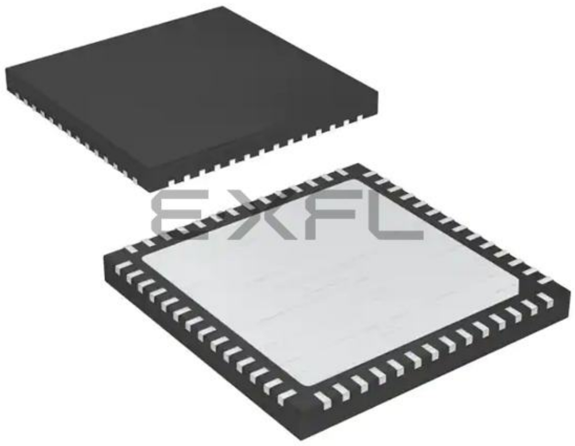
Analog Devices Inc./Maxim Integrated - MAXQ2000-RBX+ Datasheet



What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	-
Core Size	16-Bit
Speed	20MHz
Connectivity	1-Wire®, SPI, UART/USART
Peripherals	LCD, POR, PWM, WDT
Number of I/O	50
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 2.75V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	56-WFQFN Exposed Pad
Supplier Device Package	56-TQFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/maxq2000-rbx

TABLE OF CONTENTS

SECTION 1: Overview . . . 1-1
SECTION 2: Architecture . . . 2-1
SECTION 3: Programming . . . 3-1
SECTION 4: System Register Descriptions . . . 4-1
SECTION 5: Peripheral Register Modules . . . 5-1
SECTION 6: General-Purpose I/O Module . . . 6-1
SECTION 7: Timer/Counter Timer . . . 7-1
SECTION 8: Timer/Counter 1 Module . . . 8-1
SECTION 9: Timer/Counter 2 Module . . . 9-1
SECTION 10: Serial I/O Module . . . 10-1
SECTION 11: Serial Peripheral Interface (SPI Module) . . . 11-1
SECTION 12: Hardware Multiplier Mode . . . 12-1
SECTION 13: 1-Wire Bus Master . . . 13-1
SECTION 14: Real-Time Clock Module . . . 14-1
SECTION 15: Test Access Port (TAP) . . . 15-1
SECTION 16: In-Circuit Debug Mode . . . 16-1
SECTION 17: In-System Programming (JTAG) . . . 17-1
SECTION 18: MAXQ Family Instruction Set Summary . . . 18-1
REVISION HISTORY . . . Rev-1

- Real-Time Clock
- 1-Wire® Bus Master
- General-Purpose Digital I/O Ports

1.4 MAXQ10 and MAXQ20 Microcontrollers

This user's guide covers both the 8-bit MAXQ10 and 16-bit MAXQ20 microcontrollers. The primary difference between the MAXQ10 and MAXQ20 implementations is the width of the internal data bus and ALU. The MAXQ10 design implements an 8-bit internal data bus and ALU, while the MAXQ20 design implements a 16-bit internal data bus and ALU. This difference is most evident when comparing the instruction set, and more specifically, those operations that involve the ALU and accumulators. The registers on the MAXQ10 and MAXQ20 can be either 8 bits or 16 bits wide.

1-Wire is a registered trademark of Dallas Semiconductor Corp.

8-bit destination ← high byte (16-bit source)

If, however, we needed to load an 8-bit register with the high byte of a 16-bit source, it would be best to use the GR register. Transferring the 16-bit source to the GR register adds a single cycle.

```
          move   GR, LC[0]          ; move LC[0] to the GR register
          move   IC, GRH            ; copy the high byte into the IC register
```

16-bit destination ← concatenation (8-bit source, 8-bit source)

Two 8-bit source registers can be concatenated and stored into a 16-bit destination by using the prefix register to hold the high-order byte for the concatenated transfer. An additional cycle may be required if either source byte register index is greater than 0Fh or the 16-bit destination is greater than 07h.

```
          move   PFX[0], IC         ; load high order source byte IC into PFX
          move   @++SP, AP          ; store @DP[0] the concatenation of IC:AP

                                    ; 16-bit destination sub-index: dst=08h
                                    ;  8-bit source sub-indexes:
                                    ;  high=10h, low=11h
          move   PFX[1], #00h       ;
          move   PFX[3], high       ; PFX=00:high
          move   dst, low           ; dst=high:low
```

Low (16-bit destination) ← 8-bit source

To modify only the low byte of a given 16-bit destination, the 16-bit register should be moved into the GR register such that the high byte can be singulated and the low byte written exclusively. An additional cycle is required if the destination index is greater than 0Fh.

```
          move   GR, DP[0]          ; move DP[0] to the GR register
          move   PFX[0], GRH        ; get the high byte of DP[0] via GRH
          move   DP[0], #20h        ; store the new DP[0] value
                                    ; 16-bit destination sub-index: dst=10h
                                    ;  8-bit source sub-index: src=11h
          move   PFX[1], #00h       ;
          move   GR, dst            ; read dst word to the GR register
          move   PFX[5], GRH        ; get the high byte of dst via GRH
          move   dst, src           ; store the new dst value
```

