serially sent via the WRSUBPIR instruction into the TAP that then transfers the instruction bytes to the fetch unit.

Probe Mode Entry on Intel processors is an interrupt/exception style event and is prioritized with other events. The priority is very low.

5.2 Probe Mode Entry

Probe Mode may be entered asynchronously (from the Core's perspective) by the request of debug software using the TAP. Both the CLTAPC and the Core TAP in the SoC have TAP instructions for requesting entry into Probe Mode.

The Core will pulse PRDY# for two bus clock cycles and this pulse is latched by the CLTAPC_CPU_TAPSTATUS TDR.

The Core may be configured to enter Probe Mode when a code or data breakpoint match occurs. These appear to be asynchronous entries in to Probe Mode from the perspective of the Debug Software since it cannot predict when or if the breakpoint match will occur.

Any standard Debug Exception (#DB) from the core may be configured to cause an entry to Probe Mode instead of allowing the exception handler to run. See Section 5.6.10, "Probe Mode Control Register", for full details.

The PROBEMODE TAP instruction offers a TAP-based entry method. To use the TAP-based Probe Mode entry, write '1 to the PROBEMODE data register. The core will automatically redirect to Probe Mode after a single step, hardware, or software breakpoint. All entries to Probe Mode are signaled to the Run Control Hardware via the PRDY# which is latched by the CLTAPC_CPU_TAPSTATUS TDR.

5.3 Probe Mode Exit

Probe Mode exit is accomplished by writing a '0 to the PROBEMODE TAP instruction's data register. Reset will also cause the core to leave Probe Mode.

5.4 Reset Break

To debug firmware from the first instruction, you must use the reset break capability. The Quark™ SE Core supports reset break when the Debug Tool is connected using the JTAG pins.

Mechanisms for reset break: use the CLTAPC_CPU_VPREQ TDR, as described in Section 2.4.4, "CLTAPC_CPU_VPRDY".

5.5 TAPSTATUS Register

The TAPSTATUS register is defined in the following table.

Table 9. TAPStatus Data Register

Bit Field	Description
31:20	Reserved
19	MCA Support: Indicates MCA (Machine Check Architecture) logic support by the core. This bit will read 0 since this feature is not enabled for this SoC.
18	Reset Break Request Indicator: Indicates prior TAP based reset break request. This bit automatically gets cleared when TAP RESET BREAK DR register is cleared by debugger.
17	Processor functional reset: Value of 1 indicates that processor functional reset is asserted.
16	Stop Clock Request Indicator: Value of 1 indicates that stop clock request input to the processor is active.
15:10	Reserved: Will read out 0s.
9	Memory operation in process: Value of 1 indicates that the core is executing an instruction submitted via de PIR.

Bit Field	Description
8	MCE Probe Mode Break: Value of 1 indicates probe mode entry is due to MCE break into probe mode. This bit will always read 0. This functionality is not enabled for this SoC.
7	MCE Probe Mode Break Enabled: Value 1 indicates that MCE Break into probe mode is currently enabled. It will always read 0 since MCE is not supported.
6	Shutdown Break Occurred. This bit is set to '1 when a shutdown break was the cause for the break.
5	Shutdown Break is enabled. This is a copy of PMCR[1]
4	Software Breakpoints: If '1, the core supports software breakpoints installed by debug software.
3	Probe Mode Redirection: Value 1 indicates that Single Step and Hardware Breakpoint into probe mode is currently enabled. Holds value of PMCR[0].
2	Probe Mode In Progress: Held high while in Probe Mode and after register state has been saved to shadow SRAM.
1	Probe Mode Request Pending: Held high when there is a pending probe mode request.
0	Reserved. Will read out 0.

5.6 Accessing Architectural Registers

The architectural registers are accessible upon entry to Probe Mode and may be accessed in any order. Debug software must cache the values of any register that may be changed during Probe Mode operations like memory access. These cached values must be written to the architectural registers prior to releasing the processor from Probe Mode when handling the 'go' command. Full details on register reads and writes are in Section 5.6.6, "Register Write".

5.6.1 Submitting Instructions to the Core

The Probe Mode Instruction Register (PIR) is used via two TAP instructions:

- The WRITEPIR TAP data register allows the debug tool to write instruction opcodes into the TAP. Once the WRITEPIR data register is populated, the opcodes are submitted to the core for execution using the SUBMITPIR TAP instruction.
- SUBMITPIR is an 'IR Only' TAP command; there is no corresponding data register. The following instructions are supported by the Core while in Probe Mode:
- Memory Read and Write via MOV r, m8/m16/m32 and MOV m8/m16/m32, r
- I/O Read and Write via IN and OUT
- MSR Read and Write via RDMSR and WRMSR
- The WBINVD and INVVD instructions
- The CPUID instruction

Note: The PG bit in the CR0 register must be 0 before submitting instructions to the core for execution.

5.6.1.1 Instruction Faults

Instructions submitted via the TAP that are executed by the core may generate faults. The debug software should make every attempt to verify that instructions used by the software internally do not fault. However, the user is free to submit any instruction or data that may result in a fault. Some potential faults: page fault, segment violation, or accessing an undefined MSR.

If a fault occurs:

- The core will not re-enter Probe Mode and Run Control Hardware will not see a pulse on the PRDY# signal.
- The core may enter the SHUTDOWN state, depending on the side effects of the fault. For example, if a valid IDT is not set up or the fault handlers are not present in memory, the core may enter SHUTDOWN.

