

# 68HC08KL8

## General Release Specification

August 10, 1998

Personal and PC Media Division  
Austin, Texas



*Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.*

## List of Sections

<b>Section 1. General Description</b>	<b>23</b>
<b>Section 2. Memory Map</b>	<b>35</b>
<b>Section 3. Random-Access Memory (RAM)</b>	<b>49</b>
<b>Section 4. Read-Only Memory (ROM)</b>	<b>51</b>
<b>Section 5. Mask Option Register (MOR)</b>	<b>53</b>
<b>Section 6. Central Processor Unit (CPU)</b>	<b>57</b>
<b>Section 7. Oscillator</b>	<b>73</b>
<b>Section 8. System Integration Module (SIM)</b>	<b>77</b>
<b>Section 9. Universal Serial Bus Module (USB)</b>	<b>103</b>
<b>Section 10. Monitor ROM (MON)</b>	<b>151</b>
<b>Section 11. Timer Interface Module (TIM)</b>	<b>165</b>
<b>Section 12. Input/Output Ports (I/O)</b>	<b>189</b>
<b>Section 13. Computer Operating Properly (COP)</b>	<b>211</b>
<b>Section 14. External Interrupt (IRQ)</b>	<b>217</b>
<b>Section 15. Keyboard Interrupt Module (KBI)</b>	<b>225</b>
<b>Section 16. Break Module (BREAK)</b>	<b>233</b>
<b>Section 17. Electrical Specifications</b>	<b>239</b>
<b>Section 18. Mechanical Specifications</b>	<b>249</b>
<b>Section 19. Ordering Information</b>	<b>253</b>



## Table of Contents

### Section 1. General Description

1.1	Contents . . . . .	23
1.2	Introduction . . . . .	24
1.3	Features . . . . .	24
1.4	MCU Block Diagram . . . . .	26
1.5	Pin Assignments . . . . .	28
1.5.1	Power Supply Pins ( $V_{DDREG}$ , $V_{SSREG}$ , $V_{DD1}$ , $V_{SS1}$ , $V_{DD2}$ , and $V_{SS2}$ ) . . . . .	30
1.5.2	Voltage Regulator Out (REGOUT) . . . . .	31
1.5.3	Oscillator Pins (OSC1 and OSC2) . . . . .	31
1.5.4	External Reset Pin ( $\overline{RST}$ ) . . . . .	31
1.5.5	External Interrupt Pin ( $\overline{IRQ1}$ ) . . . . .	32
1.5.6	USB Data Pins (D+ and D-) . . . . .	32
1.5.7	Port A Input/Output (I/O) Pins (PTA7–PTA0) . . . . .	32
1.5.8	Port B (I/O) Pins (PTB7–PTB0) . . . . .	32
1.5.9	Port C I/O Pins (PTC7–PTC0) . . . . .	32
1.5.10	Port D I/O Pins (PTD7/ $\overline{KBD7}$ –PTD0/ $\overline{KBD0}$ ) . . . . .	33
1.5.11	Port E I/O Pins (PTE6–PTE3 and PTE2/TCH1, PTE1/TCH0, PTE0/TCLK) . . . . .	33

## Section 2. Memory Map

2.1	Contents . . . . .	35
2.2	Introduction . . . . .	35
2.3	Input/Output (I/O) Section . . . . .	37
2.4	Unimplemented Memory Locations . . . . .	37
2.5	Reserved Memory Locations . . . . .	37
2.6	Monitor ROM . . . . .	47

## Section 3. Random-Access Memory (RAM)

3.1	Contents . . . . .	49
3.2	Introduction . . . . .	49
3.3	Functional Description . . . . .	49

## Section 4. Read-Only Memory (ROM)

4.1	Contents . . . . .	51
4.2	Introduction . . . . .	51
4.3	Functional Description . . . . .	51

## Section 5. Mask Option Register (MOR)

5.1	Contents . . . . .	53
5.2	Introduction . . . . .	53
5.3	Functional Description . . . . .	54

## Section 6. Central Processor Unit (CPU)

6.1	Contents . . . . .	57
6.2	Introduction . . . . .	57
6.3	Features . . . . .	58
6.4	CPU Registers . . . . .	59
6.4.1	Accumulator . . . . .	59
6.4.2	Index Register . . . . .	60
6.4.3	Stack Pointer . . . . .	61
6.4.4	Program Counter . . . . .	62
6.4.5	Condition Code Register . . . . .	63
6.5	Arithmetic/Logic Unit (ALU) . . . . .	65
6.6	Instruction Set Summary . . . . .	65
6.7	Opcode Map . . . . .	71

## Section 7. Oscillator

7.1	Contents . . . . .	73
7.2	Introduction . . . . .	73
7.3	Oscillator External Connections . . . . .	74
7.4	I/O Signals . . . . .	75
7.4.1	Crystal Amplifier Input Pin (OSC1) . . . . .	75
7.4.2	Crystal Amplifier Output Pin (OSC2) . . . . .	75
7.4.3	Oscillator Enable Signal (SIMOSCEN) . . . . .	75
7.4.4	External Clock Source (CGMXCLK) . . . . .	75
7.4.5	Oscillator Out (CGMOUT) . . . . .	76
7.5	Low-Power Modes . . . . .	76
7.5.1	Wait Mode . . . . .	76
7.5.2	Stop Mode . . . . .	76
7.6	Oscillator During Break Mode . . . . .	76

## Section 8. System Integration Module (SIM)

8.1	Contents . . . . .	77
8.2	Introduction . . . . .	78
8.3	SIM Bus Clock Control and Generation . . . . .	81
8.3.1	Bus Timing . . . . .	82
8.3.2	Clock Startup from POR . . . . .	82
8.3.3	Clocks in Stop Mode and Wait Mode . . . . .	82
8.4	Reset and System Initialization. . . . .	82
8.4.1	External Pin Reset. . . . .	83
8.4.2	Active Resets from Internal Sources. . . . .	84
8.4.2.1	Power-On Reset. . . . .	85
8.4.2.2	Computer Operating Properly (COP) Reset. . . . .	86
8.4.2.3	Illegal Opcode Reset . . . . .	86
8.4.2.4	Illegal Address Reset . . . . .	86
8.4.2.5	Universal Serial Bus Reset . . . . .	87
8.5	SIM Counter . . . . .	87
8.5.1	SIM Counter During Power-On Reset. . . . .	87
8.5.2	SIM Counter During Stop Mode Recovery . . . . .	88
8.5.3	SIM Counter and Reset States . . . . .	88
8.6	Exception Control . . . . .	88
8.6.1	Interrupts . . . . .	88
8.6.1.1	Hardware Interrupts . . . . .	91
8.6.1.2	SWI Instruction. . . . .	92
8.6.2	Interrupt Status Registers . . . . .	92
8.6.2.1	Interrupt Status Register 1 . . . . .	93
8.6.2.2	Interrupt Status Register 2 . . . . .	94
8.6.3	Reset. . . . .	95
8.6.4	Break Interrupts. . . . .	95
8.6.5	Status Flag Protection in Break Mode. . . . .	95
8.7	Low-Power Modes . . . . .	96
8.7.1	Wait Mode . . . . .	96
8.7.2	Stop Mode. . . . .	97



8.8	SIM Registers . . . . .	99
8.8.1	Break Status Register . . . . .	99
8.8.2	Reset Status Register . . . . .	101
8.8.3	Break Flag Control Register . . . . .	102

## Section 9. Universal Serial Bus Module (USB)

9.1	Contents . . . . .	103
9.2	Features . . . . .	105
9.3	Overview . . . . .	106
9.3.1	USB Protocol . . . . .	107
9.3.1.1	Sync Pattern . . . . .	109
9.3.1.2	Packet Identifier Field . . . . .	110
9.3.1.3	Address Field (ADDR) . . . . .	110
9.3.1.4	Endpoint Field (ENDP) . . . . .	110
9.3.1.5	Cyclic Redundancy Check (CRC) . . . . .	111
9.3.1.6	End-of-Packet (EOP) . . . . .	113
9.3.2	Reset Signaling . . . . .	114
9.3.3	Suspend . . . . .	115
9.3.4	Resume After Suspend . . . . .	115
9.3.4.1	Host Initiated Resume . . . . .	115
9.3.4.2	USB Reset Signalling . . . . .	116
9.3.4.3	Remote Wakeup . . . . .	116
9.3.5	Low-Speed Device . . . . .	117
9.4	Clock Requirements . . . . .	117
9.5	Hardware Description . . . . .	118
9.5.1	Voltage Regulator . . . . .	118
9.5.2	Regulator Bypass Option . . . . .	119
9.5.3	USB Transceiver . . . . .	120
9.5.3.1	Output Driver Characteristics . . . . .	120
9.5.3.2	Low Speed (1.5 Mbs) Driver Characteristics . . . . .	120
9.5.3.3	Receiver Data Jitter . . . . .	122
9.5.3.4	Data Source Jitter . . . . .	122
9.5.3.5	Data Signal Rise and Fall Time . . . . .	123

## Table of Contents

9.5.4	USB Control Logic . . . . .	124
9.5.4.1	Data Encoding/Decoding . . . . .	125
9.5.4.2	Bit Stuffing . . . . .	126
9.6	I/O Register Description . . . . .	127
9.6.1	USB Address Register . . . . .	131
9.6.2	USB Interrupt Register 0 . . . . .	132
9.6.3	USB Interrupt Register 1 . . . . .	134
9.6.4	USB Control Register 0 . . . . .	136
9.6.5	USB Control Register 1 . . . . .	138
9.6.6	USB Control Register 2 . . . . .	140
9.6.7	USB Status Register . . . . .	142
9.6.8	USB Endpoint 0 Data Registers . . . . .	143
9.6.9	USB Endpoint 1/Endpoint 2 Data Registers . . . . .	144
9.7	USB Interrupts . . . . .	145
9.7.1	USB End-of-Transaction Interrupt . . . . .	145
9.7.1.1	Receive Control Endpoint 0 . . . . .	145
9.7.1.2	Transmit Control Endpoint 0 . . . . .	148
9.7.1.3	Transmit Endpoint 1 and Transmit Endpoint 2 . . . . .	149
9.7.2	Resume Interrupt . . . . .	150
9.7.3	End-of-Packet Interrupt . . . . .	150

## Section 10. Monitor ROM (MON)

10.1	Contents . . . . .	151
10.2	Introduction . . . . .	151
10.3	Features . . . . .	152
10.4	Functional Description . . . . .	152
10.4.1	Entering Monitor Mode . . . . .	154
10.4.2	Data Format . . . . .	156
10.4.3	Break Signal . . . . .	156
10.4.4	Baud Rate . . . . .	157
10.4.5	Commands . . . . .	157
10.5	Security . . . . .	163

## Section 11. Timer Interface Module (TIM)

11.1	Contents .....	165
11.2	Introduction .....	166
11.3	Features .....	166
11.4	Functional Description .....	166
11.4.1	TIM Counter Prescaler .....	170
11.4.2	Input Capture .....	170
11.4.3	Output Compare .....	170
11.4.3.1	Unbuffered Output Compare .....	170
11.4.3.2	Buffered Output Compare .....	171
11.4.4	Pulse Width Modulation (PWM) .....	172
11.4.4.1	Unbuffered PWM Signal Generation .....	173
11.4.4.2	Buffered PWM Signal Generation .....	174
11.4.4.3	PWM Initialization .....	174
11.5	Interrupts .....	176
11.6	Wait Mode .....	176
11.7	TIM During Break Interrupts .....	177
11.8	I/O Signals .....	177
11.8.1	TIM Clock Pin (PTE0/TCLK) .....	178
11.8.2	TIM Channel I/O Pins (PTE1/TCH0:PTE2/TCH1) .....	178
11.9	I/O Registers .....	178
11.9.1	TIM Status and Control Register .....	179
11.9.2	TIM Counter Registers .....	181
11.9.3	TIM Counter Modulo Registers .....	182
11.9.4	TIM Channel Status and Control Registers .....	183
11.9.5	TIM Channel Registers .....	187

## Section 12. Input/Output Ports (I/O)

12.1	Contents .....	189
12.2	Introduction .....	190
12.3	Port A .....	192
12.3.1	Port A Data Register .....	192
12.3.2	Data Direction Register A .....	193
12.4	Port B .....	195
12.4.1	Port B Data Register .....	195
12.4.2	Data Direction Register B .....	196
12.5	Port C .....	198
12.5.1	Port C Data Register .....	198
12.5.2	Data Direction Register C .....	199
12.6	Port D .....	201
12.6.1	Port D Data Register .....	201
12.6.2	Data Direction Register D .....	202
12.7	Port E .....	204
12.7.1	Port E Data Register .....	204
12.7.2	Data Direction Register E .....	206
12.8	Port Options .....	208
12.8.1	Port Option Control Register .....	208