High (16-bit destination) ← 8-bit source

To modify only the high byte of a given 16-bit destination, the 16-bit register should be moved into the GR register such that the low byte can be singulated and the high byte can be written exclusively. Additional cycles are required if the destination index is greater than 0Fh or if the source index is greater than 0Fh.

```
          move   GR, DP[0]          ; move DP[0] to the GR register
          move   PFX[0], #20h       ; get the high byte of DP[0] via GRH
          move   DP[0], GRL         ; store the new DP[0] value
                                    ; 16-bit destination sub-index: dst=10h
                                    ;  8-bit source sub-index: src=11h
          move   PFX[1], #00h       ;
          move   GR, dst            ; read dst word to the GR register
          move   PFX[1], #00h
          move   PFX[4], src        ; get the new src byte
          move   dst, GRL           ; store the new dst value
```

If the high byte needs to be cleared to 00h, the operation can be shortened by transferring only the GRL byte to the 16-bit destination (example follows):

```
          move   GR, DP[0]          ; move DP[0] to the GR register
          move   DP[0], GRL         ; store the new DP[0] value, 00h used for high byte
```

3.4 Reading and Writing Register Bits

The MOVE instruction can also be used to directly set or clear any one of the lowest 8 bits of a peripheral register in module 0h-5h or a system register in module 8h. The set or clear operation will not affect the upper byte of a 16-bit register that is the target of the set or clear operation. If a set or clear instruction is used on a destination register that does not support this type of operation, the register high byte will be written with the prefix data and the low byte will be written with the bit mask (i.e. all 0's with a single 1 for the set bit operation or all ones with a single 0 for the clear bit operation).

Register bits can be set or cleared individually using the MOVE instruction as follows.

```
move  IGE, #1            ; set IGE (Interrupt Global Enable) bit
move  APC.6, #0          ; clear IDS bit (APC.6)
```

As with other instructions, prefixing is required to select destination registers beyond index 07h.

The MOVE instruction may also be used to transfer any one of the lowest 8 bits from a register source or any bit of the active accumulator (Acc) to the Carry flag. There is no restriction on the source register module for the 'MOVE C, src.bit' instruction.

```
move  C, IIR.3           ; copy IIR.3 to Carry
move  C, Acc.7           ; copy Acc.7 to Carry
```

Prefixing is required to select source registers beyond index 15h.

3.5 Using the Arithmetic and Logic Unit

The MAXQ provides either an 8-bit (MAXQ10) or 16-bit (MAXQ20) ALU, which allows operations to be performed between the active accumulator and any other register. The default ALU configuration provides eight accumulator registers that are also either 8-bit (MAXQ10) or 16-bit (MAXQ20) wide, of which any one may be selected as the active accumulator. Many MAXQ devices will be equipped with 16 working accumulators.

3.5.1 Selecting the Active Accumulator

Any of the 16 accumulator registers A[0] through A[15] may be selected as the active accumulator by setting the low four bits of the Accumulator Pointer Register (AP) to the index of the accumulator register you want to select.

```
move  AP, #01h           ; select A[1] as the active accumulator
move  AP, #0Fh           ; select A[15] as the active accumulator
```

The current active accumulator can be accessed as the Acc register, which is also the register used as the implicit destination for all arithmetic and logical operations.

```
move  A[0], #55h         ; set A[0] =   55 hex (MAXQ10)
                         ;          = 0055 hex (MAXQ20)
move  AP, #00h           ; select A[0] as active accumulator
move  Acc, #55h          ; set A[0] =   55 hex (MAXQ10)
                         ;          = 0055 hex (MAXQ20)
```

3.5.2 Enabling Auto-Increment and Auto-Decrement

The accumulator pointer AP can be set to automatically increment or decrement after each arithmetic or logical operation. This is useful for operations involving a number of accumulator registers, such as adding or subtracting two multibyte integers.

If auto-increment/decrement is enabled, the AP register increments or decrements after any of the following operations:

- ADD src (Add source to active accumulator)
- ADDC src (Add source to active accumulator with carry)
- SUB src (Subtract source from active accumulator)
- SUBB src (Subtract source from active accumulator with borrow)
- AND src (Logical AND active accumulator with source)
- OR src (Logical OR active accumulator with source)
- XOR src (Logical XOR active accumulator with source)
- CPL (Bit-wise complement active accumulator)
- NEG (Negate active accumulator)

4.19 Data Pointer Control Register (DPC, Eh[4h])

Initialization: (MAXQ10) This register is cleared to 0000h on all forms of reset.
(MAXQ20) This register is cleared to 001Ch on all forms of reset.
Access: Unrestricted direct read/write access.