Debug software can watch for TAPSTATUS[1] ("Probe Mode Request Pending") timeouts after submitting instructions to the core. If a timeout occurs, debug software may attempt to force the core back into Probe Mode via the PROBEMODE TAP instruction. If the core has entered SHUTDOWN, there may be no other option other than resetting the core to restart instruction execution.

5.6.2 EIP Management

On probe mode entry from SW break, ITP/Debugger must decrement EIP by 1 before replacing the 0xF1 with the real opcode byte.

5.6.3 DR7 Management

While the core is in Probe Mode, debug software must write the value of 0x00000000 to the DR7 register. This is needed to disable any data or I/O breakpoints while the core is halted. This is the reset value for DR7 for this architecture.

5.6.3.1 EIP and Software Breakpoints

Software breakpoints behave as instruction traps in the Quark™ SE Core. After entering Probe Mode due to a software break point trigger, the EIP register points to the instruction immediately after the 0xF1 opcode used for the breakpoint.

Debug software must check for software breakpoint matches using EIP – 1 when determining the break cause and replacing the displaced opcode byte when resuming execution from the break address.

5.6.4 WRITEPIR Register Format

The 64-bit WRITEPIR TAP data register allows ITP to submit instruction opcodes for execution by the core. If the instruction submitted by Debug software does not use all 64 bits, the remaining bytes must be 'padded' with NOP instructions, opcode 0x90.

The data value must be transformed using a two-step process prior to the addition of the NOP pads.

The following sample using the instruction MOV EAX, DWORD PTR [EDX] illustrates this process.

The opcodes for this instruction used in this sample are 0x66678B02.

- Add enough NOP opcodes (0x90) to create the full 64 bit value:
  - 0x66678B02 becomes 0x66678B0290909090
  - Reverse the 64 bit value:
  - 0x66678B0290909090 becomes 0x0909090940D1E666

5.6.5 Register Read

The template to read a register is as follows:

1. Find the PIR value for the desired register in Table 13.
   a. Use WRITEPIR to shift in the 64-bit value from the table.
   b. Shift in SUBMITPIR TAP instruction.
2. Write the 'SRAMACCESS' pseudo opcode (from Table 13) to the PIR.
3. Shift in SUBMITPIR TAP instruction.
4. Write the 'SRAM2PDR' pseudo opcode (from Table 13) to the PIR.
5. Shift in SUBMITPIR TAP instruction.
6. Use the PDR_ACCESS TAP instruction to read the 32 bits of data.

5.6.6 Register Write

Note: This process has changed from the version of the Lakemont core used in prior SOCs. This new algorithm is backwards compatible with older cores and may be used as the sole register write algorithm by debug tools.

1. Find the PIR value for the desired register in Table 13
   a. Use WRITEPIR to shift in the 64-bit value from the table.
   b. Shift in SUBMITPIR TAP instruction.
2. Shift in the 'SRAMACCESS' pseudo opcode (from Table 13) to the PIR data register.
3. Shift in SUBMITPIR TAP instruction.
4. Shift the data to be written in to the TAP PDR using the RWPDR TAP instruction.
5. Shift in the 'PDR2SRAM' pseudo opcode (from Table 13) to the PIR data register.

6. Shift in SUBMITPIR TAP instruction.

5.6.7 Special Cases for Register Access

5.6.7.1 PMCR

This internal register is write-only; reads return all zeroes. Its bits can be read only through TAPSTATUS (bits 3, 5, and 7).

5.6.8 Checking for HALT State

When the core executes the HLT instruction, it stops instruction execution and places the processor in a HALT state. Any of the following will resume execution:

- Enabled interrupt (including NMI and SMI)
- Debug exception
- BINIT# signal
- INIT# signal
- RESET# signal

If an interrupt (including NMI) is used to resume execution after a HLT instruction, the saved instruction pointer (CS:EIP) points to the instruction following the HLT instruction.

The core also emits a special bus cycle alerting other agents on the bus that the core has entered HALT. It also automatically shuts-off the core functional clocks to most units following entry into HALT state. A probe mode request when the processor is in HALT state will cause the processor to restart its functional clocks and service the probe mode entry request.

The debugger can determine if the core is in HALT by reading the HALT register and checking bit 16. If bit 16 is 1, the core is in the HALT state.

5.6.9 Pseudo Opcodes for Architectural Register Access

Each register in Table 11 is described in detail in the Intel Software Developer's Manuals here: http://www.intel.com/content/www/us/en/processors/architectures-software- developer-manuals.html

Table 10. Register Access PIR Values

Register	64 Bit PIR Value	Register	64 Bit PIR Value	Register	64 Bit PIR Value
CR0	0x000000001D660000	TSSlimit	0x000000181D660000	CSlimit	0x0000000C1D660000
CR3	0x000000801D660000	IDTar	0x000000981D660000	ESar	0x0000008C1D660000