## Section 13. Computer Operating Properly (COP)

13.1	Contents .....	211
13.2	Introduction .....	211
13.3	Functional Description .....	212
13.4	I/O Signals .....	213
13.4.1	CGMXCLK .....	213
13.4.2	STOP Instruction .....	213
13.4.3	COPCTL Write .....	214
13.4.4	Power-On Reset .....	214

13.4.5	Internal Reset . . . . .	214
13.4.6	Reset Vector Fetch . . . . .	214
13.4.7	COPD (COP Disable) . . . . .	214
13.4.8	COPRS (COP Rate Select) . . . . .	214
13.5	COP Control Register . . . . .	215
13.6	Interrupts . . . . .	215
13.7	Monitor Mode . . . . .	215
13.8	Low-Power Modes . . . . .	215
13.8.1	Wait Mode . . . . .	215
13.8.2	Stop Mode . . . . .	216
13.9	COP Module During Break Mode . . . . .	216

## Section 14. External Interrupt (IRQ)

14.1	Contents . . . . .	217
14.2	Introduction . . . . .	217
14.3	Features . . . . .	218
14.4	Functional Description . . . . .	218
14.5	$\overline{\text{IRQ1}}$ Pin . . . . .	221
14.6	IRQ Module During Break Interrupts . . . . .	222
14.7	IRQ Status and Control Register . . . . .	222

## Section 15. Keyboard Interrupt Module (KBI)

15.1	Contents . . . . .	225
15.2	Introduction . . . . .	225
15.3	Features . . . . .	226
15.4	Functional Description . . . . .	227
15.5	Keyboard Initialization . . . . .	228

15.6	Low-Power Modes	229
15.6.1	Wait Mode	229
15.6.2	Stop Mode	229
15.7	Keyboard Module During Break Interrupts	230
15.8	I/O Registers	230
15.8.1	Keyboard Status and Control Register	230
15.8.2	Keyboard Interrupt Enable Register	232

### Section 16. Break Module (BREAK)

16.1	Contents	233
16.2	Introduction	233
16.3	Features	234
16.4	Functional Description	234
16.4.1	Flag Protection During Break Interrupts	236
16.4.2	CPU During Break Interrupts	236
16.4.3	TIM During Break Interrupts	236
16.4.4	COP During Break Interrupts	236
16.5	Break Module Registers	237
16.5.1	Break Status and Control Register	237
16.5.2	Break Address Registers	238
16.6	Low-Power Modes	238
16.6.1	Wait Mode	238
16.6.2	Stop Mode	238

### Section 17. Electrical Specifications

17.1	Contents	239
17.2	Introduction	239
17.3	Absolute Maximum Ratings	240
17.4	Functional Operating Range	241
17.5	Thermal Characteristics	241

17.6	DC Electrical Characteristics	242
17.7	Control Timing	243
17.8	Oscillator Characteristics	243
17.9	USB DC Electrical Characteristics	244
17.10	USB Low-Speed Source Electrical Characteristics	245
17.11	USB Signaling Levels	246
17.12	Timer Interface Module Characteristics	247
17.13	Memory Characteristics	247

## **Section 18. Mechanical Specifications**

18.1	Contents	249
18.2	Introduction	249
18.3	Quad Flat Pack (Case 848B-04)	250
18.4	Shrink Dual In-Line Package (Case 858-01)	251

## **Section 19. Ordering Information**

19.1	Contents	253
19.2	Introduction	253
19.3	MC Order Numbers	253





## List of Figures

Figure	Title	Page
1-1	MCU Block Diagram . . . . .	27
1-2	52-Pin QFP Assignments (Top View). . . . .	28
1-3	42-Pin SDIP Pin Assignments . . . . .	29
1-4	Power Supply Bypassing . . . . .	30
1-5	Regulator Supply Capacitor Configuration . . . . .	31
2-1	Memory Map . . . . .	36
2-2	Control, Status, and Data Registers. . . . .	38
5-1	Mask Option Register (MOR) . . . . .	54
6-1	CPU Registers . . . . .	59
6-2	Accumulator (A) . . . . .	59
6-3	Index Register (H:X) . . . . .	60
6-4	Stack Pointer (SP) . . . . .	61
6-5	Program Counter (PC) . . . . .	62
6-6	Condition Code Register (CCR) . . . . .	63
7-1	Oscillator External Connections . . . . .	74
8-1	SIM Block Diagram. . . . .	79
8-2	SIM Register Summary . . . . .	80
8-3	SIM Clock Signals . . . . .	81
8-4	External Reset Timing . . . . .	83
8-5	Internal Reset Timing . . . . .	84
8-6	Sources of Internal Reset. . . . .	84
8-7	POR Recovery . . . . .	85
8-8	Interrupt Processing . . . . .	89
8-9	Interrupt Entry. . . . .	90

Figure	Title	Page
8-10	Interrupt Recovery . . . . .	90
8-11	Interrupt Recognition Example . . . . .	91
8-12	Interrupt Status Register 1 (INT1) . . . . .	93
8-13	Interrupt Status Register 2 (INT2) . . . . .	94
8-14	Wait Mode Entry Timing . . . . .	96
8-15	Wait Recovery from Interrupt or Break . . . . .	97
8-16	Wait Recovery from Internal Reset . . . . .	97
8-17	Stop Mode Entry Timing . . . . .	98
8-18	Stop Mode Recovery from Interrupt or Break . . . . .	98
8-19	Break Status Register (BSR) . . . . .	99
8-20	Reset Status Register (RSR) . . . . .	101
8-21	Break Flag Control Register (BFCR) . . . . .	102
9-1	USB Block Diagram . . . . .	106
9-2	Supported Transaction Types Per Endpoint . . . . .	107
9-3	Supported USB Packet Types . . . . .	108
9-4	Sync Pattern . . . . .	109
9-5	SOP, Sync Signaling, and Voltage Levels . . . . .	109
9-6	CRC Block Diagram for Token Packets . . . . .	111
9-7	CRC Block Diagram for Data Packets . . . . .	112
9-8	EOP Transaction Voltage Levels . . . . .	113
9-9	EOP Width Timing . . . . .	113
9-10	External Low-Speed Device Configuration . . . . .	117
9-11	Regulator Electrical Connections . . . . .	118
9-12	Regulator Bypass Option . . . . .	119
9-13	Receiver Characteristics . . . . .	121
9-14	Differential Input Sensitivity Over Entire Common Mode Range . . . . .	121
9-15	Data Jitter . . . . .	122
9-16	Data Signal Rise and Fall Time . . . . .	123
9-17	NRZI Data Encoding . . . . .	125
9-18	Flow Diagram for NRZI . . . . .	125
9-19	Bit Stuffing . . . . .	126
9-20	Flow Diagram for Bit Stuffing . . . . .	127
9-21	Register Summary . . . . .	128
9-22	USB Address Register (UADDR) . . . . .	131

Figure	Title	Page
9-23	USB Interrupt Register 0 (UIR0).....	132
9-24	USB Interrupt Register 1 (UIR1).....	134
9-25	USB Control Register 0 (UCR0).....	136
9-26	USB Control Register 1 (UCR1).....	138
9-27	USB Control Register 2 (UCR2).....	140
9-28	USB Status Register (USR) .....	142
9-29	USB Endpoint 0 Data Register (UE0D0–UE0D7) .....	143
9-30	USB Endpoint 1/Endpoint 2 Data Register (UE1D0–UE1D7).....	144
9-31	OUT Token Data Flow for Receive Endpoint 0 .....	146
9-32	SETUP Token Data Flow for Receive Endpoint 0 .....	147
9-33	IN Token Data Flow for Transmit Endpoint 0 .....	148
9-34	IN Token Data Flow for Transmit Endpoint 1/Endpoint 2.....	149
10-1	Monitor Mode Circuit .....	153
10-2	Monitor Data Format .....	156
10-3	Break Transaction .....	156
10-4	Read Transaction.....	158
10-5	Write Transaction.....	158
10-6	Stack Pointer at Monitor Mode Entry .....	162
10-7	Monitor Mode Entry Timing .....	163
11-1	TIM Block Diagram.....	167
11-2	TIM I/O Register Summary.....	168
11-3	PWM Period and Pulse Width .....	172
11-4	TIM Status and Control Register (TSC) .....	179
11-5	TIM Counter Registers (TCNTH:TCNTL).....	181
11-6	TIM Counter Modulo Registers (TMODH:TMODL) .....	182
11-7	TIM Channel Status and Control Registers (TSC0:TSC1).....	183
11-8	CHxMAX Latency.....	186
11-9	TIM Channel Registers (TCH0H/L:TCH1H/L) .....	187
12-1	I/O Port Register Summary .....	190
12-2	Port A Data Register (PTA) .....	192

## List of Figures

Figure	Title	Page
12-3	Data Direction Register A (DDRA) . . . . .	193
12-4	Port A I/O Circuit . . . . .	193
12-5	Port B Data Register (PTB) . . . . .	195
12-6	Data Direction Register B (DDRB) . . . . .	196
12-7	Port B I/O Circuit . . . . .	196
12-8	Port C Data Register (PTC) . . . . .	198
12-9	Data Direction Register C (DDRC) . . . . .	199
12-10	Port C I/O Circuit . . . . .	199
12-11	Port D Data Register (PTD) . . . . .	201
12-12	Data Direction Register D (DDRD) . . . . .	202
12-13	Port D I/O Circuit . . . . .	202
12-14	Port E Data Register (PTE) . . . . .	204
12-15	Data Direction Register E (DDRE) . . . . .	206
12-16	Port E I/O Circuit . . . . .	206
12-17	Port Option Control Register (POC) . . . . .	208
13-1	COP Block Diagram . . . . .	212
13-2	COP I/O Register Summary . . . . .	212
13-3	COP Control Register (COPCTL) . . . . .	215
14-1	IRQ Module Block Diagram . . . . .	220
14-2	IRQ I/O Register Summary . . . . .	220
14-3	IRQ Status and Control Register (ISCR) . . . . .	223
15-1	Keyboard Module Block Diagram . . . . .	226
15-2	I/O Register Summary . . . . .	226
15-3	Keyboard Status and Control Register (KBSCR) . . . . .	231
15-4	Keyboard Interrupt Enable Register (KBIER) . . . . .	232
16-1	Break Module Block Diagram . . . . .	235
16-2	Break I/O Register Summary . . . . .	235
16-3	Break Status and Control Register (BRKSCR) . . . . .	237
16-4	Break Address Registers (BRKH and BRKL) . . . . .	238

## List of Tables

Table	Title	Page
2-1	Vector Addresses .....	47
6-1	Instruction Set Summary .....	65
6-2	Opcode Map .....	72
8-1	Signal Name Conventions .....	81
8-2	PIN Bit Set Timing .....	83
8-3	Interrupt Sources .....	92
9-1	Supported Packet Identifiers .....	110
10-1	Monitor Mode Entry .....	154
10-2	Mode Differences .....	155
10-3	Monitor Baud Rate Selection .....	157
10-4	READ (Read Memory) Command .....	159
10-5	WRITE (Write Memory) Command .....	159
10-6	IREAD (Indexed Read) Command .....	160
10-7	IWRITE (Indexed Write) Command .....	160
10-8	READSP (Read Stack Pointer) Command .....	161
10-9	RUN (Run User Program) Command .....	161
11-1	Prescaler Selection .....	180
11-2	Mode, Edge, and Level Selection .....	185
12-1	Port A Pin Functions .....	194
12-2	Port B Pin Functions .....	197
12-3	Port C Pin Functions .....	200
12-4	Port D Pin Functions .....	203
12-5	Port E Pin Functions .....	207
19-1	MC Order Numbers .....	253



## Section 1. General Description

### 1.1 Contents

1.2	Introduction .....	24
1.3	Features .....	24
1.4	MCU Block Diagram .....	26
1.5	Pin Assignments .....	28
1.5.1	Power Supply Pins ( $V_{DDREG}$ , $V_{SSREG}$ , $V_{DD1}$ , $V_{SS1}$ , $V_{DD2}$ , and $V_{SS2}$ ) .....	30
1.5.2	Voltage Regulator Out (REGOUT) .....	31
1.5.3	Oscillator Pins (OSC1 and OSC2) .....	31
1.5.4	External Reset Pin ( $\overline{RST}$ ) .....	31
1.5.5	External Interrupt Pin ( $\overline{IRQ1}$ ) .....	32
1.5.6	USB Data Pins (D+ and D-) .....	32
1.5.7	Port A Input/Output (I/O) Pins (PTA7–PTA0) .....	32
1.5.8	Port B (I/O) Pins (PTB7–PTB0) .....	32
1.5.9	Port C I/O Pins (PTC7–PTC0) .....	32
1.5.10	Port D I/O Pins (PTD7/ $\overline{KBD7}$ –PTD0/ $\overline{KBD0}$ ) .....	33
1.5.11	Port E I/O Pins (PTE6–PTE3 and PTE2/TCH1, PTE1/TCH0, PTE0/TCLK) .....	33

### 1.2 Introduction

The MC68HC08KL8 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

### 1.3 Features

Features of the MC68HC08KL8 include:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 1.5-MHz Internal Bus Operation
- Eight Kbytes of On-Chip Read-Only Memory (ROM)
- On-Chip Programming Firmware for Use with Host Personal Computer
- ROM Data Security<sup>1</sup>
- 368 Bytes of On-Chip Random Access Memory (RAM)
- 39 General-Purpose Input/Output (I/O) Ports, 24 with Software Configurable Pullups
- 16-Bit, 2-Channel Timer Interface Module (TIM)
- 8-Bit Keyboard Interrupt (KBI) Port
- Available in a 52-lead plastic quad flat pack (QFP) package:
  - 39 general-purpose input/output (I/O), 24 with software configurable pullups
  - 16-bit, 2-channel timer interface module (TIM)
  - Eight light-emitting diode (LED) direct drive port pins

---

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the EPROM/OTPROM difficult for unauthorized users.