BIT	FUNCTION		
DPC.1 to DPC.0 (SDPS1, SDPS0)	Source Data Pointer Select Bits 1:0. These bits select one of the three data pointers as the active source pointer for the load operation. A new data pointer must be selected before being used to read data memory:		
	SDPS1	SDPS0	SOURCE POINTER SELECTION
	0	0	DP[0]
	0	1	DP[1]
	1	0	FP (BP[Offs])
	1	1	Reserved (select FP if set)
	These bits default to 00b but do not activate DP[0] as an active source pointer until the SDPS bits are explicitly cleared to 00b or the DP[0] register is written by an instruction. Also, modifying the register contents of a data/frame pointer register (DP[0], DP[1], BP or Offs) will change the setting of the SDPS bits to reflect the active source pointer selection.		
DPC.2 (WBS0)	Word/Byte Select 0. This bit selects access mode for DP[0]. When WBS0 is set to logic 1, the DP[0] is operated in word mode for data memory access; when WBS0 is cleared to logic 0, DP[0] is operated in byte mode for data memory access.		
DPC.3 (WBS1)	Word/Byte Select 1. This bit selects access mode for DP[1]. When WBS1 is set to logic 1, the DP[1] is operated in word mode for data memory access; when WBS1 is cleared to logic 0, DP[1] is operated in byte mode for data memory access.		
DPC.4 (WBS2)	Word/Byte Select 2. This bit selects access mode for BP[Offs]. When WBS2 is set to logic 1, the BP[Offs] is operated in word mode for data memory access; when WBS2 is cleared to logic 0, BP[Offs] is operated in byte mode for data memory access.		
DPC.15 to DPC.5	Reserved. Read returns 0.		

4.20 General Register (GR, Eh[5h])

Initialization: This register is cleared to 0000h on all forms of reset.
Access: Unrestricted direct read/write access.

BIT	FUNCTION
GR.15 to GR.0	This register is intended primarily for supporting byte operations on 16-bit data. The 16-bit register is byte-readable, byte-writeable through the corresponding GRL and GRH 8-bit registers and byte-swappable through the GRS 16-bit register.

4.21 General Register Low Byte (GRL, Eh[6h])

Initialization: This register is cleared to 00h on all forms of reset.
Access: Unrestricted direct read/write access.

BIT	FUNCTION
GRL.7 to GRL.0	This register reflects the low byte of the GR register and is intended primarily for supporting byte operations on 16-bit data. Any data written to the GRL register will also be stored in the low byte of the GR register.

4.22 Frame Pointer Base Register (BP, Eh[7h])

Initialization: This register is cleared to 0000h on all forms of reset.
Access: Unrestricted direct read/write access.

BIT	FUNCTION
BP.15 to BP.0	This register serves as the base pointer for the Frame Pointer (FP). The Frame Pointer is formed by unsigned addition of Frame Pointer Base Register (BP) and Frame Pointer Offset Register (Offs). The content of this base pointer register is not affected by increment/decrement operations performed on the offset (OFFS) register.

SECTION 5: PERIPHERAL REGISTER MODULES

The MAXQ microcontroller uses Peripheral Registers to control and monitor peripheral modules. These registers reside in Modules 0h through 5h, with sub-index values 0h to 1Fh. While the peripherals must reside in modules 0h through 5h, they are not necessarily tied to specific index numbers inside this range and can be moved, removed, and/or duplicated for certain MAXQ-based products as space permits. For this reason, each peripheral modules and its associated registers/bits are covered separately. Consult the individual device data sheet or user's guide supplement for the exact peripheral register map.

SECTION 6: GENERAL-PURPOSE I/O MODULE

This section contains the following information:

6.1 I/O Port: Type A 6-2
6.2 I/O Port: Type B 6-2
6.3 I/O Port: Type C 6-3
6.4 I/O Port: Type D 6-3
6.5 I/O Port Peripheral Registers 6-4
6.5.1 Port Output x Register (POx) 6-4
6.5.2 Port Input x Register (PIx) 6-4
6.5.3 Port Direction x Register (PDx) 6-4
6.5.4 (Type A) External Interrupt Enable Register (EIEx) 6-5
6.5.5 (Type A) External Interrupt Flag Register (EIFx) 6-6
6.5.6 (Type D) External Interrupt Enable Register (EIEx) 6-6
6.5.7 (Type D) External Interrupt Flag Register (EIFx) 6-7
6.5.8 (Type D) External Interrupt Edge Select Register (EIESx) 6-8

LIST OF FIGURES

Figure 6-1. Type A Port Pin Schematic 6-2
Figure 6-2. Type B Port Pin Schematic 6-2
Figure 6-3. Type C Port Pin Schematic 6-3
Figure 6-4. Type D Port Pin Schematic 6-3