Register	64 Bit PIR Value	Register	64 Bit PIR Value	Register	64 Bit PIR Value
EFLAGS	0x000000401D660000	IDTbase	0x000000581D660000	ESbase	0x0000004C1D660000
EIP	0x000000C01D660000	IDTlimit	0x000000D81D660000	ESlimit	0x000000CC1D660000
EDI	0x000000201D660000	GDTar	0x000000381D660000	CR4	0x0000002C1D660000
ESI	0x000000A01D660000	GDTbase	0x000000B81D660000	SIP	0x000000AC1D660000
EBP	0x000000601D660000	GDTlimit	0x000000781D660000	TMPD	0x0000006C1D660000
ESP	0x000000E01D660000	LDTar	0x000000F81D660000	TMPB	0x000000EC1D660000
EBX	0x000000101D660000	LDTbase	0x000000041D660000	TMPC	0x0000001C1D660000
EDX	0x000000901D660000	LDTlimit	0x000000841D660000	HALT	0x0000009C1D660000
ECX	0x000000501D660000	GSar	0x000000441D660000	REV	0x0000005C1D660000
EAX	0x000000D01D660000	GSbase	0x000000C41D660000	BASE	0x000000DC1D660000
DR6	0x000000301D660000	GSlimit	0x000000241D660000	PDR6	0x0000003C1D660000
DR7	0x000000B01D660000	FSar	0x000000A41D660000	CR2	0x000000BC1D660000
TR	0x000000701D660000	FSbase	0x000000641D660000	DR0	0x0000007C1D660000
LDTR	0x000000F01D660000	FSlimit	0x000000E41D660000	DR1	0x000000FC1D660000
GS	0x000000081D660000	DSar	0x000000141D660000	DR2	0x000000021D660000
FS	0x000000881D660000	DSbase	0x000000941D660000	DR3	0x000000821D660000
DS	0x000000481D660000	DSlimit	0x000000541D660000	PMCR	0x000000421D660000
SS	0x000000C81D660000	SSar	0x000000D41D660000	SRAMACCES	0x0000000E9D660000
CS	0x000000281D660000	SSbase	0x000000341D660000	SRAM2PDR	0x4CF0000000000000
ES	0x000000A81D660000	SSlimit	0x000000B41D660000	PDR2SRAM	0x0CF0000000000000
TSSar	0x000000681D660000	CSar	0x000000741D660000	-	-
TSSbase	0x000000E81D660000	CSbase	0x000000F41D660000	-	-

5.6.10 Probe Mode Control Register

This register is a 32-bit internal register used by probe mode logic to enable processor redirection to probe mode during various events. It currently allows code, data & IO breakpoint, single-stepping, software breakpoints, machine-check error, and shutdowns to be redirected to Probe Mode.

This register is accessible via the TAP. The pseudo opcode PIR value for this register is 0x000000421D660000.

Table 11. PMCR Description

Bit Position	Description
31:3	Reserved. Write to these bits are ignored.

Bit Position	Description
2	Machine Check Redirection: When this bit is set to 1 MACHINE CHECK events cause an entry to probe mode. This feature is disabled for this SoC.
1	Shutdown Redirection Bit, when set, it redirects the CPU into probe mode as a result of a processor shutdown.
0	Probe Mode Redirection Bit. When set, all Code, Data & I/O Breakpoints, Software Breakpoints & Single Step redirect to probe mode.

5.6.11 Accessing Model Specific Registers (MSR)

MSRs may be accessed (read and written) using the following flow:

1. Write the MSR index value to the ECX register using normal internal register write operation.
2. Read:
    a. Submit an ‘RDMSR’ instruction to the core via WRITEPIR and SUBMITPIR.
    b. Bits 31:0 of the MSR data will be in EAX; bits 63:32 of the MSR data will be EDX.
3. Write:
    a. Move bits 31:0 of the data to be written to EAX; move bits 63:32 of the data to EDX.
    b. Submit a ‘WRMSR’ instruction to the core via WRITEPIR and SUBMITPIR.

5.7 Reading and Writing Memory

Memory may be read and written using the Quark™ SE Core by direct injection of the macro-instructions via the TAP’s PIR register. The core will:

1. Restore all internal register states.
2. Exit Probe Mode.
3. Execute the single instruction.
4. Re-enter Probe Mode.
5. When the re-entry to Probe Mode completes, TAPSTATUS[2] bit is set indicating to debug software that the core is ready for the next instruction.

If the memory access fails, the core will not re-enter probe mode. Debug software may use this to detect failures and attempt to force the core back into Probe Mode, or let the user know that the core must be reset.

5.7.1 Management of Architectural Registers for Memory Access

All register reads and writes used as part of memory accesses are performed using the pseudo opcodes to interact with the internal registers. Only the MOV instructions used for the actual read or write are executed by the core. The MOV instructions must be written to the WRITEPIR data register using the standard NOP padding and byte reversal.

5.7.1.1 DS Selector

The DS selector must be changed before debug software can access the full 4GB address space while in Probe Mode. Prior to any memory access, the registers listed in the following table must be set to the values specified for each. Debug software must take care to read these registers after Probe Mode entry and cache the values so that they may be restored prior to Probe Mode exit.

Table 12. DS Selector Values for Memory Access

Register	Value
DS Base	0x00000000
DS Limit	0xFFFFFFFF
DS AR	0x004F9300

5.7.2 CSLIMIT Considerations

When injecting CPU macro instructions during probe mode, the value of CSLIMIT should not fall below 32’h20 as this could lead to generation of undesired exceptions. For reliable operation, ITP/Debugger should modify CSLIMIT to 32’hFFFF_FFFF upon probe mode entry and restore the original value back upon probe mode exit.