- Available in a 42-lead shrink dual in-line plastic (SDIP) package:
  - 29 general-purpose input/output (I/O), 21 with software configurable pullups
  - Five light-emitting diode (LED) direct drive port pins
- Full Compatibility with *Universal Serial Bus (USB) Specification* Rev. 1.0
  - Supports Non-Isochronous Data
  - Bidirectional Half-Duplex Link
  - 1.5 Mbps Data Rate (Low Speed)
- On-Chip USB Transceiver
- On-Chip 3.3-V Regulator for USB Transceiver
- USB Data Control Logic
  - Packet Decoding/Generation
  - CRC Generation and Checking
  - Non-Return-to-Zero Inverted (NRZI) Encoding/Decoding and Bit-Stuffing
- Two 8-Byte Transmit Buffers
  - One Dedicated for Endpoint 0
  - One Shared by Endpoint 1 and Endpoint 2
- One 8-Byte Receive Buffer
  - Dedicated for Control Endpoint 0
- USB Suspend/Resume Operation
- System Protection Features
  - Optional Computer Operating Properly (COP) Reset
  - Illegal Opcode Detection with Optional Reset
  - Illegal Address Detection with Optional Reset
- Low-Power Design, Fully Static with Stop Mode and Wait Mode
- Master Reset Pin with Internal Pullup and Power-On Reset
- External Asynchronous Interrupt Pin with Internal Pullup ( $\overline{\text{IRQ1}}$ )

Features of the CPU08 include:

- Enhanced HC05 Programming Model
- Extensive Loop Control Functions
- 16 Addressing Modes (Eight More Than the HC05)
- 16-Bit Index Register and Stack Pointer
- Memory-to-Memory Data Transfers
- Fast  $8 \times 8$  Multiply Instruction
- Fast 16/8 Divide Instruction
- Binary-Coded Decimal (BCD) Instructions
- Optimization for Controller Applications
- Third Party C Language Support

### 1.4 MCU Block Diagram

**Figure 1-1** shows the structure of the MC68HC08KL8.

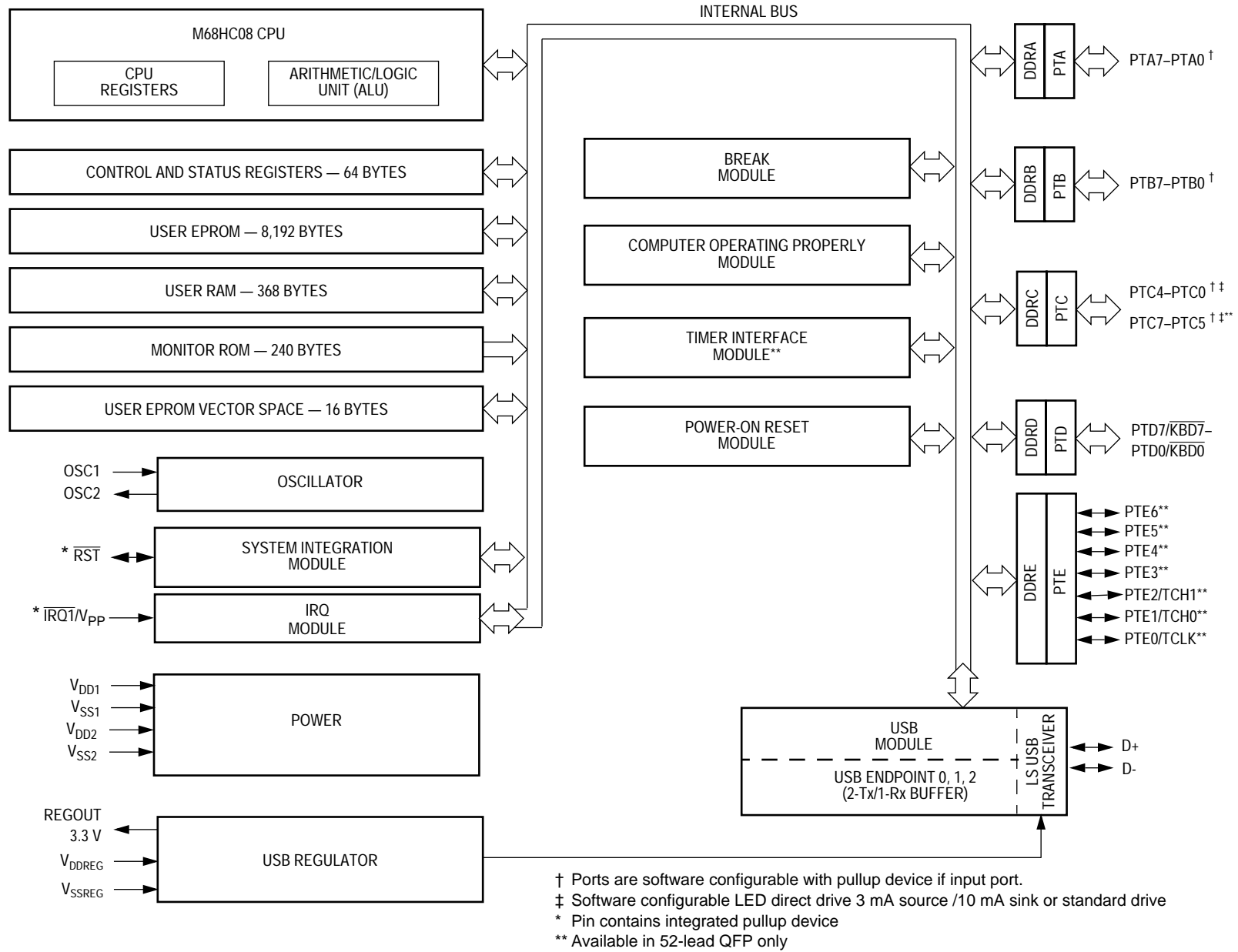
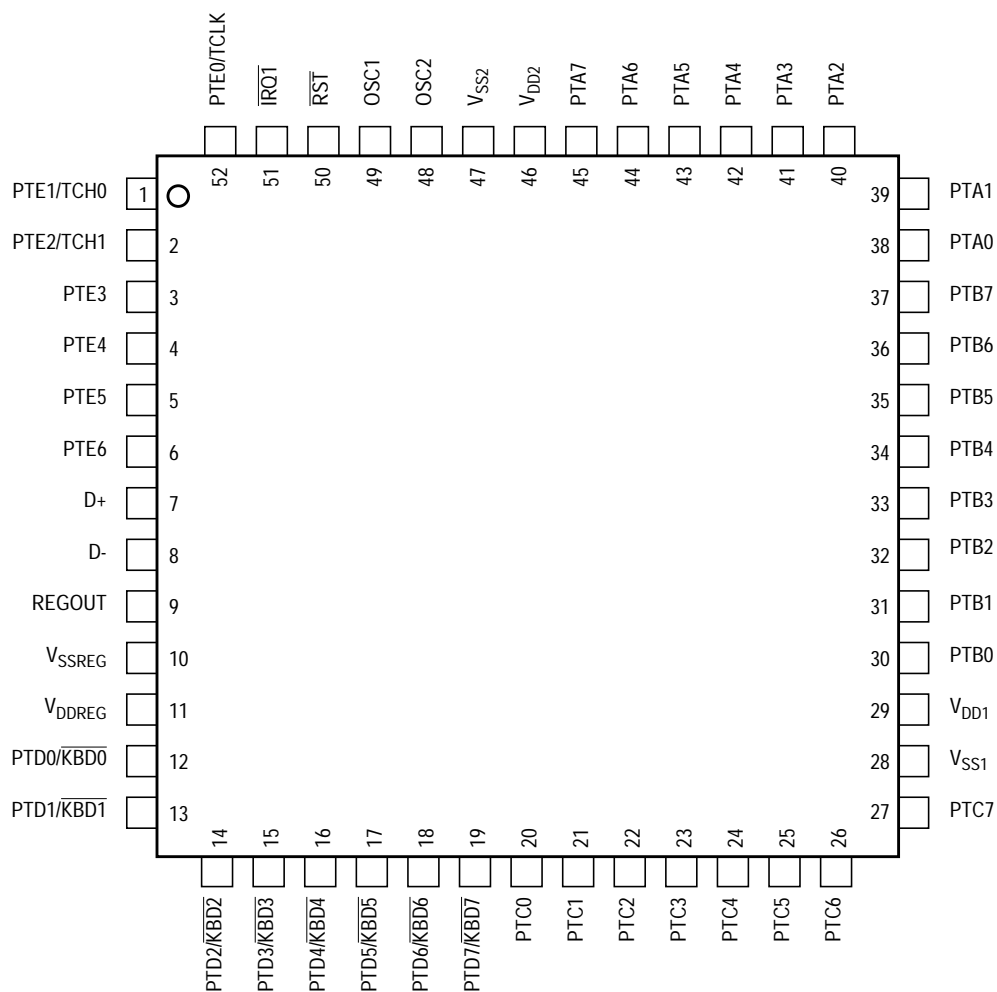


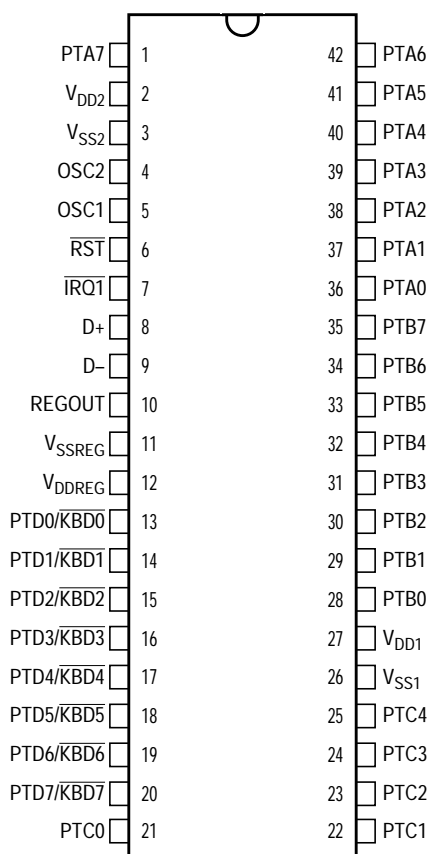
Figure 1-1. MCU Block Diagram

## 1.5 Pin Assignments

**Figure 1-2** shows the 52-pin QFP assignments and **Figure 1-3** shows the 42-pin SDIP assignments.



**Figure 1-2. 52-Pin QFP Assignments (Top View)**



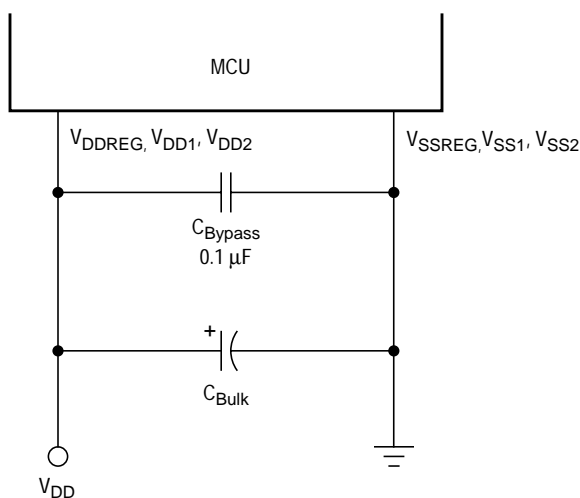
**Figure 1-3. 42-Pin SDIP Pin Assignments**

### 1.5.1 Power Supply Pins ( $V_{DDREG}$ , $V_{SSREG}$ , $V_{DD1}$ , $V_{SS1}$ , $V_{DD2}$ , and $V_{SS2}$ )

$V_{DDREG}$  and  $V_{SSREG}$  are the power supply and ground pins used by the on-board regulator. In regulator bypass, these become the supply pins for the USB transceiver.