LIST OF TABLES

Table 6-1. Weak Pullup Control 6-3

6.5.5 (Type A) External Interrupt Flag Register (EIFx)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	—	—	IE5	IE4	IE3	IE2	IE1	IE0
Reset	0	0	0	0	0	0	0	0
Access	r	r	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 7 and 6: Reserved

Bit 5: External Interrupt 5 Flag (IE5). This flag is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the INT5 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

Bit 4: External Interrupt 4 Flag (IE4). This flag is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the INT4 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

Bit 3: External Interrupt 3 Flag (IE3). This flag is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the INT3 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

Bit 2: External Interrupt 2 Flag (IE2). This flag is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the INT2 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

Bit 1: External Interrupt 1 Flag (IE1). This flag is set when a negative edge (IT0 = 1) or a positive edge (IT0 = 0) is detected on the INT1 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

Bit 0: External Interrupt 0 Flag (IE0). This flag is set when a negative edge (IT0 = 1) or a positive edge (IT0 = 0) is detected on the INT0 pin. This bit remains set until cleared in software. Setting this bit by software causes an interrupt if enabled.

6.5.6 (Type D) External Interrupt Enable Register (EIEx)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bit 7: Enable External Interrupt 7 (EX7)

0 = external interrupt 7 function disabled

1 = external interrupt 7 function enabled

Bit 6: Enable External Interrupt 6 (EX6)

0 = external interrupt 6 function disabled

1 = external interrupt 6 function enabled.

Bit 5: Enable External Interrupt 5 (EX5)

0 = external interrupt 5 function disabled

1 = external interrupt 5 function enabled

Bit 4: Enable External Interrupt 4 (EX4)

0 = external interrupt 4 function disabled

1 = external interrupt 4 function enabled

6.5.8 (Type D) External Interrupt Edge Select Register (EIESx)

Bit #	7	6	5	4	3	2	1	0
Name	IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bit 7: Interrupt 7 Edge Select (IT7). This bit selects the edge detection mode for external interrupt 7.

0 = INT7 is positive-edge triggered

1 = INT7 is negative-edge triggered

Bit 6: Interrupt 6 Edge Select (IT6). This bit selects the edge detection mode for external interrupt 6.

0 = INT6 is positive-edge triggered

1 = INT6 is negative-edge triggered

Bit 5: Interrupt 5 Edge Select (IT5). This bit selects the edge detection mode for external interrupt 5.

0 = INT5 is positive-edge triggered

1 = INT5 is negative-edge triggered

Bit 4: Interrupt 4 Edge Select (IT4). This bit selects the edge detection mode for external interrupt 4.

0 = INT4 is positive-edge triggered

1 = INT4 is negative-edge triggered

Bit 3: Interrupt 3 Edge Select (IT3). This bit selects the edge detection mode for external interrupt 3.

0 = INT3 is positive-edge triggered

1 = INT3 is negative-edge triggered

Bit 2: Interrupt 2 Edge Select (IT2). This bit selects the edge detection mode for external interrupt 2.

0 = INT2 is positive-edge triggered

1 = INT2 is negative-edge triggered

Bit 1: Interrupt 1 Edge Select (IT1). This bit selects the edge detection mode for external interrupt 1.

0 = INT1 is positive-edge triggered

1 = INT1 is negative-edge triggered

Bit 0: Interrupt 0 Edge Select (IT0). This bit selects the edge detection mode for external interrupt 0.

0 = INT0 is positive-edge triggered

1 = INT0 is negative-edge triggered

SECTION 7: TIMER/COUNTER 0 MODULE

This section contains the following information:

7.1 Timer 0 7-2
7.1.1 Timer 0 Mode: 13-Bit Timer/Counter 7-2
7.1.2 Timer 0 Mode: 16-Bit Timer/Counter 7-3
7.1.3 Timer 0 Mode: 8-Bit Timer with Auto-Reload 7-3
7.1.4 Timer 0 Mode: Two 8-Bit Timer/Counters 7-4
7.2 Timer/Counter 0 Peripheral Registers 7-5
7.2.1 Timer/Counter 0 Control Register (T0CN) 7-5
7.2.2 Timer/Counter 0 High Register (T0H) 7-6
7.2.3 Timer/Counter 0 Low Register (T0L) 7-6

LIST OF FIGURES

Figure 7-1. Timer/Counter 0 13-Bit/16-Bit Modes 7-3
Figure 7-2. Timer/Counter 0 8-Bit Auto-Reload Mode 7-3
Figure 7-3. Timer/Counter 0 Dual 8-Bit Mode 7-4