5.7.2.1 Adjust CPL Prior to Memory Access

The code-privilege level in effect prior to Probe Mode entry may prevent the use of the WBINVD instruction needed to flush the cache. Debug software must set the CPL to 0 in both the CS access byte and the SS access byte if they are non-zero as part of the architectural save-state process. This is done by changing bits 13 and 14 in the SSAR and CSAR registers to zero. If they are zero upon Probe Mode entry, Debug software does not need to change them.

5.7.2.2 Disable Interrupts Prior to Memory Access

The processor must not be allowed to handle pending interrupts when it is released from Probe Mode to handle memory reads/writes. Interrupt handling is disabled by setting the IF bit in the EFLAGS register to 0.

5.7.2.3 CR0

The value of the CR0.PG (bit 31, paging enabled) bit influences memory reads and writes. To read from a physical memory address provided by the user or during address translation, Debug software must set the CR0.PG to bit to ‘0. Memory reads and writes using linear, two-field virtual, and three-field virtual must be translated to the physical address form before being used to read or write memory.

5.7.3 Memory Read

1. Clear CR0.PG bit if is it 1.
2. Write address to EAX.
3. Submit the read instruction to the core: MOV EDX, DWORD PTR [EAX].
4. Read memory data from EDX via the SRAM and PDR.
5. Restore CR0.PG bit if it was changed in the first step.

5.7.4 Memory Write

1. Clear CR0.PG bit if it is 1.
2. Write address to EAX.
3. Write data to EDX.
4. Submit the memory write instruction to the core: MOV DWORD PTR [EAX], EDX.
5. Restore the CR0.PG bit if it was changed in the first step.

5.8 Reading and Writing I/O Ports

The Quark™ SE Core supports I/O port reads and writes using the IN and OUT instructions submitted to the core via the TAP.

Note: The CR0.PG bit must be 0 prior to using the PIR TAP instructions to submit I/O read and write instructions.

5.8.1 I/O Read Example in Probe Mode

1. Write the I/O address to read from to the DX register.
2. Submit one of the following based on the desired access width:
   - 1 byte wide read: IN AL, DX
   - 2 byte wide read: IN AX, DX
   - 4 byte wide read: IN EAX, DX

5.8.2 I/O Write Example in Probe Mode

1. Write the I/O address for the write to the DX register.
2. Write the data to be written to the EAX register.
3. Submit one of the following based on the desired access width:
   - 1 byte wide write: OUT DX, AL
   - 2 byte wide write: OUT DX, AX
   - 4 byte wide write: OUT DX, EAX

5.9 Hardware Breakpoints

1. Write the linear address to DR0-3.
2. Set DR6 and DR7 bits as needed.
3. Set bit 0 in PMCR to convert the #DB exception to a Probe Mode entry.

5.10 Software Breakpoints

To determine if an instance of the Quark™ SE Core supports software breakpoints, check bit 4 in the TAPSTATUS register. If the bit is ‘1, the core supports software breakpoints.

Configure software breakpoints in the Core by replacing one byte of the original instruction opcode(s) in memory with the value 0xF1 and setting PMCR[IR] to ‘1.

5.11 Single Step

1. Set EFLAGS[TF] to ‘1 and PMCR[IR] to ‘1.
2. Release from Probe Mode and wait for TAPSTATUS[2].

On Probe Mode entry from SW break, debug software must decrement EIP by 1 before replacing the 0xF1 with the real opcode byte.

5.12 Redirections into Probe Mode

5.12.1 Shutdown Break

The processor may be configured to enter Probe Mode when a SHUTDOWN event occurs. This is enabled by setting bit 1 in the PMCR register to ‘1.

To determine if the cause for an asynchronous entry into Probe Mode was caused by a shutdown break, bit 6 in the TAPSTATUS data register will be set to ‘1. PMCR bit 5 holds a copy of the enable bit in PMCR. This allows debug software to check if the shutdown redirection is enabled without putting the core into Probe Mode.

6.0 Flash Programming Options

6.1 Flash Overview

As described in the block diagram in Figure 1, the on-die Flash memory is split into two sections, an 8-KB OTP “ROM” section, and a 32-KB Flash section.

The OTP section of Flash operates as normal Flash (can be erased and rewritten) as long as a specific bit in the Flash image is not programmed (set to zero). That bit is the first bit in the array, that is, bit 0 of memory address 0xFFFFE000. Once the bit is programmed Erase and Write operations of the 8-KB OTP section of Flash are permanently disabled.

Note: Once programming of the OTP area is disabled, JTAG debug access to the SoC is also permanently disabled.

6.2 Flash Controller

There is one instance of Flash Controllers within the SoC used for programming of the Flash contents.

- The first Flash Controller controls programming of the 8-KB OTP “ROM” and 32-KB of the Flash section.

Detailed descriptions of the registers used within each controller are described in Chapter 7.0, “Memory Mapped Registers”.

6.3 Supported Operations

6.3.1 Mass Erase

The Flash controller supports a mass erase operation.

In the case of the first controller an option exists to erase both the OTP section if the OTP bit has not been programmed, and the 32-KB Flash section associated with the controller. There is no option to only erase the OTP section.

6.3.1.1 Operation

1. Set the MASS_ERASE bit of the CTRL register. If also erasing the OTP section, the MASS_ERASE_INFO bit should be set in the same register write operation.
2. Wait for ER_DONE (bit [0]) in the FLASH_STTS register to be set, this signals the erase operation is complete.