$V_{DD1}$ ,  $V_{SS1}$ ,  $V_{DD2}$ , and  $V_{SS2}$  are the power supply and ground pins. The MCU operates from a single power supply.

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as [Figure 1-4](#) shows. Place the bypass capacitors as close to the MCU power pins as possible. Use high-frequency-response ceramic capacitors for  $C_{Bypass}$ .  $C_{Bulk}$  are optional bulk current bypass capacitors for use in applications that require the port pins to source high current levels.

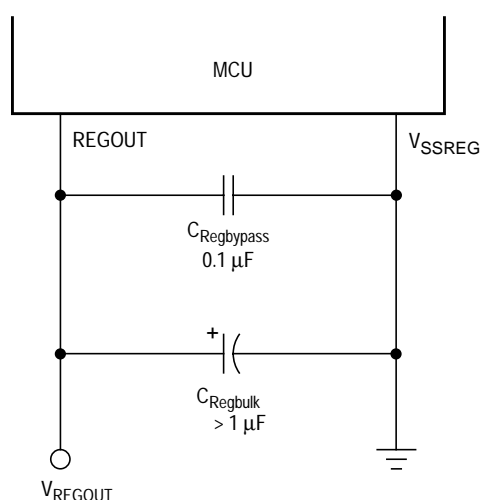


NOTE: Component values shown represent typical applications.

**Figure 1-4. Power Supply Bypassing**

## 1.5.2 Voltage Regulator Out (REGOUT)

REGOUT is the 3.3-V output of the on-chip voltage regulator. It is used to supply the voltage for the external pullup resistor required on the USB's D– line. REGOUT also is used internally for the USB data driver. The REGOUT pin requires an external bulk capacitor 1  $\mu$ F or larger and a 0.1- $\mu$ F ceramic bypass capacitor as **Figure 1-5** shows. Place the bypass capacitors as close to the REGOUT pin as possible. (See **Section 9. Universal Serial Bus Module (USB)**.)



**Figure 1-5. Regulator Supply Capacitor Configuration**

## 1.5.3 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. (See **Section 7. Oscillator**.)

## 1.5.4 External Reset Pin ( $\overline{\text{RST}}$ )

A logic 0 on the  $\overline{\text{RST}}$  pin forces the MCU to a known startup state.  $\overline{\text{RST}}$  is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. The  $\overline{\text{RST}}$  pin contains an internal pullup device. (See **Section 8. System Integration Module (SIM)**.)

### 1.5.5 External Interrupt Pin ( $\overline{\text{IRQ1}}$ )

$\overline{\text{IRQ1}}$  is an asynchronous external interrupt pin. The  $\overline{\text{IRQ1}}$  pin contains an internal pullup device. (See [Section 14. External Interrupt \(IRQ\)](#).)

### 1.5.6 USB Data Pins (D+ and D–)

D+ and D– are the differential data lines used by the USB module. (See [Section 9. Universal Serial Bus Module \(USB\)](#).)

### 1.5.7 Port A Input/Output (I/O) Pins (PTA7–PTA0)

PTA7–PTA0 are general-purpose bidirectional I/O port pins. (See [Section 12. Input/Output Ports \(I/O\)](#).) Each pin contains a software configurable pullup device when the pin is configured as an input. (See [12.8 Port Options](#).)

### 1.5.8 Port B (I/O) Pins (PTB7–PTB0)

PTB7–PTB0 are general-purpose bidirectional I/O port pins. (See [Section 12. Input/Output Ports \(I/O\)](#).) Each pin contains a software configurable pullup device when the pin is configured as an input. (See [12.8 Port Options](#).)

### 1.5.9 Port C I/O Pins (PTC7–PTC0)

PTC7–PTC0 are general-purpose bidirectional I/O port pins. (See [Section 12. Input/Output Ports \(I/O\)](#).) Port C pins are software configurable to be LED direct drive ports. Each pin contains a software configurable pullup device when the pin is configured as an input. (See [12.8 Port Options](#).)

**NOTE:** *On the 42-lead SDIP package the PTC7–PTC5 pads are not bonded out. Set these ports to output to avoid floating inputs.*



#### 1.5.10 Port D I/O Pins (PTD7/ $\overline{\text{KBD7}}$ –PTD0/ $\overline{\text{KBD0}}$ )

PTD7/ $\overline{\text{KBD7}}$ –PTD0/ $\overline{\text{KBD0}}$  are general-purpose bidirectional I/O port pins. (See [Section 12. Input/Output Ports \(I/O\)](#).) Any or all of the port D pins can be programmed to serve as external interrupt pins. (See [Section 15. Keyboard Interrupt Module \(KBI\)](#).)

#### 1.5.11 Port E I/O Pins (PTE6–PTE3 and PTE2/TCH1, PTE1/TCH0, PTE0/TCLK)

Port E is a 7-bit special function port that shares three of its pins with the timer interface module. ([Section 11. Timer Interface Module \(TIM\)](#) and [Section 12. Input/Output Ports \(I/O\)](#).)

**NOTE:** *On the 42-lead SDIP package the port E pads are not bonded out. Set these ports to output to avoid floating inputs.*



## Section 2. Memory Map

### 2.1 Contents

2.2	Introduction . . . . .	35
2.3	Input/Output (I/O) Section. . . . .	37
2.4	Unimplemented Memory Locations . . . . .	37
2.5	Reserved Memory Locations . . . . .	37
2.6	Monitor ROM . . . . .	47

### 2.2 Introduction

The CPU08 can address 64 Kbytes of memory space. The memory map, shown in **Figure 2-1**, includes:

- Eight Kbytes of read-only memory (ROM)
- 368 bytes of RAM
- 16 bytes of user-defined vectors
- 240 bytes of monitor ROM

# Memory Map

\$0000 ↓ \$003F \$0040 ↓ \$01AF \$01B0 ↓ \$DDFF \$DE00 ↓ \$FDFF	I/O REGISTERS, 64 BYTES
	RAM, 368 BYTES
	UNIMPLEMENTED 56,400 BYTES
	ROM, 8,192 BYTES
\$FE00	BREAK STATUS REGISTER (BSR)
\$FE01	RESET STATUS REGISTER (RSR)
\$FE02	RESERVED
\$FE03	BREAK FLAG CONTROL REGISTER (BFCR)
\$FE04	INTERRUPT STATUS REGISTER 1 (INT1)
\$FE05	INTERRUPT STATUS REGISTER 2 (INT2)
\$FE06	RESERVED
\$FE07 ↓ \$FE0B \$FE0C \$FE0D \$FE0E \$FE0F \$FE10 ↓ \$FEFF \$FF00 ↓ \$FFEF \$FFF0 ↓ \$FFFF	RESERVED, 4 BYTES
	BREAK ADDRESS HIGH REGISTER (BRKH)
	BREAK ADDRESS LOW REGISTER (BRKL)
	BREAK STATUS AND CONTROL REGISTER (BSCR)
	RESERVED
	MONITOR ROM, 240 BYTES
	UNIMPLEMENTED, 240 BYTES
	VECTORS, 16 BYTES

**Figure 2-1. Memory Map**

## 2.3 Input/Output (I/O) Section

Addresses \$0000–\$003F contain most of the control, status, and data registers. Additional I/O registers have these addresses:

- \$FE00 (break status register, BSR)
- \$FE01 (reset status register, RSR)
- \$FE03 (break flag control register, BFCR)
- \$FE04 (interrupt status register 1, INT1)
- \$FE05 (interrupt status register 2, INT2)
- \$FE0C and \$FE0D (break address registers, BRKH and BRKL)
- \$FE0E (break status and control register, BSCR)
- \$FFFF (COP control register, COPCTL)

## 2.4 Unimplemented Memory Locations

Some addresses are unimplemented. Accessing an unimplemented address can cause an illegal address reset if illegal address resets are enabled. In the memory map and in the I/O register summary, unimplemented addresses are shaded.


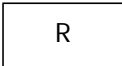
Some I/O bits are read-only: the write function is unimplemented. Writing to a read-only I/O bit has no effect on MCU operation. In register figures, the write function of read-only bits is shaded.

## 2.5 Reserved Memory Locations

Some addresses are reserved. Writing to a reserved address can have unpredictable effects on MCU operation. In the memory map and in the I/O register summary, reserved addresses are marked with the word Reserved.

Some I/O bits are reserved. Writing to a reserved bit can have unpredictable effects on MCU operation. In register figures, reserved bits are marked with the letter R.

# Memory Map

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0000	Port A Data Register (PTA)	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by Reset							
\$0001	Port B Data Register (PTB)	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by Reset							
\$0002	Port C Data Register (PTC)	Read:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by Reset							
\$0003	Port D Data Register (PTD)	Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:								
		Reset:	Unaffected by Reset							
\$0004	Data Direction Register A (DDRA)	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0005	Data Direction Register B (DDRB)	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0006	Data Direction Register C (DDRC)	Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0007	Data Direction Register D (DDRD)	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
				= Unimplemented			= Reserved		X = Indeterminate	


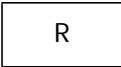
**Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 9)**

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0008	Port E Data Register (PTE)	Read:	0	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by Reset							
\$0009	Unimplemented	Read:								
		Write:								
\$000A	Unimplemented	Read:								
		Write:								
\$000B	Unimplemented	Read:								
		Write:								
\$000C	Data Direction Register E (DDRE)	Read:	0	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$000D	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
\$000E	Keyboard Interrupt Enable Register (KBIER)	Read:								
		Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Reset:	0	0	0	0	0	0	0	0
\$000F	Port Option Control Register (POC)	Read:	0	0	LDD	0	0	PCP	PBP	PAP
		Write:								
		Reset:	0	0	1	0	0	0	0	0
\$0010	TIM Status and Control Register (TSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
\$0011	Unimplemented	Read:								
		Write:								
				= Unimplemented		R	= Reserved		X = Indeterminate	

**Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 9)**

# Memory Map

Addr.	Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0012	TIM Counter Register High (TCNTH)	Read: Bit 15	14	13	12	11	10	9	Bit 8
		Write:							
		Reset:	0	0	0	0	0	0	0
\$0013	TIM Counter Register Low (TCNTL)	Read: Bit 7	6	5	4	3	2	1	Bit 0
		Write:							
		Reset:	0	0	0	0	0	0	0
\$0014	TIM Counter Modulo Register High (TMODH)	Read: Bit 15	14	13	12	11	10	9	Bit 8
		Write:							
		Reset:	1	1	1	1	1	1	1
\$0015	TIM Counter Modulo Register Low (TMDL)	Read: Bit 7	6	5	4	3	2	1	Bit 0
		Write:							
		Reset:	1	1	1	1	1	1	1
\$0016	TIM Channel 0 Status and Control Register (TSC0)	Read: CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write: 0							
		Reset:	0	0	0	0	0	0	0
\$0017	TIM Channel 0 Register High (TCH0H)	Read: Bit 15	14	13	12	11	10	9	Bit 8
		Write:							
		Reset:	Indeterminate after Reset						
\$0018	TIM Channel 0 Register Low (TCH0L)	Read: Bit 7	6	5	4	3	2	1	Bit 0
		Write:							
		Reset:	Indeterminate after Reset						
\$0019	TIM Channel 1 Status and Control Register (TSC1)	Read: CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write: 0							
		Reset:	0	0	0	0	0	0	0
\$001A	TIM Channel 1 Register High (TCH1H)	Read: Bit 15	14	13	12	11	10	9	Bit 8
		Write:							
		Reset:	Indeterminate after Reset						