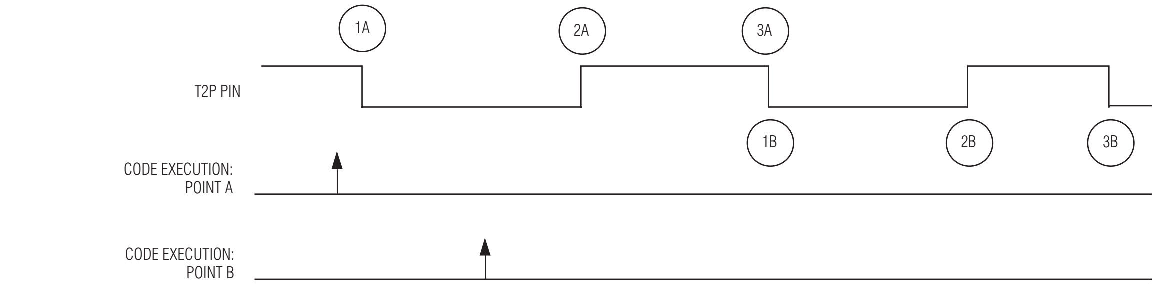
LIST OF TABLES

Table 7-1. Timer 0 Mode Summary 7-2

9.3.4 Measure Duty Cycle Repeatedly

To measure the duty cycle of the signal seen on the T2P input pin, Timer 2 could be configured for a single-shot delayed run with both edges defined for capture. The CPRL2 bits should be configured to 1 to request reloads on each edge. To prevent reloads on one of the edges, gating should be enabled. The T2POL[0] bit specifies which edge starts/ends the capture cycle and which edge does not have a reload associated with it.

```
; ------------------ Reset State:  T2R = T2V = T2C = 0000h ------------------------
MOVE T2CFG, #00000110b         ; T2CI          =0     (sysclk/N input)
                               ; T2DIV[2:0]    =000   (/1)
                               ; T2MD          =0     (16-bit)
                               ; CCF[1:0]      =11    (both edges)
                               ; C/T2          =0     (timer/capture)
MOVE T2CNA, #10101111b         ; ET2           =1     (enable Timer 2 ints)
                               ; T2OE[0]       =0     (input)
                               ; T2POL[0]      =1     (no reload on rising edge
;                                                     single-shot start/end on falling edge)
                               ; TR2L:TR2      =01    (start timer 2 on single shot condition)
                               ; CPRL2         =1     (reload on capture edge)
                               ; SS2           =1     (single-shot mode)
                               ; G2EN          =1     (gating enabled)
; ------------------ TCC2 Interrupt : LOW TIME=T2C
;------------------- TCC2 Interrupt : PERIOD = T2C
```



EVENTS:
1A: FALLING EDGE CAUSES CAPTURE/RELOAD; SINGLE-SHOT CAPTURE CYCLE BEGINS.
2A: RISING EDGE CAUSES CAPTURE; LOW TIME = T2C.
3A: FALLING EDGE CAUSES CAPTURE/RELOAD; SINGLE-SHOT CAPTURE CYCLE ENDS; PERIOD = T2C. TIMER CONTINUES TO OPERATE SINCE TR2 = 1, ALLOWING THE NEXT LOW TIME/PERIOD TO BE MEASURED.
1B–3B: SAME SEQUENCE AS 1A–3A, EXCEPT THAT THE SINGLE-SHOT CAPTURE CYCLE DOES NOT BEGIN UNTIL THE FIRST FALLING EDGE IS DETECTED.

Figure 9-7. Timer 2 Application Example—Measure Duty Cycle

12.5.3 Multiplier Operand B Register (MB)

Bit #	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	MB.15	MB.14	MB.13	MB.12	MB.11	MB.10	MB.9	MB.8
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MB.7	MB.6	MB.5	MB.4	MB.3	MB.2	MB.1	MB.0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: Multiplier Operand B Register (MB.[15:0]). This operand B register is used by the application code to load 16-bit values for multiplier operations.

12.5.4 Multiplier Accumulator 2 Register (MC2)

Bit #	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	MC2.15	MC2.14	MC2.13	MC2.12	MC2.11	MC2.10	MC2.9	MC2.8
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	MC2.7	MC2.6	MC2.5	MC2.4	MC2.3	MC2.2	MC2.1	MC2.0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: Multiplier Accumulator 2 Register (MC2.[15:0]). The MC2 register represents the two most significant bytes of the accumulator register. The 48-bit accumulator is formed by MC2, MC1 and MC0. For a signed operation, the most significant bit of this register is the sign bit. The MC2 register width is dependent upon the hardware multiplier accumulator width of the given MAXQ device. For a MAXQ device having only a 32-bit accumulator, the MC2 register will not be present.