6.3.2 Page Erase

The controller allows for partial erase functions, known as Page Erase operations. A Page is defined as a 2-KB section of Flash. The OTP section of is capable of being page erased if the OTP bit has not been programmed.

6.3.2.1 Operation

1. Write the flash page offset address into WR_ADDR (bits [19:2]) of the FLASH_WR_CTRL or ROM_WR_CTRL register.
2. Set the ER_REQ (bit [1]) of the FLASH_WR_CTRL or ROM_WR_CTRL register.
3. Wait for ER_DONE (bit [0]) in the FLASH_STTS register to be set, this signals the page erase is complete

6.3.3 Data Program

Data is programmed into Flash a DWORD (32 bits or 4 bytes) at a time. A pair of registers is used for each operation: a write control register and a write data register. There are three pairs of these registers:

- Flash Controller 0
  - ROM write control/data – for programming locations in the OTP area
  - Flash write control/data – for programming locations in the 32-KB area

6.3.3.1 Operation

Before writing the flash the intended region must be erased. This operation brings the contents of the each byte location to 0xFF. When programming an area of flash less than a full page care should be taken. If the partial area being programmed is all erased, then a partial update is OK, otherwise a page read/erase/write sequence is required in which the required changes in data are made to the temporary copy held between the read and write operations.

The following steps program a single DWORD; repeat the steps to program multiple locations. Programming operations are always 32-bit aligned.

1. Write the value to be programmed into the FLASH_WR_DATA or ROM_WR_DATA register.
2. Write the DWORD offset address inside the region into WR_ADDR (bits [19:2]) of the FLASH_WR_CTRL or ROM_WR_CTRL register.
3. Start the programming operation, by setting the WR_REQ (bit [0]) of the FLASH_WR_CTRL or ROM_WR_CTRL register.
4. Wait for WR_DONE (bit [1]) in the FLASH_STTS register to be set, this signals the DWORD write is complete.

§

7.0 Memory Mapped Registers

Detailed descriptions of all memory mapped registers within the SoC accessed by the debugger are described in the next section.

7.1 Register Field Access Types

Table 13. Register Field Access Types

Access Type	Meaning	Description
RO	Read Only	In some cases, if a register is read only, writes to this register location have no effect. In other cases, two separate registers are located at the same location where a read accesses one of the registers and a write accesses the other. See the I/O and memory map tables for details.
WO	Write Only	In some cases, if a register is write only, reads to this register location have no effect. In other cases, two separate registers are located at the same location where a read accesses one of the registers and a write accesses the other. See the I/O and memory map tables for details.
R/W	Read/Write	A register with this attribute can be read and written.
R/WC	Read/Write Clear	A register bit with this attribute can be read and written. However, a write of 1 clears (sets to 0) the corresponding bit and a write of 0 has no effect.
R/WO	Read/Write-Once	A register bit with this attribute can be written only once after power up. After the first write, the bit becomes read only.
R/WLO	Read/Write, Lock-Once	A register bit with this attribute can be written to the non-locked value multiple times, but to the locked value only once. After the locked value has been written, the bit becomes read only.
RW/V	Read/Write, hardware clear	A register bit with this attribute can be read and written. This is a trigger bit - a write of 1 sets (sets to 1) the corresponding bit which is then cleared by hardware. A write of 0 has no effect.
RO/V	Read Only, hardware modify	A register bit with this attribute is read only. A write has no effect. Hardware can modify this bit type.
RO/C/V	Read Only, Read Clear, hardware modify	A register bit with this attribute is read only. A read clears this bit (sets to 0). Hardware can set this bit type.
RW/1S	Read/write 1 to set	A register bit with this attribute is read/write. A write of 1 sets this bit (sets to 1) after which it can only be cleared by a soft reset.
RW/L	Read/write lock	A register bit with this attribute is read/write. It can also be locked so further writes are blocked. Typically the lock is enabled by a RW/1S register bit.

Access Type	Meaning	Description
RW/1C/V	Read/Write	A register with this attribute is read/write 1 to clear (set to 0) and can be loaded by hardware.
Reserved	Reserved	The value of reserved bits must never be changed.
Default	Default	When the processor is reset, it sets its registers to predetermined default states. The default state represents the minimum functionality feature set required to successfully bring up the system. Hence, it does not represent the optimal system configuration. It is the responsibility of the system initialization software to determine configuration, operating parameters, and optional system features that are applicable, and to program the processor registers accordingly.

7.2 Flash Controller 0

7.2.1 Register Summary

Table 14. Flash Controller 0 Registers

MEM Address	Default	Name
0xB0100000	0000_0060h	TMG_CTRL
0xB0100004	0000_0000h	ROM_WR_CTRL
0xB0100008	0000_0000h	ROM_WR_DATA
0xB010000C	0000_0000h	FLASH_WR_CTRL
0xB0100010	0000_0000h	FLASH_WR_DATA
0xB0100014	0000_0000h	FLASH_STTS
0xB0100018	0000_0000h	CTRL

7.2.2 TMG_CTRL (TMG_CTRL)

Table 15. Flash Timing Control Register

Bits	Access Type	Default	Description
31:15	RO	17'h0	**RSV (RSV)** Reserved
14	RW	1'h0	**CLK_SLOW (CLK_SLOW)** Slow clock - when 1, zero wait state flash access is possible. When 0, flash accesses will always have one or more wait states. This bit must be set to zero when clock frequencies are above 6.7 MHz.