 = Unimplemented
  = Reserved
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 9)**



Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
\$001B	TIM Channel 1 Register Low (TCH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							
\$001C	Unimplemented	Read:								
		Write:								
\$001D	Unimplemented	Read:								
		Write:								
\$001E	IRQ Status and Control Register (ISCR)	Read:	0	0	0	0	IRQF1	0	IMASK1	MODE1
		Write:						ACK1		
		Reset:	0	0	0	0	0	0	0	0
\$001F	Mask Option Register (MOR)	Read:	R	REGBP	R	ROMSEC	SSREC	COPRS	STOP	COPD
		Write:								
		Reset:	Unaffected by Reset							
\$0020	USB Endpoint 0 Data Register 0 (UE0D0)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0021	USB Endpoint 0 Data Register 1 (UE0D1)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0022	USB Endpoint 0 Data Register 2 (UE0D2)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0023	USB Endpoint 0 Data Register 3 (UE0D3)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
				= Unimplemented		R	= Reserved		X = Indeterminate	

**Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 9)**

# Memory Map

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0	
\$0024	USB Endpoint 0 Data Register 4 (UE0D4)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0	
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0	
		Reset:	Indeterminate after Reset								
\$0025	USB Endpoint 0 Data Register 5 (UE0D5)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0	
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0	
		Reset:	Indeterminate after Reset								
\$0026	USB Endpoint 0 Data Register 6 (UE0D6)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0	
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0	
		Reset:	Indeterminate after Reset								
\$0027	USB Endpoint 0 Data Register 7 (UE0D7)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0	
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0	
		Reset:	Indeterminate after Reset								
\$0028	USB Endpoint 1/2 Data Register 0 (UE1D0)	Read:									
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0	
		Reset:	Indeterminate after Reset								
\$0029	USB Endpoint 1/2 Data Register 1 (UE1D1)	Read:									
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0	
		Reset:	Indeterminate after Reset								
\$002A	USB Endpoint 1/2 Data Register 2 (UE1D2)	Read:									
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0	
		Reset:	Indeterminate after Reset								
\$002B	USB Endpoint 1/2 Data Register 3 (UE1D3)	Read:									
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0	
		Reset:	Indeterminate after Reset								
\$002C	USB Endpoint 1/2 Data Register 4 (UE1D4)	Read:									
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0	
		Reset:	Indeterminate after Reset								
				= Unimplemented			R	= Reserved			X = Indeterminate


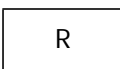
**Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 9)**

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
\$002D	USB Endpoint 1/2 Data Register 5 (UE1D5)	Read:								
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset:	Indeterminate after Reset							
\$002E	USB Endpoint 1/2 Data Register 6 (UE1D6)	Read:								
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset:	Indeterminate after Reset							
\$002F	USB Endpoint 1/2 Data Register 7 (UE1D7)	Read:								
		Write:	UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset:	Indeterminate after Reset							
\$0030	Unimplemented	Read:								
		Write:								
↓	Unimplemented	Read:								
		Write:								
\$0036	Unimplemented	Read:								
		Write:								
\$0037	USB Control Register 2 (UCR2)	Read:	0	0	TX1ST	0	ENABLE2	ENABLE1	STALL2	STALL1
		Write:	RSTFR	TX1STR						
		Reset:	0	0	0	0	0	0	0	0
\$0038	USB Address Register (UADDR)	Read:	USBEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0039	USB Interrupt Register 0 (UIR0)	Read:	TXD0F	RXD0F	RSTF	SUSPND	TXD0IE	RXD0IE	0	0
		Write:							TXD0FR	RXD0FR
		Reset:	0	0	0	0	0	0	0	0
\$003A	USB Interrupt Register 1 (UIR1)	Read:	TXD1F	EOPF	RESUMF	0	TXD1IE	EOPIE	0	0
		Write:				RESUMFR			TXD1FR	EOPFR
		Reset:	0	0	0	0	0	0	0	0
				= Unimplemented		R	= Reserved		X = Indeterminate	

**Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 9)**

# Memory Map

Addr.	Name	Bit 7	6	5	4	3	2	1	Bit 0
\$003B	USB Control Register 0 (UCR0)	Read:	T0SEQ	STALL0	TX0E	RX0E	TP0SIZ3	TP0SIZ2	TP0SIZ1
		Write:							
		Reset:	0	0	0	0	0	0	0
\$003C	USB Control Register 1 (UCR1)	Read:	T1SEQ	ENDADD	TX1E	FRESUM	TP1SIZ3	TP1SIZ2	TP1SIZ1
		Write:							
		Reset:	0	0	0	0	0	0	0
\$003D	USB Status Register (USR)	Read:	RSEQ	SETUP			RPSIZ3	RPSIZ2	RPSIZ1
		Write:							
		Reset:	X	X			X	X	X
\$003E	Unimplemented	Read:							
		Write:							
\$003F	Unimplemented	Read:							
		Write:							
\$FE00	Break Status Register (BSR)	Read:	R	R	R	R	R	SBSW	R
		Write:							
		Reset:						0	
\$FE01	Reset Status Register (RSR)	Read:	POR	PIN	COP	ILOP	ILAD	USB	0
		Write:							
		POR:	1	0	0	0	0	0	0
\$FE02	Reserved	Read:	R	R	R	R	R	R	R
		Write:							
\$FE03	Break Flag Control Register (BFCR)	Read:	BCFE	R	R	R	R	R	R
		Write:							
		Reset:	0						

 = Unimplemented
  = Reserved
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 9)**

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0	
\$FE04	Interrupt Status Register 1 (INT1)	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0	
		Write:	R	R	R	R	R	R	R	R	
		Reset:	0	0	0	0	0	0	0	0	
\$FE05	Interrupt Status Register 2 (INT2)	Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7	
		Write:	R	R	R	R	R	R	R	R	
		Reset:	0	0	0	0	0	0	0	0	
\$FE06	Reserved	Read:									
		Write:	R	R	R	R	R	R	R	R	
\$FE07	Unimplemented	Read:									
		Write:									
\$FE08	Unimplemented	Read:									
		Write:									
\$FE09	Unimplemented	Read:									
		Write:									
\$FE0A	Unimplemented	Read:									
		Write:									
\$FE0B	Unimplemented	Read:									
		Write:									
\$FE0C	Break Address Register High (BRKH)	Read:									
		Write:	Bit 15	14	13	12	11	10	9	Bit 8	
		Reset:	0	0	0	0	0	0	0	0	
\$FE0D	Break Address Register Low (BRKL)	Read:									
		Write:	Bit 7	6	5	4	3	2	1	Bit 0	
		Reset:	0	0	0	0	0	0	0	0	
				= Unimplemented			R	= Reserved			X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 9)**

# Memory Map

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
\$FE0E	Break Status and Control Register (BRKSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
		Reset:	0	0	0	0	0	0	0	0
\$FFFF	COP Control Register (COPCTL)	Read:	Low Byte of Reset Vector							
		Write:	Writing Clears COP Counter (Any Value)							
		Reset:	Unaffected by Reset							
				= Unimplemented		R	= Reserved		X = Indeterminate	

**Figure 2-2. Control, Status, and Data Registers (Sheet 9 of 9)**

**Table 2-1** is a list of vector locations.

**Table 2-1. Vector Addresses**

	Address	Vector
Low ↑ Priority ↓ High	\$FFF0	Keyboard Vector (High)
	\$FFF1	Keyboard Vector (Low)
	\$FFF2	TIM Overflow Vector (High)
	\$FFF3	TIM Overflow Vector (Low)
	\$FFF4	TIM Channel 1 Vector (High)
	\$FFF5	TIM Channel 1 Vector (Low)
	\$FFF6	TIM Channel 0 Vector (High)
	\$FFF7	TIM Channel 0 Vector (Low)
	\$FFF8	USB Vector (High)
	\$FFF9	USB Vector (Low)
	\$FFFA	IRQ1 Vector (High)
	\$FFFFB	IRQ1 Vector (Low)
	\$FFFC	SWI Vector (High)
	\$FFFD	SWI Vector (Low)
	\$FFFE	Reset Vector (High)
	\$FFFF	Reset Vector (Low)

## 2.6 Monitor ROM

The 240 bytes at addresses \$FE10–\$FEFF are reserved ROM addresses that contain the instructions for the monitor functions. (See [Section 10. Monitor ROM \(MON\)](#).)





## Section 3. Random-Access Memory (RAM)

### 3.1 Contents

3.2	Introduction . . . . .	49
3.3	Functional Description . . . . .	49

### 3.2 Introduction

This section describes the 368 bytes of random-access memory (RAM).

### 3.3 Functional Description

Addresses \$0040–\$01AF are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

**NOTE:** *For correct operation, the stack pointer must point only to RAM locations.*

Within page zero are 192 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at \$00FF, direct addressing mode instructions can access efficiently all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers.

**NOTE:** *For M6805 Family compatibility, the H register is not stacked.*

## Random-Access Memory (RAM)

During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

**NOTE:** *Be careful when using nested subroutines. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.*

## Section 4. Read-Only Memory (ROM)

### 4.1 Contents

4.2	Introduction . . . . .	51
4.3	Functional Description . . . . .	51

### 4.2 Introduction

This section describes the 8 KBytes of read-only memory (ROM) and 16 bytes of user vectors.

### 4.3 Functional Description

An unprogrammed or erased location reads as \$00. These addresses are user ROM locations:

- \$DE00–\$FDFF
- \$FFF0–\$FFFF (These locations are reserved for user-defined interrupt and reset vectors.)

**NOTE:** *A security feature prevents viewing of the ROM contents.<sup>1</sup>*

---

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the ROM contents difficult for unauthorized users.



## Section 5. Mask Option Register (MOR)

### 5.1 Contents

5.2	Introduction .....	53
5.3	Functional Description .....	54

### 5.2 Introduction

This section describes the mask options and the mask option register (MOR). The mask options are hard-wired connections specified at the same time as the ROM code, which allow the user to customize the MCU. The MOR controls the enable or disable of the following functions:

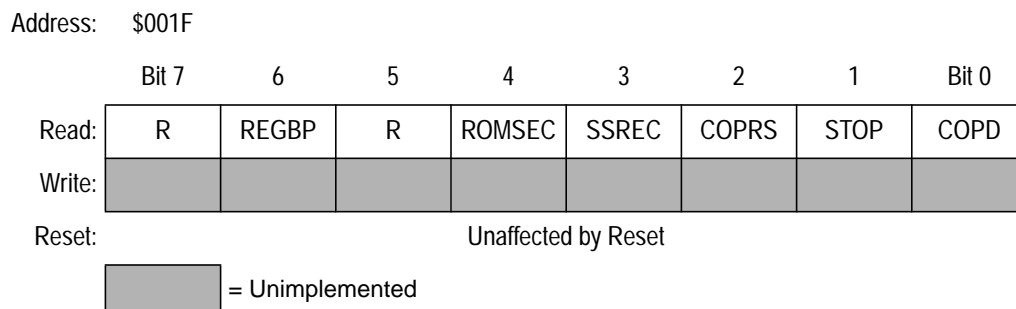
- USB regulator bypass
- ROM security<sup>1</sup>
- Stop mode recovery time (32 or 4096 CGMXCLK cycles)
- COP timeout period ( $2^{18} - 2^4$  or  $2^{13} - 2^4$  CGMXCLK cycles)
- STOP instruction
- Operation of the computer operating properly module (COP)

---

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the ROM difficult for unauthorized users.

## Mask Option Register (MOR)

### 5.3 Functional Description



**Figure 5-1. Mask Option Register (MOR)**

**NOTE:** *Writing to a reserved address can have unpredictable effects on MCU operation.*

#### REGBP — Regulator Bypass Bit

The REGBP causes the on-chip regulator to pass  $V_{DDREG}$  to the REGOUT pin.

1 =  $V_{DDREG}$  voltage applied to the REGOUT pin.

0 = Regulator output drives 3.3 volts on REGOUT pin.

**NOTE:** *The bypass mode is reserved for transceiver testing and should not be used in normal operation.*

#### ROMSEC — ROM Security Bit

ROMSEC enables the ROM security feature. Setting the ROMSEC bit prevents reading of the ROM contents. Access to the ROM is denied to unauthorized users of customer specified software.