SECTION 16: IN-CIRCUIT DEBUG MODE

This section contains the following information:

16.1 Background Mode Operation 16-3
16.1.1 Breakpoint Registers 16-5
16.1.1.1 Breakpoint 0 Register (BP0) 16-5
16.1.1.2 Breakpoint 1 Register (BP1) 16-5
16.1.1.3 Breakpoint 2 Register (BP2) 16-6
16.1.1.4 Breakpoint 3 Register (BP3) 16-6
16.1.1.5 Breakpoint 4 Register (BP4) (REGE = 0) 16-7
16.1.1.6 Breakpoint 4 Register (BP4) (REGE = 1) 16-7
16.1.1.7 Breakpoint 5 Register (BP5) (REGE = 0) 16-8
16.1.1.8 Breakpoint 5 Register (BP5) (REGE = 1) 16-8
16.1.2 Using Breakpoints 16-9
16.2 Debug Mode 16-9
16.2.1 Debug Mode Commands 16-10
16.2.2 Read Register Map Command Host-ROM Interaction 16-12
16.2.3 Single-Step Operation (Trace) 16-12
16.2.4 Return 16-12
16.2.5 Debug Mode Special Considerations 16-12
16.3 In-Circuit Debug Peripheral Registers 16-13
16.3.1 In Circuit Debug Temp 0 Register (ICDT0) 16-13
16.3.2 In Circuit Debug Temp 1 Register (ICDT1) 16-14
16.3.3 In-Circuit Debug Control Register (ICDC) 16-14
16.3.4 In-Circuit Debug Flag Register (ICDF) 16-15
16.3.5 In-Circuit Debug Buffer Register (ICDB) 16-15
16.3.6 In Circuit Debug Data Register (ICDD) 16-16
16.3.7 In Circuit Debug Address Register (ICDA) 16-16

LIST OF FIGURES

Figure 16-1. In-Circuit Debugger 16-2

LIST OF TABLES

Table 16-1. Background Mode Commands 16-4
Table 16-2. Debug Mode Commands 16-11

SECTION 17: IN-SYSTEM PROGRAMMING (JTAG)

Internal nonvolatile memory of MAXQ microcontrollers can be initialized via Bootstrap Loader mode. To enable the Bootstrap loader and establish a desired communication channel, the System Programming instruction (100b) must be loaded into the TAP instruction register using the IR-Scan sequence. Once the instruction is latched in the instruction parallel buffer (IR2:0) and is recognized by the TAP controller in the Update-IR state, a 3-bit data shift register is activated as the communication channel for DR-Scan sequences. The TAP retains the System Programming instruction until a new instruction is shifted in or the TAP controller returns to the Test-Logic-Reset state. This 3-bit shift register formed between the TDI and TDO pins is directly interfaced to the 3-bit Serial Programming Buffer (SPB). The System Programming Buffer (SPB) contains three bits with the following functions:

- SPB.0: System Programming Enable (SPE). Setting this bit to logic 1 denotes that system programming is desired upon exiting reset. When it is cleared to logic 0, no system programming is needed. The logic state of SPE is examined by the reset vector in the Utility ROM to determine the program flow after a reset. When SPE = 1, the Bootstrap Loader selected by the PSS1:0 bits are activated to perform a Bootstrap Loader function. When SPE = 0, the Utility ROM transfers execution control to the normal user program.
- SPB.2:1: Programming Source Select (PSS1:PSS0). These bits allow the host to select programming interface sources. The PSS bits have no functions when the SPE bit is cleared.

PSS1	PSS0	PROGRAMMING SOURCE
0	0	JTAG
0	1	UART
1	0	SPI
1	1	Reserved

The DR-Scan sequence is used to configure the SPB bits. The data content of the SPB register is reflected in the ICDF register and allows read/write access by the CPU. These bits are cleared by power-on reset or Test-Logic-Reset of the TAP controller.

17.1 JTAG Bootloader Operation

Devices that support a JTAG bootloader have the benefit of using the same status bit handshaking hardware as is used for in-circuit debugging. When the SPE bit of the System Programming Buffer (SPB) is set to 1 and JTAG is selected as the programming source (PSS1:0 = 00b), the background and active debug mode state machines are disabled. Once the host loads the Debug instruction into the TAP instruction register (IR2:0), the 10-bit shift register interface to ICDB and the status bits becomes available for host-to-ROM bootloader communication. The status bits should be interpreted as follows for JTAG bootloader operation:

BITS 1:0	STATUS	CONDITION
00	Reserved	Invalid condition.
01	Reserved	Invalid condition
10	Loader-Busy	ROM Loader is busy executing code or processing the current command.
11	Loader-Valid	ROM Loader is supplying valid output data to the host in current shift operation.

When the using the JTAG bootloader option (SPE = 1, PSS1:0 = 00b), the sole purpose of the debug hardware is to simultaneously transfer the data byte shifted in from the host into the ICDB register and transfer the contents of an internal holding register (loaded by ROM code writes of ICDB) into the shift register for output to the host. This transfer takes place on the falling edge of TCK at the Update-DR state. The debug hardware additionally clears the TXC bit at this point in the state diagram. The ROM loader code controls the status bit output to the host by asserting TXC = 1 when it has valid data to be shifted out. The ROM code may flexibly implement whatever communication protocol and command set it wishes within the data byte portion of the shifted 10-bit word.