Bits	Access Type	Default	Description
13:10	RW	4'h0	**READ_WAIT_STATE_H (READ_WAIT_STATE_H)** Flash SE high pulse width in system clocks plus one. Set to 0 for clock frequencies below 66 Mhz.
9:6	RW	4'h1	**READ_WAIT_STATE_L (READ_WAIT_STATE_L)** Flash SE low pulse width in system clocks plus one. This must be set to one when the system clock frequency is above 20 MHz. This determines when the Flash controller generates a read data valid indication and is based on the Flash data access time.
5:0	RW	6'h20	**MICRO_SEC_CNT (MICRO_SEC_CNT)** Number of clocks in a micro second.

7.2.3 ROM_WR_CTRL (ROM_WR_CTRL)

Table 16. ROM Write Control Register

Bits	Access Type	Default	Description
31:20	RO	12'h0	**RSV (RSV)** Reserved

Bits	Access Type	Default	Description
19:2	RW	18'b0	**WR_ADDR (WR_ADDR)** Write Address. WR_ADDR is a byte address. The write granularity to flash is a dword so for writes the 2 LSBs are unused. The following table shows the WR_ADDR to system address mapping:

WR_ADDR[17:0]	System Address
18'h0	0xFFFF_E000
18'h4	0xFFFF_E004
…	…
18'h1FFC	0xFFFF_FFFC

The page erase granularity to Flash is 2 kbytes. The following table shows the WR_ADDR address setting required to erase each of the ROM pages The ROM pages can be erased and re-programmed as long as the OTP bit has not been programmed.

WR_ADDR[17:0]	System Address
18'h0	0xFFFF_E000 - 0xFFFF_E7FF
18'h800	0xFFFF_E800 - 0xFFFF_EFFF
18'h1000	0xFFFF_F000 - 0xFFFF_F7FF
18'h1800	0xFFFF_F800 - 0xFFFF_FFFF

Bits	Access Type	Default	Description
1	RW/V	1'b0	**ER_REQ (ER_REQ)** Erase request - set to '1' to trigger a ROM Page Erase. Check the FLASH_STTS.ER_DONE bit to determine when the erase completes. ER_REQ is self-clearing. ER_REQ has no effect after CTRL.FL_WR_DIS has been written to 1'b1. Hardware blocks all erases after ROM has been programmed.

Bits	Access Type	Default	Description
0	RW/V	1'b0	**WR_REQ (WR_REQ)** Write request - set WR_REQ to '1' to trigger a ROM write. Check the FLASH_STTS.WR_DONE bit to determine when the write completes. WR_REQ is self-clearing. WR_REQ has no effect after CTRL.FL_WR_DIS has been written to 1'b1. ***Hardware blocks all ROM writes the OTP bit in ROM is programmed. The OTP bit in ROM is programmed by writing '0' to bit 0 of address 0 in the ROM. Once this is done all further writes to ROM are blocked by hardware. The OTP bit should not be programmed in a debug environment.***

7.2.4 ROM_WR_DATA (ROM_WR_DATA)

Table 17. ROM Write Data Register

Bits	Access Type	Default	Description
31:0	RW	32'h0	**DATA (DATA)** ROM Write Data

7.2.5 FLASH_WR_CTRL (FLASH_WR_CTRL)

Table 18. DCCM Base Address Register

Bits	Access Type	Default	Description
31:20	RO	12'h0	**RSV (RSV)** Reserved

Bits	Access Type	Default	Description
19:2	RW	18'b0	**WR_ADDR (WR_ADDR)** Write Address. WR_ADDR is a byte address. The write granularity to flash is a dword so for writes the 2 LSBs are unused. The following table shows the WR_ADDR to system address mapping:

WR_ADDR[17:0]	System Address
18'h0	0x4000_0000
18'h4	0x4000_0004
...	...
18'h2_FFFC	0x4002_FFFC

The page erase granularity to Flash is 2 kbytes. The following table shows the WR_ADDR address setting required to erase each of the Flash pages.

WR_ADDR[17:0]	System Address
18'h0	0x4000_0000 - 0x4000_07FF
18'h800	0x4000_0800 - 0x4000_0FFF
...	...
18'h2_F800	0x4002_F800 - 0x4002_FFFF

Bits	Access Type	Default	Description
1	RW/V	1'b0	**ER_REQ (ER_REQ)** Erase request - set to '1' to trigger a Flash Page Erase. The page number is specified by WR_ADDR[17:11]. ER_REQ is self-clearing. ER_REQ has no effect after CTRL.FL_WR_DIS has been written to '1'.
0	RW/V	1'b0	**WR_REQ (WR_REQ)** Write request - set WR_REQ to '1' to trigger a Flash write. Check the FLASH_STTS.WR_DONE bit to determine when the write completes. WR_REQ is self-clearing. WR_REQ has no effect after CTRL.FL_WR_DIS has been written to '1'.