1 = ROM security enabled

0 = ROM security disabled

#### SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay.

1 = Stop mode recovery after 32 CGMXCLK cycles

0 = Stop mode recovery after 4096 CGMXCLK cycles

**NOTE:** *Exiting stop mode by pulling reset will result in the long stop recovery. If using an external crystal oscillator, do not set the SSREC bit.*

**COPRS — COP Rate Select Bit**

COPRS selects the COP timeout period. Reset clears COPRS. (See [Section 13. Computer Operating Properly \(COP\)](#).)

1 = COP timeout period =  $2^{13} - 2^4$  CGMXCLK cycles

0 = COP timeout period =  $2^{18} - 2^4$  CGMXCLK cycles

**STOP — STOP Instruction Enable Bit**

STOP enables the STOP instruction.

1 = STOP instruction enabled

0 = STOP instruction treated as illegal opcode

**COPD — COP Disable Bit**

COPD disables the COP module. (See [Section 13. Computer Operating Properly \(COP\)](#).)

1 = COP module disabled

0 = COP module enabled





## Section 6. Central Processor Unit (CPU)

### 6.1 Contents

6.2	Introduction . . . . .	57
6.3	Features . . . . .	58
6.4	CPU Registers . . . . .	59
6.4.1	Accumulator . . . . .	59
6.4.2	Index Register . . . . .	60
6.4.3	Stack Pointer . . . . .	61
6.4.4	Program Counter . . . . .	62
6.4.5	Condition Code Register . . . . .	63
6.5	Arithmetic/Logic Unit (ALU) . . . . .	65
6.6	Instruction Set Summary . . . . .	65
6.7	Opcode Map . . . . .	71

### 6.2 Introduction

This section describes the central processor unit (CPU). The M68HC08 CPU is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The *CPU08 Reference Manual* (Motorola document number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

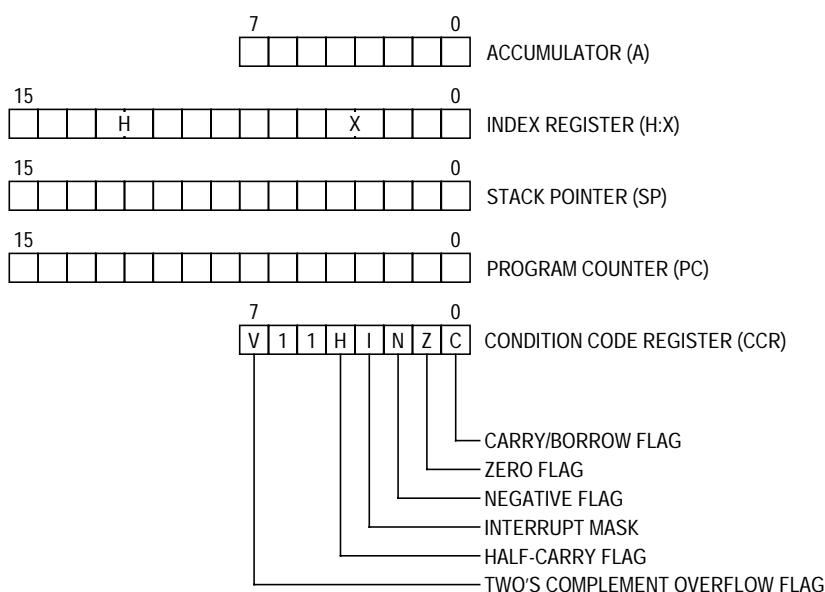
### 6.3 Features

Features of the CPU include:

- Fully Upward, Object-Code Compatibility with M68HC05 Family
- 16-Bit Stack Pointer with Stack Manipulation Instructions
- 16-Bit Index Register with X-Register Manipulation Instructions
- 8-MHz CPU Internal Bus Frequency
- 64-Kbyte Program/Data Memory Space
- 16 Addressing Modes
- Memory-to-Memory Data Moves Without Using Accumulator
- Fast 8-Bit by 8-Bit Multiply and 16-Bit by 8-Bit Divide Instructions
- Enhanced Binary-Coded Decimal (BCD) Data Handling
- Modular Architecture with Expandable Internal Bus Definition for Extension of Addressing Range Beyond 64 Kbytes
- Low-Power Stop Mode and Wait Mode

## 6.4 CPU Registers

**Figure 6-1** shows the five CPU registers. CPU registers are not part of the memory map.



**Figure 6-1. CPU Registers**

### 6.4.1 Accumulator

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic/logic operations.



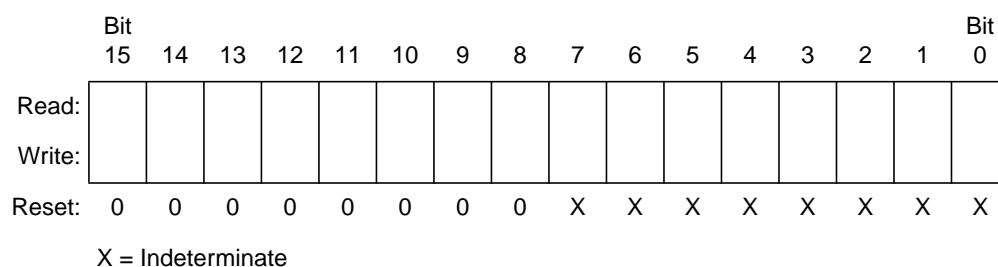
**Figure 6-2. Accumulator (A)**

## 6.4.2 Index Register

The 16-bit index register allows indexed addressing of a 64-Kbyte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

The index register can serve also as a temporary data storage location.



**Figure 6-3. Index Register (H:X)**

6.4.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to \$00FF. The reset stack pointer (RSP) instruction also sets the least significant byte (LSB) to \$FF but does not affect the most significant byte (MSB). The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.

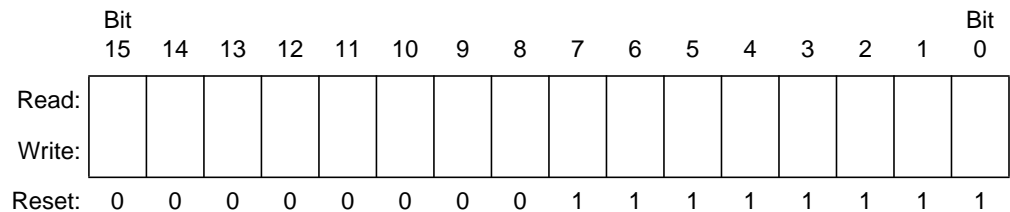


Figure 6-4. Stack Pointer (SP)

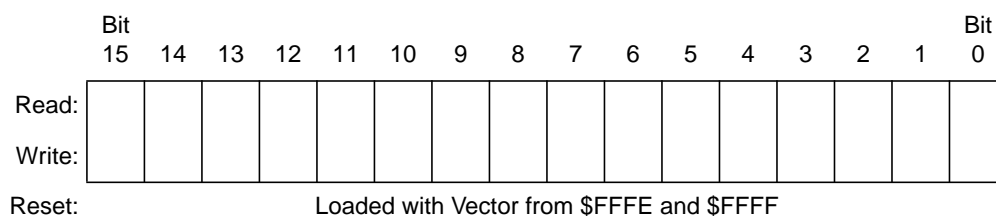
**NOTE:** *The location of the stack is arbitrary and may be relocated anywhere in RAM. Moving the SP out of page zero (\$0000 to \$00FF) frees direct address (page zero) space. For correct operation, the stack pointer must point only to RAM locations.*

## 6.4.4 Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

Normally, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

During reset, the program counter is loaded with the reset vector address located at \$FFFE and \$FFFF. The vector address is the address of the first instruction to be executed after exiting the reset state.



**Figure 6-5. Program Counter (PC)**

## 6.4.5 Condition Code Register

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to logic 1. The following paragraphs describe the functions of the condition code register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	V	1	1	H	I	N	Z	C
Write:								
Reset:	X	1	1	X	1	X	X	X

X = Indeterminate

**Figure 6-6. Condition Code Register (CCR)**

### V — Overflow Flag

The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.

1 = Overflow

0 = No overflow

### H — Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an ADD or ADC operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C flags to determine the appropriate correction factor.

1 = Carry between bits 3 and 4

0 = No carry between bits 3 and 4

### I — Interrupt Mask

When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the interrupt vector is fetched.

1 = Interrupts disabled

0 = Interrupts enabled

**NOTE:** *To maintain M6805 Family compatibility, the upper byte of the index register (H) is not stacked automatically. If the interrupt service routine modifies H, then the user must stack and unstack H using the PSHH and PULH instructions.*

After the I bit is cleared, the highest-priority interrupt request is serviced first.

A return-from-interrupt (RTI) instruction pulls the CPU registers from the stack and restores the interrupt mask from the stack. After any reset, the interrupt mask is set and can only be cleared by the clear interrupt (CLI) mask software instruction.

### N — Negative flag

The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result.

1 = Negative result

0 = Non-negative result

### Z — Zero flag

The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of \$00.

1 = Zero result

0 = Non-zero result

### C — Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.

1 = Carry out of bit 7

0 = No carry out of bit 7



## 6.5 Arithmetic/Logic Unit (ALU)

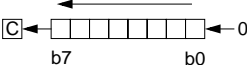
The ALU performs the arithmetic and logic operations defined by the instruction set.

Refer to the *CPU08 Reference Manual* (Motorola document number CPU08RM/AD) for a description of the instructions and addressing modes and more details about the CPU's architecture.

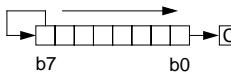
## 6.6 Instruction Set Summary

**Table 6-1** provides a summary of the M68HC08 instruction set.

**Table 6-1. Instruction Set Summary**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #opr ADC opr ADC opr ADC opr,X ADC opr,X ADC ,X ADC opr,SP ADC opr,SP	Add with Carry	$A \leftarrow (A) + (M) + (C)$	↑	↑	—	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
ADD #opr ADD opr ADD opr ADD opr,X ADD opr,X ADD ,X ADD opr,SP ADD opr,SP	Add without Carry	$A \leftarrow (A) + (M)$	↑	↑	—	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	AB BB CB DB EB FB 9EEB 9EDB	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
AIS #opr	Add Immediate Value (Signed) to SP	$SP \leftarrow (SP) + (16 \ll M)$	—	—	—	—	—	—	IMM	A7	ii	2
AIX #opr	Add Immediate Value (Signed) to H:X	$H:X \leftarrow (H:X) + (16 \ll M)$	—	—	—	—	—	—	IMM	AF	ii	2
AND #opr AND opr AND opr AND opr,X AND opr,X AND ,X AND opr,SP AND opr,SP	Logical AND	$A \leftarrow (A) \& (M)$	0	—	—	↑	↑	—	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL opr ASLA ASLX ASL opr,X ASL ,X ASL opr,SP	Arithmetic Shift Left (Same as LSL)		↑	—	—	↑	↑	↑	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff ff	4 1 1 4 3 5