CMP src — Compare Accumulator

Description: Compare for equality between the active accumulator and the least significant byte of the specified src. The MAXQ20 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: E

Operation: Acc = src: E ← 1

Acc <> src: E ← 0

Encoding: 15 … 0

f111	1000	ssss	ssss

MAXQ10 Example(s):

```
CMP A[1]      ; Acc = 45h, A[1] = 10h, E=0
CMP  #45h     ; Acc = 45h, E=1
CMP DP[0]     ; Acc = 45h, DP[0]=0345h, E=1
```

MAXQ20 Example(s):

```
CMP  #45h     ; Acc = 0145h, E=0
CMP  #145h    ; PFX[0] register used
              ; MOVE PFX[0], #01h (smart-prefixing)
              ; CMP  #45h  E=1
```

CPL — Complement Acc

Description: Performs a logical bitwise complement (1's complement) on the active accumulator (Acc or A[AP]) and returns the result to the active accumulator.

Status Flags: S, Z

Operation: Acc ← ~Acc

Encoding: 15 … 0

1000	1010	0001	1010

MAXQ10 Example(s):

```
              ; Acc = FFh, S=1, Z=0
CPL           ; Acc ← 00h, S=0, Z=1
              ; Acc = 09h, S=0, Z=0
CPL           ; Acc ← F6h, S=1, Z=0
```

MAXQ20 Example(s):

```
              ; Acc = FFFFh, S=1, Z=0
CPL           ; Acc ← 0000h, S=0, Z=1
              ; Acc = 0990h, S=0, Z=0
CPL           ; Acc ← F66Fh, S=1, Z=0
```

OR Acc.<b> **Logical OR Carry Flag with Accumulator Bit**

Description: Performs a logical-OR between the Carry (C) status flag and a specified bit of the active accumulator (Acc.<b>) and returns the result to the Carry.

Status Flags: C

Operation: C ← C OR Acc.<b>

Encoding: 15 0

1010	1010	bbbb	1010

MAXQ10 Example(s):

```
                    ; Acc = 45h, C=0 at start
OR Acc.1            ; Acc.1=0   → C=0
OR Acc.2            ; Acc.2=1   → C=1
```

MAXQ20 Example(s):

```
                    ; Acc = 2345h, C=0 at start
OR Acc.1            ; Acc.1=0   → C=0
OR Acc.2            ; Acc.2=1   → C=1
```

Special Notes: For the MAXQ10, the accumulator width is only 8 bits. Thus, only bit index encoding ('bbbb') for bits 0 ('0000') through 7 ('0111') is supported.

POP dst **Pop Word from the Stack**

Description: Pops a single word from the stack (@SP) to the specified dst and decrements the stack pointer (SP).

Status Flags: S, Z (if dst = Acc or AP or APC)
C, E (if dst = PSF)

Operation: dst ← @ SP--

Encoding: 15 0

1ddd	dddd	0000	1101

Example(s):

```
                    ; GR ← 1234h
POP GR              ; @DP[0] ← 76h (WBS0=0)
POP @DP[0]          ; @DP[0] ← 0876h (WBS0=1)
```

Stack Data:

xxxxh	
1234h	← SP (initial)
0876h	← SP (after POP GR)
xxxxh	← SP (after POP @DP[0])
xxxxh	

RETI **Return from Interrupt**

Description: RETI pops a single word from the stack (@SP) into the Instruction Pointer (IP) and decrements the stack pointer (SP). Additionally, RETI returns the interrupt logic to a state in which it can acknowledge additional interrupts.

Status Flags: None

Operation: IP ← @SP--

INS ← 0

Encoding: 15 0

1000	1100	1000	1101

Example(s): See RETI

RETI C/RETI NC, RETI Z/RETI NZ, RETI S **Conditional Return from Interrupt on Status Flag**

Description: Performs conditional return (RETI) based upon the state of a specific processor status flag. RETI C returns if the Carry flag is set while RETI NC returns if the Carry flag is clear. RETI Z returns if the Zero flag is set while RETI NZ returns if the Zero flag is clear. RETI S returns if the Sign flag is set. See RETI for additional information on the return operation.