7.2.6 FLASH_WR_DATA (FLASH_WR_DATA)

Table 19. Flash Write Data Register

Bits	Access Type	Default	Description
31:0	RW	32'h0	**DATA (DATA)** Flash Write Data

7.2.7 FLASH_STTS (FLASH_STTS)

Table 20. Flash Status Register

Bits	Access Type	Default	Description
31:3	RO	29'h0	**RSV (RSV)** Reserved
2	RO/V	1'h0	**ROM_PROG (ROM_PROG)** ROM programmed - when set this indicates that ROM has been programmed and any further attempt to write ROM is blocked.
1	RO/C/V	1'b0	**WR_DONE (WR_DONE)** Write done - when set this indicates that a write operation has completed. WR_DONE is cleared on read.
0	RO/C/V	1'h0	**ER_DONE (ER_DONE)** Erase done - when set this indicates that an erase has completed. ER_DONE is cleared on read.

7.2.8 CTRL (CTRL)

Table 21. Control Register

Bits	Access Type	Default	Description
31:9	RO	23'h0	**RSV (RSV)** Reserved
8	RW/V	1'h0	**ERC (ERC)** Erase reference
7	RW/V	1'h0	**MASS_ERASE (MASS_ERASE)** Mass Erase - set to '1' to trigger an erase of Flash. This has no effect after FL_WR_DIS has been written to '1'.
6	RW	1'h0	**MASS_ERASE_INFO (MASS_ERASE_INFO)** Mass Erase Info - Valid when MASS_ERAES is '1'. When MASS_ERASE_INFO = '1' then the ROM portion of Flash is erased during a Mass Erase. When MASS_ERASE_INFO = '0' then the ROM portion of Flash is not erased during a Mass Erase. If the ROM portion of Flash is detected as programmed by the Flash Controller then hardware will block a ROM erase. This has no effect after FL_WR_DIS has been written to '1'.
5	RW	1'h0	**LVE_MODE (LVE_MODE)** Low Voltage mode
4	RW/1S	1'h0	**FL_WR_DIS (FL_WR_DIS)** Flash Write Disable

Bits	Access Type	Default	Description
3	RW/1S	1'h0	**ROM_RD_DIS_U (ROM_RD_DIS_U)** Rom read disable for upper 4k region of ROM
2	RW/1S	1'h0	**ROM_RD_DIS_L (ROM_RD_DIS_L)** Rom read disable for lower 4k region of ROM
1	RW	1'h0	**PRE_FLUSH (PRE_FLUSH)** Prefetch buffer Flush
0	RW	1'h0	**PRE_EN (PRE_EN)** Prefetch Enable. When '1' cache line prefetching is enabled. When '0' cacheline prefetching is disabled.

7.3 System Control/Status

7.3.1 Register Summary

Table 22. System Control/Status Registers

MEM Address	Default	Instance Name	Name
0xB0800008	0000_0302h	OSC0_CFG1	Hybrid Oscillator Configuration 1
0xB0800024	0000_0007h	CCU_EXT_CLOCK_CTL	External Clock Control
0xB0800038	0000_0087h	CCU_SYS_CLK_CTL	System Clock Control
0xB0800570	0000_0000h	RSTC	Reset Control
0xB0800574	0000_0000h	RSTS	Reset Status

7.3.2 Hybrid Oscillator Configuration 1 (OSC0_CFG1)

Table 23. Hybrid Oscillator Configuration Register

Bits	Access Type	Default	Description
31:30	RO	2'h0	**RSVD (RSVD)** Reserved
29:20	RW/P/L	10'h0	**Trim Code (OSC0_FTRIMOTP)** 10-bit trim code

Bits	Access Type	Default	Description
19:16	RW/P/L	4'h0	**Crystal oscillator trim (OSC0_FADJ_XTAL)** Load cap corresponding to each trim code. 10 pF is the default load cap and the corresponding default trim code is 0000. 0111b: 5.55 pF 0110b: 6.18 pF 0101b: 6.82 pF 0100b: 7.45 pF 0011b: 8.08 pF 0010b: 8.71 pF 0001b: 9.34 pF 0000b: 10 pF 1111b: 10.61 pF 1110b: 11.24 pF 1101b: 11.88 pF 1100b: 12.51 pF 1011b: 13.14 pF 1010b: 13.77 pF 1001b: 14.4 pF 1000b: 15.03 pF
15	RO	1'h0	**RSVD (RSVD)** Reserved
14:13	RW/P/L	2'h0	**Hybrid Oscillator Temperature Control (OSC0_TEMPCOMPPRG)** Bits to control the temperature compensation of silicon oscillator 00b: Default temperature compensation 01b: Default temperature compensation 10b: Reduce PTAT current in temperature compensation 11b: Increase PTAT current in temperature compensation
12:10	RW/P/L	3'h0	**Hybrid Oscillator Bias Current Control (OSC0_IBIASPRG)** Bits to control the bias current of the silicon oscillator
9:8	RW/P	2'h3	**Silicon Oscillator Frequency Selection (OSC0_SI_FREQ_SEL)** 00b: 32 MHz 01b: 16 MHz 10b: 8 MHz 11b: 4 MHz
7:5	RW/P/L	3'h0	**Silicon Oscillator Start-Up (OSC0_START_UP)** Bits to control the start-up of silicon oscillator core