**Table 6-1. Instruction Set Summary (Continued)**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ASR <i>opr</i> ASRA ASRX ASR <i>opr</i> ,X ASR <i>opr</i> ,X ASR <i>opr</i> ,SP	Arithmetic Shift Right		↑	—	—	—	↑	↑	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC <i>rel</i>	Branch if Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	—	—	—	—	—	—	REL	24	rr	3
BCLR <i>n</i> , <i>opr</i>	Clear Bit n in M	$M_n \leftarrow 0$	—	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS <i>rel</i>	Branch if Carry Bit Set (Same as BLO)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	—	—	—	—	—	—	REL	25	rr	3
BEQ <i>rel</i>	Branch if Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 1$	—	—	—	—	—	—	REL	27	rr	3
BGE <i>opr</i>	Branch if Greater Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 0$	—	—	—	—	—	—	REL	90	rr	3
BGT <i>opr</i>	Branch if Greater Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 0$	—	—	—	—	—	—	REL	92	rr	3
BHCC <i>rel</i>	Branch if Half Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? (H) = 0$	—	—	—	—	—	—	REL	28	rr	3
BHCS <i>rel</i>	Branch if Half Carry Bit Set	$PC \leftarrow (PC) + 2 + rel ? (H) = 1$	—	—	—	—	—	—	REL	29	rr	3
BHI <i>rel</i>	Branch if Higher	$PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 0$	—	—	—	—	—	—	REL	22	rr	3
BHS <i>rel</i>	Branch if Higher or Same (Same as BCC)	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	—	—	—	—	—	—	REL	24	rr	3
BIH <i>rel</i>	Branch if IRQ Pin High	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 1$	—	—	—	—	—	—	REL	2F	rr	3
BIL <i>rel</i>	Branch if IRQ Pin Low	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 0$	—	—	—	—	—	—	REL	2E	rr	3
BIT # <i>opr</i> BIT <i>opr</i> BIT <i>opr</i> BIT <i>opr</i> ,X BIT <i>opr</i> ,X BIT ,X BIT <i>opr</i> ,SP BIT <i>opr</i> ,SP	Bit Test	(A) & (M)	0	—	—	—	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE <i>opr</i>	Branch if Less Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 1$	—	—	—	—	—	—	REL	93	rr	3
BLO <i>rel</i>	Branch if Lower (Same as BCS)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	—	—	—	—	—	—	REL	25	rr	3
BLS <i>rel</i>	Branch if Lower or Same	$PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 1$	—	—	—	—	—	—	REL	23	rr	3
BLT <i>opr</i>	Branch if Less Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 1$	—	—	—	—	—	—	REL	91	rr	3
BMC <i>rel</i>	Branch if Interrupt Mask Clear	$PC \leftarrow (PC) + 2 + rel ? (I) = 0$	—	—	—	—	—	—	REL	2C	rr	3
BMI <i>rel</i>	Branch if Minus	$PC \leftarrow (PC) + 2 + rel ? (N) = 1$	—	—	—	—	—	—	REL	2B	rr	3
BMS <i>rel</i>	Branch if Interrupt Mask Set	$PC \leftarrow (PC) + 2 + rel ? (I) = 1$	—	—	—	—	—	—	REL	2D	rr	3
BNE <i>rel</i>	Branch if Not Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 0$	—	—	—	—	—	—	REL	26	rr	3
BPL <i>rel</i>	Branch if Plus	$PC \leftarrow (PC) + 2 + rel ? (N) = 0$	—	—	—	—	—	—	REL	2A	rr	3
BRA <i>rel</i>	Branch Always	$PC \leftarrow (PC) + 2 + rel$	—	—	—	—	—	—	REL	20	rr	3

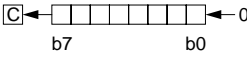
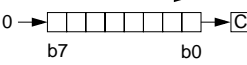
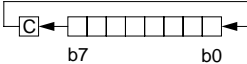
**Table 6-1. Instruction Set Summary (Continued)**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BRCLR <i>n,opr,rel</i>	Branch if Bit <i>n</i> in M Clear	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 0$	–	–	–	–	–	↓	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN <i>rel</i>	Branch Never	$PC \leftarrow (PC) + 2$	–	–	–	–	–	–	REL	21	rr	3
BRSET <i>n,opr,rel</i>	Branch if Bit <i>n</i> in M Set	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 1$	–	–	–	–	–	↓	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET <i>n,opr</i>	Set Bit <i>n</i> in M	$Mn \leftarrow 1$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR <i>rel</i>	Branch to Subroutine	$PC \leftarrow (PC) + 2$ ; push (PCL) $SP \leftarrow (SP) - 1$ ; push (PCH) $SP \leftarrow (SP) - 1$ $PC \leftarrow (PC) + rel$	–	–	–	–	–	–	REL	AD	rr	4
CBEQ <i>opr,rel</i> CBEQA # <i>opr,rel</i> CBEQX # <i>opr,rel</i> CBEQ <i>opr,X+,rel</i> CBEQ <i>X+,rel</i> CBEQ <i>opr,SP,rel</i>	Compare and Branch if Equal	$PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (X) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 2 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 4 + rel ? (A) - (M) = \$00$	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	$C \leftarrow 0$	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	$I \leftarrow 0$	–	–	0	–	–	–	INH	9A		2
CLR <i>opr</i> CLRA CLR X CLR H CLR <i>opr,X</i> CLR ,X CLR <i>opr,SP</i>	Clear	$M \leftarrow \$00$ $A \leftarrow \$00$ $X \leftarrow \$00$ $H \leftarrow \$00$ $M \leftarrow \$00$ $M \leftarrow \$00$ $M \leftarrow \$00$	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd   ff ff ff	3 1 1 1 3 2 4
CMP # <i>opr</i> CMP <i>opr</i> CMP <i>opr</i> CMP <i>opr,X</i> CMP <i>opr,X</i> CMP ,X CMP <i>opr,SP</i> CMP <i>opr,SP</i>	Compare A with M	$(A) - (M)$	↑	–	–	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
COM <i>opr</i> COMA COMX COM <i>opr,X</i> COM ,X COM <i>opr,SP</i>	Complement (One's Complement)	$M \leftarrow (\overline{M}) = \$FF - (M)$ $A \leftarrow (\overline{A}) = \$FF - (M)$ $X \leftarrow (\overline{X}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$	0	–	–	↑	↑	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd  ff ff ff	4 1 1 4 3 5

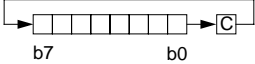
**Table 6-1. Instruction Set Summary (Continued)**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CPHX #opr CPHX opr	Compare H:X with M	$(H:X) - (M:M + 1)$	↑	—	—	↑	↑	↑	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #opr CPX opr CPX opr CPX ,X CPX opr,X CPX opr,X CPX opr,SP CPX opr,SP	Compare X with M	$(X) - (M)$	↑	—	—	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	—	—	↑	↑	↑	INH	72		2
DBNZ opr,rel DBNZA rel DBNZX rel DBNZ opr,X,rel DBNZ X,rel DBNZ opr,SP,rel	Decrement and Branch if Not Zero	$A \leftarrow (A) - 1$ or $M \leftarrow (M) - 1$ or $X \leftarrow (X) - 1$ $PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$ $PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$ $PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$ $PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$ $PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$ $PC \leftarrow (PC) + 4 + rel ? (result) \neq 0$	—	—	—	—	—	—	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC opr DECA DECX DEC opr,X DEC ,X DEC opr,SP	Decrement	$M \leftarrow (M) - 1$ $A \leftarrow (A) - 1$ $X \leftarrow (X) - 1$ $M \leftarrow (M) - 1$ $M \leftarrow (M) - 1$ $M \leftarrow (M) - 1$	↑	—	—	↑	↑	—	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd  ff ff	4 1 1 4 3 5
DIV	Divide	$A \leftarrow (H:A)/(X)$ $H \leftarrow \text{Remainder}$	—	—	—	—	↑	↑	INH	52		7
EOR #opr EOR opr EOR opr EOR opr,X EOR opr,X EOR ,X EOR opr,SP EOR opr,SP	Exclusive OR M with A	$A \leftarrow (A \oplus M)$	0	—	—	↑	↑	—	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
INC opr INCA INCX INC opr,X INC ,X INC opr,SP	Increment	$M \leftarrow (M) + 1$ $A \leftarrow (A) + 1$ $X \leftarrow (X) + 1$ $M \leftarrow (M) + 1$ $M \leftarrow (M) + 1$ $M \leftarrow (M) + 1$	↑	—	—	↑	↑	—	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd  ff ff	4 1 1 4 3 5
JMP opr JMP opr JMP opr,X JMP opr,X JMP ,X	Jump	$PC \leftarrow \text{Jump Address}$	—	—	—	—	—	—	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR opr JSR opr JSR opr,X JSR opr,X JSR ,X	Jump to Subroutine	$PC \leftarrow (PC) + n$ ( $n = 1, 2, \text{ or } 3$ ) Push (PCL); $SP \leftarrow (SP) - 1$ Push (PCH); $SP \leftarrow (SP) - 1$ $PC \leftarrow \text{Unconditional Address}$	—	—	—	—	—	—	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	4 5 6 5 4
LDA #opr LDA opr LDA opr LDA opr,X LDA opr,X LDA ,X LDA opr,SP LDA opr,SP	Load A from M	$A \leftarrow (M)$	0	—	—	↑	↑	—	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LDHX #opr LDHX opr	Load H:X from M	$H:X \leftarrow (M:M + 1)$	0	—	—	↑	↑	—	IMM DIR	45 55	ii jj dd	3 4
LDX #opr LDX opr LDX opr LDX opr,X LDX opr,X LDX ,X LDX opr,SP LDX opr,SP	Load X from M	$X \leftarrow (M)$	0	—	—	↑	↑	—	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL opr LSLA LSLX LSL opr,X LSL ,X LSL opr,SP	Logical Shift Left (Same as ASL)		↑	—	—	↑	↑	↑	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 4 3 5
LSR opr LSRA LSRX LSR opr,X LSR ,X LSR opr,SP	Logical Shift Right		↑	—	—	0	↑	↑	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV opr,opr MOV opr,X+ MOV #opr,opr MOV X+,opr	Move	$(M)_{\text{Destination}} \leftarrow (M)_{\text{Source}}$ $H:X \leftarrow (H:X) + 1 \text{ (IX+D, DIX+)}$	0	—	—	↑	↑	—	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	—	0	—	—	—	0	INH	42		5
NEG opr NEGA NEGX NEG opr,X NEG ,X NEG opr,SP	Negate (Two's Complement)	$M \leftarrow -(M) = \$00 - (M)$ $A \leftarrow -(A) = \$00 - (A)$ $X \leftarrow -(X) = \$00 - (X)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$	↑	—	—	↑	↑	↑	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	—	—	—	—	—	—	INH	9D		1
NSA	Nibble Swap A	$A \leftarrow (A[3:0]:A[7:4])$	—	—	—	—	—	—	INH	62		3
ORA #opr ORA opr ORA opr ORA opr,X ORA opr,X ORA ,X ORA opr,SP ORA opr,SP	Inclusive OR A and M	$A \leftarrow (A)   (M)$	0	—	—	↑	↑	—	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); $SP \leftarrow (SP) - 1$	—	—	—	—	—	—	INH	87		2
PSHH	Push H onto Stack	Push (H); $SP \leftarrow (SP) - 1$	—	—	—	—	—	—	INH	8B		2
PSHX	Push X onto Stack	Push (X); $SP \leftarrow (SP) - 1$	—	—	—	—	—	—	INH	89		2
PULA	Pull A from Stack	$SP \leftarrow (SP + 1)$ ; Pull (A)	—	—	—	—	—	—	INH	86		2
PULH	Pull H from Stack	$SP \leftarrow (SP + 1)$ ; Pull (H)	—	—	—	—	—	—	INH	8A		2
PULX	Pull X from Stack	$SP \leftarrow (SP + 1)$ ; Pull (X)	—	—	—	—	—	—	INH	88		2
ROL opr ROLA ROLX ROL opr,X ROL ,X ROL opr,SP	Rotate Left through Carry		↑	—	—	↑	↑	↑	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5

**Table 6-1. Instruction Set Summary (Continued)**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ROR <i>opr</i> RORA RORX ROR <i>opr</i> ,X ROR ,X ROR <i>opr</i> ,SP	Rotate Right through Carry		↑	—	—	↑	↑	↑	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← \$FF	—	—	—	—	—	—	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↑	↑	↑	↑	↑	↑	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	—	—	—	—	—	—	INH	81		4
SBC # <i>opr</i> SBC <i>opr</i> SBC <i>opr</i> SBC <i>opr</i> ,X SBC <i>opr</i> ,X SBC ,X SBC <i>opr</i> ,SP SBC <i>opr</i> ,SP	Subtract with Carry	A ← (A) – (M) – (C)	↑	—	—	↑	↑	↑	IMM DIR EXT IX2 D2 IX1 E2 IX F2 SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	—	—	—	—	—	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	—	—	1	—	—	—	INH	9B		2
STA <i>opr</i> STA <i>opr</i> STA <i>opr</i> ,X STA <i>opr</i> ,X STA ,X STA <i>opr</i> ,SP STA <i>opr</i> ,SP	Store A in M	M ← (A)	0	—	—	↑	↑	—	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ff ee ff	3 4 4 3 2 4 5
STHX <i>opr</i>	Store H:X in M	(M:M + 1) ← (H:X)	0	—	—	↑	↑	—	DIR	35	dd	4
STOP	Enable IRQ Pin; Stop Oscillator	I ← 0; Stop Oscillator	—	—	0	—	—	—	INH	8E		1
STX <i>opr</i> STX <i>opr</i> STX <i>opr</i> ,X STX <i>opr</i> ,X STX ,X STX <i>opr</i> ,SP STX <i>opr</i> ,SP	Store X in M	M ← (X)	0	—	—	↑	↑	—	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ff ee ff	3 4 4 3 2 4 5
SUB # <i>opr</i> SUB <i>opr</i> SUB <i>opr</i> SUB <i>opr</i> ,X SUB <i>opr</i> ,X SUB ,X SUB <i>opr</i> ,SP SUB <i>opr</i> ,SP	Subtract	A ← (A) – (M)	↑	—	—	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ff ee ff	2 3 4 4 3 2 4 5
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	—	—	1	—	—	—	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↑	↑	↑	↑	↑	↑	INH	84		2