Status Flags: None

RETI C

Operation: C=1: IP ← @SP--

INS ← 0

C=0: IP ← IP + 1

Encoding: 15 0

1010	1100	1000	1101

Example(s): RETI C ; C=1, return from interrupt (RETI) is performed

RETI NC

Operation: C=0: IP ← @SP--

INS ← 0

C=1: IP ← IP +1

Encoding: 15 0

1110	1100	1000	1101

Example(s): RETI NC ; C=1, return from interrupt (RETI) does not occur

RETI Z

Operation: Z=1: IP ← @SP--

INS ← 0

Z=0: IP ← IP + 1

Encoding: 15 0

1001	1100	1000	1101

Example(s): RETI Z ; Z=0, return from interrupt (RETI) does not occur

---

RETI NZ

Operation: Z=0: IP ← @SP--

INS ←0

Z=1: IP ← IP +1

Encoding: 15 0

1101	1100	1000	1101

Example(s): RETI NZ ; Z=0, return from interrupt (RETI) is performed

---

RETI S

Operation: S=1: IP ← @SP--

INS ← 0

S=0: IP ← IP + 1

Encoding: 15 0

1100	1100	1000	1101

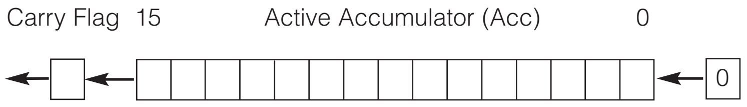
Example(s): RETI S ; S=0, return from interrupt (RETI) does not occur

(MAXQ20 Version) **Shift Accumulator Left Arithmetically**
SLA/SLA2/SLA4 **One, Two, or Four Times**

Description: Shifts the active accumulator left once, twice, or four times respectively for SLA, SLA2, and SLA4. For each shift iteration, a 0 is shifted into the lsb, and the msb is shifted into the Carry (C) flag. For signed data, this shifting process effectively retains the sign orientation of the data to the point at which overflow/underflow would occur.

Status Flags: C, S, Z

SLA Operation:



C ← Acc.15; Acc.[15:1]← Acc.[14:0]; Acc.0 ← 0

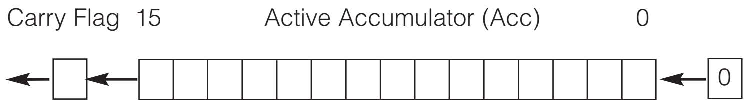
Encoding:

15			0
1000	1010	0010	1010

Example(s):

```
                        ; Acc = E345h, C=0, S=1, Z=0
SLA                     ; Acc = C68h, C=1, S=1, Z=0
SLA                     ; Acc = 8D14h, C=1, S=1, Z=0
```

SLA2 Operation:



C ← Acc.14 ; Acc.[15:2]← Acc.[13:0] ; Acc.[1:0] ← 0

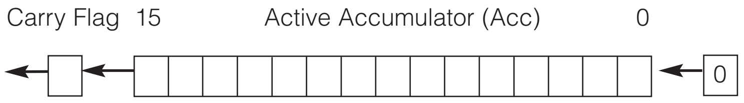
Encoding:

15			0
1000	1010	0011	1010

Example(s):

```
                        ; Acc = E345h, C=0, S=1, Z=0
SLA2                    ; Acc = 8D14h, C=1, S=1, Z=0
```

SLA4 Operation:



C ← Acc.12; Acc.[15:4]← Acc.[11:0]; Acc.[3:0] ← 0

Encoding:

15			0
1000	1010	0110	1010

Example(s):

```
                        ; Acc = E345h, C=0, S=1, Z=0
SLA4                    ; Acc = 3450h, C=0, S=0, Z=0
```

SUB/SUBB src **Subtract /Subtract with Borrow**

Description: Subtracts the specified src from the active accumulator (Acc) and returns the result back to the active accumulator. The SUBB additionally subtracts the borrow (Carry Flag), which may have resulted from previous subtraction. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ20 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: C, S, Z, OV

SUB Operation: Acc ← Acc - src

Encoding: 15 0

f101	1010	ssss	ssss

MAXQ10 Example(s):

```
                        ; Acc = 23h to start, A[1]= 12h
SUB    A[1]             ; Acc = 11h, C=0, S=0, Z=0
SUB    A[1]             ; Acc = FFh, C=1, S=1, Z=0
```

MAXQ20 Example(s):

```
                        ; Acc = 2345h to start, A[1]= 1250h
SUB    A[1]             ; Acc = 10F5h, C=0, S=0, Z=0, OV=0
SUB    A[1]             ; Acc = FEA5h, C=1, S=1, Z=0, OV=0
SUB    A[2]             ; A[2] =7FFFh
                        ; → Acc = 7EA6h; C=0, S=0, Z=0, OV=1
```

SUBB Operation: Acc ← Acc - (src + C)

Encoding: 15 0

f111	1010	ssss	ssss

MAXQ10 Example(s):

```
                        ; Acc = 23h, A[1]= 12h, C=1
SUBB   A[1]             ; Acc = 10h, C=0, S=0, Z=0
SUBB   A[1]             ; Acc = FEh, C=1, S=1, Z=0
SUBB   #0Dh             ; Acc = F0h, C=0, S=1, Z=0
```

MAXQ20 Example(s):

```
                        ; Acc = 2345h, A[1]= 1250h, C=1
SUBB   A[1]             ; Acc = 10F4h, C=0, S=0, Z=0
SUBB   A[1]             ; Acc = FEA4h, C=1, S=1, Z=0
```

Special Notes: The active accumulator (Acc) is not allowed as the src for these operations.