Bits	Access Type	Default	Description
4	RW/P/L	1'h0	**Crystal Oscillator Bypass Mode Enable (OSC0_BYP_XTAL)** Enable bypass mode for crystal oscillator 0b: Disable 1b: Enable
3	RW/P	1'h0	**Hybrid Oscillator Mode Select (OSC0_MODE_SEL)** Selects between crystal and silicon oscillator output 0b: Silicon oscillator output 1b: Crystal oscillator output
2	RW/V/L	1'h0	**Hybrid Oscillator Power-Down Control (OSC0_PD)** 0b: Hybrid Oscillator in active mode 1b: Hybrid Oscillator in power down mode
1	RW/P	1'h1	**Silicon Oscillator Enable (OSC0_EN_SI_OSC)** Enables the Silicon Oscillator 0b: Silicon Oscillator Disabled 1b: Silicon Oscillator Enabled
0	RW/P	1'h0	**Crystal Oscillator Enable (OSC0_EN_CRYSTAL)** Enables the Crystal Oscillator 0b: Crystal Oscillator Disabled 1b: Crystal Oscillator Enabled

7.3.3 External Clock Control (CCU_EXT_CLOCK_CTL)

Table 24. External Clock Control Register

Bits	Access Type	Default	Description
31:5	RO	27'h0000000	**RSVD (RSVD)** Reserved
4:3	RW/P	2'h0	**External clock divider (CCU_EXT_CLK_DIV)** 00b: divide by 1 01b: divide by 2 10b: divide by 4 11b: divide by 8
2	RW/P	1'b1	**External clock divider enable (CCU_EXT_CLK_DIV_EN)** This bit must be written from 0 -> 1 to apply the value
1	RW/P	1'b1	**External Clock Enable (CCU_EXT_CLK_EN)** 1b: enable 0b: disable
0	RW/P	1'b1	**External RTC enable (CCU_EXT_RTC_EN)** 1b: enable 0b: disable

7.3.4 System Clock Control (CCU_SYS_CLK_CTL)

Table 25. System Clock Control Register

Bits	Access Type	Default	Description
31:10	RO	22'h000000	**RSVD (RSVD)** Reserved
9:8	RW/P	2'h0	**System Clock Divider (CCU_SYS_CLK_DIV)** 00b: divide by 1 01b: divide by 2 10b: divide by 4 11b: divide by 8
7	RW/P	1'b1	**System Clock Divider Enable (CCU_SYS_CLK_DIV_EN)** This bit must be written from 0 -> 1 to apply the value
6:3	RW/P/L	4'h0	**RTC Clock Divider (CCU_RTC_CLK_DIV)** 0000b: divide by 1 0001b: divide by 2 0010b: divide by 4 0011b: divide by 8 0100b: divide by 16 0101b: divide by 32 0110b: divide by 64 0111b: divide by 128 1000b: divide by 256 1001b: divide by 512 1010b: divide by 1024 1011b: divide by 2048 1100b: divide by 4096 1101b: divide by 8192 1110b: divide by 16384 1111b: divide by 32768
2	RW/P/L	1'b1	**RTC Clock Divider Enable (CCU_RTC_CLK_DIV_EN)** This bit must be written from 0 -> 1 to apply the value
1	RW/P/L	1'b1	**RTC Clock Enable (CCU_RTC_CLK_EN)** 1b: enable 0b: disable
0	RW/P	1'b1	**Select Clock (CCU_SYS_CLK_SEL)** 0b: 32 kHz RTC Crystal Oscillator 1b: 32 MHz Hybrid Oscillator

7.3.5 Reset Control (RSTC)

Note: A write to this register with RSTC.COLD or RSTC.WARM set initiates a reset. Software must only write to one of these bits at a time, else the behavior is undefined.

These bits automatically clear once the reset occurs, so there is no need for software to clear them.

Table 26. Reset Control Registers

Bits	Access Type	Default	Description
31:4	RO	28'h0000000	**RSVD (RSVD)** Reserved
3	RW/P	1'h0	**Cold Reset (COLD)** When this bit is set, the SoC performs a cold reset.

Bits	Access Type	Default	Description
2	RO	1'h0	**RSVD (RSVD)** Reserved
1	RW	1'h0	**Warm Reset (WARM)** When this bit is set, the SoC performs a warm reset.
0	RO	1'h0	**RSVD (RSVD)** Reserved

7.3.6 Reset Status (RSTS)

Table 27. Reset Status Register

Bits	Access Type	Default	Description
31:5	RO	27'h0000000	**RSVD (RSVD)** Reserved
4	RW/1C/V/P	1'h0	**Sensor Subsystem Halt Interrupt Triggered Warm Reset (SS_HALT_WRST)** When this bit is set, it indicates that an enabled Sensor Subsystem Halt interrupt triggered a warm reset.
3	RW/1C/V/P	1'h0	**Host Processor Halt Interrupt Triggered Warm Reset (HOST_HALT_WRST)** When this bit is set, it indicates that an enabled Host Halt interrupt triggered a warm reset.
2	RW/1C/V/P	1'h0	**Sensor Subsystem Watchdog Timer Triggered Warm Reset (SS_WDG_WRST)** When this bit is set, it indicates that Watchdog Timer in the Sensor Subsystem triggered a warm reset.
1	RW/1C/V/P	1'h0	**Watchdog Timer Triggered Warm Reset (WDG_WRST)** When this bit is set, it indicates that Watchdog Timer in the Peripheral block triggered a warm reset.
0	RW/1C/V/P	1'h0	**Software Initiated Warm Reset (SW_WRST)** When this bit is set, it indicates that warm reset was initiated by software writing to RSTC.WARM.

§