**Table 6-1. Instruction Set Summary (Continued)**

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
TAX	Transfer A to X	$X \leftarrow (A)$	—	—	—	—	—	—	INH	97		1
TPA	Transfer CCR to A	$A \leftarrow (CCR)$	—	—	—	—	—	—	INH	85		1
TST <i>opr</i> TSTA TSTX TST <i>opr</i> ,X TST ,X TST <i>opr</i> ,SP	Test for Negative or Zero	$(A) - \$00$ or $(X) - \$00$ or $(M) - \$00$	0	—	—	—	↑	↑	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 4
TSX	Transfer SP to H:X	$H:X \leftarrow (SP) + 1$	—	—	—	—	—	—	INH	95		2
TXA	Transfer X to A	$A \leftarrow (X)$	—	—	—	—	—	—	INH	9F		1
TXS	Transfer H:X to SP	$(SP) \leftarrow (H:X) - 1$	—	—	—	—	—	—	INH	94		2

A	Accumulator	<i>n</i>	Any bit
C	Carry/borrow bit	<i>opr</i>	Operand (one or two bytes)
CCR	Condition code register	PC	Program counter
dd	Direct address of operand	PCH	Program counter high byte
dd rr	Direct address of operand and relative offset of branch instruction	PCL	Program counter low byte
DD	Direct to direct addressing mode	REL	Relative addressing mode
DIR	Direct addressing mode	<i>rel</i>	Relative program counter offset byte
DIX+	Direct to indexed with post increment addressing mode	<i>rr</i>	Relative program counter offset byte
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing	SP1	Stack pointer, 8-bit offset addressing mode
EXT	Extended addressing mode	SP2	Stack pointer 16-bit offset addressing mode
ff	Offset byte in indexed, 8-bit offset addressing	SP	Stack pointer
H	Half-carry bit	U	Undefined
H	Index register high byte	V	Overflow bit
hh ll	High and low bytes of operand address in extended addressing	X	Index register low byte
I	Interrupt mask	Z	Zero bit
ii	Immediate operand byte	&	Logical AND
IMD	Immediate source to direct destination addressing mode		Logical OR
IMM	Immediate addressing mode	⊕	Logical EXCLUSIVE OR
INH	Inherent addressing mode	( )	Contents of
IX	Indexed, no offset addressing mode	—( )	Negation (two's complement)
IX+	Indexed, no offset, post increment addressing mode	#	Immediate value
IX+D	Indexed with post increment to direct addressing mode	«	Sign extend
IX1	Indexed, 8-bit offset addressing mode	←	Loaded with
IX1+	Indexed, 8-bit offset, post increment addressing mode	?	If
IX2	Indexed, 16-bit offset addressing mode	:	Concatenated with
M	Memory location	↑	Set or cleared
N	Negative bit	—	Not affected

## 6.7 Opcode Map

See [Table 6-2](#).

Table 6-2. Opcode Map

		Bit Manipulation		Branch	Read-Modify-Write						Control			Register/Memory									
		DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
INHERENT	IMMEDIATE	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
																					DIR	REL	REL
0		0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F			
1	BRSET0	DIR 2	REL 2	BRA 2	NEG 1 INH	NEGA 1 INH	NEGX 1 INH	NEG 4 IX1	NEG 5 SP1	NEG 3 IX	RTI 7 INH	BGE 3 REL	SUB 2 IMM	SUB 3 DIR	SUB 4 EXT	SUB 4 IX2	SUB 5 SP2	SUB 3 IX1	SUB 4 SP1	SUB 2 IX			
	BRCLR0	DIR 3	REL 3	BRN 3	CBEQA 4 IMM	CBEQ 4 IMM	CBEQX 5 IMM	CBEQ 5 IX1+	CBEQ 6 SP1	CBEQ 2 IX+	RTS 4 INH	BLT 3 REL	CMP 2 IMM	CMP 3 DIR	CMP 4 EXT	CMP 4 IX2	CMP 5 SP2	CMP 3 IX1	CMP 4 SP1	CMP 1 IX			
	BRSET1	DIR 3	REL 2	BHI 2	MUL 5 INH	MUL 5 INH	DIV 7 INH	NSA 3 INH	NSA 3 SP1	DAA 2 INH		BGT 3 REL	SBC 2 IMM	SBC 3 DIR	SBC 4 EXT	SBC 4 IX2	SBC 5 SP2	SBC 3 IX1	SBC 4 SP1	SBC 1 IX			
3	BRCLR1	DIR 3	REL 2	BLS 3	COMA 1 INH	COM 1 INH	COMX 1 INH	COM 4 IX1	COM 5 SP1	COM 3 IX	SWI 9 INH	BLE 3 REL	CPX 2 IMM	CPX 3 DIR	CPX 4 EXT	CPX 4 IX2	CPX 5 SP2	CPX 3 IX1	CPX 4 SP1	CPX 2 IX			
	BRSET2	DIR 3	REL 2	BCC 3	LSRA 4 INH	LSR 4 INH	LSRX 4 INH	LSR 4 IX1	LSR 5 SP1	LSR 3 IX	TAP 2 INH	TXS 2 INH	AND 2 IMM	AND 3 DIR	AND 4 EXT	AND 4 IX2	AND 5 SP2	AND 3 IX1	AND 4 SP1	AND 1 IX			
	BRCLR2	DIR 3	REL 2	BCS 3	LDHX 4 IMM	LDHX 4 IMM	LDHX 4 DIR	CPHX 3 IMM	CPHX 4 SP1	CPHX 2 DIR	TPA 1 INH	TSX 1 INH	BIT 2 IMM	BIT 3 DIR	BIT 4 EXT	BIT 4 IX2	BIT 5 SP2	BIT 3 IX1	BIT 4 SP1	BIT 2 IX			
6	BRSET3	DIR 3	REL 2	BNE 3	RORA 1 INH	ROR 1 INH	RORX 1 INH	ROR 4 IX1	ROR 5 SP1	ROR 3 IX	PULA 2 INH		LDA 2 IMM	LDA 3 DIR	LDA 4 EXT	LDA 4 IX2	LDA 5 SP2	LDA 3 IX1	LDA 4 SP1	LDA 2 IX			
	BRCLR3	DIR 3	REL 2	BEQ 3	ASRA 1 INH	ASR 1 INH	ASRX 1 INH	ASR 4 IX1	ASR 5 SP1	ASR 3 IX	PSHA 1 INH	TAX 1 INH	AIS 2 IMM	STA 3 DIR	STA 4 EXT	STA 4 IX2	STA 5 SP2	STA 3 IX1	STA 4 SP1	STA 2 IX			
	BRSET4	DIR 3	REL 2	BHCC 3	LSLA 4 INH	LSL 4 INH	LSLX 4 INH	LSL 4 IX1	LSL 5 SP1	LSL 3 IX	PULX 2 INH	CLC 1 INH	EOR 2 IMM	EOR 3 DIR	EOR 4 EXT	EOR 4 IX2	EOR 5 SP2	EOR 3 IX1	EOR 4 SP1	EOR 2 IX			
9	BRCLR4	DIR 3	REL 2	BHCS 3	ROLA 1 INH	ROL 1 INH	RO LX 1 INH	ROL 4 IX1	ROL 5 SP1	ROL 3 IX	PSHX 2 INH	SEC 1 INH	ADC 2 IMM	ADC 3 DIR	ADC 4 EXT	ADC 4 IX2	ADC 5 SP2	ADC 3 IX1	ADC 4 SP1	ADC 2 IX			
	BRSET5	DIR 3	REL 2	BPL 3	DECA 4 INH	DEC 4 INH	DECX 4 INH	DEC 4 IX1	DEC 5 SP1	DEC 3 IX	PULH 2 INH	CLI 2 INH	ORA 2 IMM	ORA 3 DIR	ORA 4 EXT	ORA 4 IX2	ORA 5 SP2	ORA 3 IX1	ORA 4 SP1	ORA 2 IX			
	BRCLR5	DIR 3	REL 3	BMI 3	DBNZ 5 INH	DBNZ 5 INH	DBNZX 5 INH	DBNZ 5 IX1	DBNZ 6 SP1	DBNZ 2 IX	PSHH 1 INH	SEI 2 INH	ADD 2 IMM	ADD 3 DIR	ADD 4 EXT	ADD 4 IX2	ADD 5 SP2	ADD 3 IX1	ADD 4 SP1	ADD 2 IX			
C	BRSET6	DIR 3	REL 2	BMC 3	INCA 4 INH	INC 4 INH	INCX 4 INH	INC 4 IX1	INC 5 SP1	INC 3 IX	CLRH 1 INH	RSP 1 INH	JMP 2 DIR	JMP 3 DIR	JMP 4 EXT	JMP 4 IX2	JMP 5 SP2	JMP 3 IX1	JMP 4 SP1	JMP 2 IX			
	BRCLR6	DIR 3	REL 2	BMS 3	TSTA 5 INH	TST 5 INH	TSTX 5 INH	TST 5 IX1	TST 6 SP1	TST 3 IX		NOP 1 INH	BSR 2 IMM	JSR 4 DIR	JSR 5 EXT	JSR 6 IX2	JSR 5 SP2	JSR 4 IX1	JSR 5 SP1	JSR 3 IX			
	BRSET7	DIR 3	REL 2	BIL 3	MOV 3 DIR	MOV 4 DIR	MOV 4 DIR	MOV 4 IX1	MOV 5 SP1	MOV 2 IX+	STOP 4 INH	*	LDX 2 IMM	LDX 3 DIR	LDX 4 EXT	LDX 4 IX2	LDX 5 SP2	LDX 3 IX1	LDX 4 SP1	LDX 2 IX			
F	BRCLR7	DIR 3	REL 2	BIH 3	CLRA 4 INH	CLR 4 INH	CLRX 4 INH	CLR 4 IX1	CLR 5 SP1	CLR 3 IX	WAIT 1 INH	TXA 1 INH	AIX 2 IMM	STX 3 DIR	STX 4 EXT	STX 4 IX2	STX 5 SP2	STX 3 IX1	STX 4 SP1	STX 2 IX			
	BRSET8	DIR 3	REL 2	BID 3	DD 5 INH	DD 5 INH	DD 5 INH	DD 5 IX1	DD 6 SP1	DD 3 IX			LDX 2 IMM	LDX 3 DIR	LDX 4 EXT	LDX 4 IX2	LDX 5 SP2	LDX 3 IX1	LDX 4 SP1	LDX 2 IX			
	BRCLR8	DIR 3	REL 2	BID 3	DD 5 INH	DD 5 INH	DD 5 INH	DD 5 IX1	DD 6 SP1	DD 3 IX			LDX 2 IMM	LDX 3 DIR	LDX 4 EXT	LDX 4 IX2	LDX 5 SP2	LDX 3 IX1	LDX 4 SP1	LDX 2 IX			
INHERENT	IMMEDIATE	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
																					DIR	REL	REL
0	5																						
INHERENT	IMMEDIATE	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
EXT	EXT	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
DD	DD	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			
IX+D	IX+D	DIR	REL	REL	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX			

MSB	0
LSB	5

High Byte of Opcode in Hexadecimal

Low Byte of Opcode in Hexadecimal

BRSET0

DIR

Cycles

Opcode Mnemonic

Number of Bytes / Addressing Mode

SP1 Stack Pointer, 8-Bit Offset

SP2 Stack Pointer, 16-Bit Offset

IX+ Indexed, No Offset with Post Increment

IX1+ Indexed, 1-Byte Offset with Post Increment

REL Relative

IX1 Indexed, 8-Bit Offset

IX2 Indexed, 16-Bit Offset

IMD Immediate-Direct

DIR+ Direct-Indexed

\* Pre-byte for stack pointer indexed instructions



## Section 7. Oscillator

### 7.1 Contents

7.2	Introduction . . . . .	73
7.3	Oscillator External Connections . . . . .	74
7.4	I/O Signals . . . . .	75
7.4.1	Crystal Amplifier Input Pin (OSC1) . . . . .	75
7.4.2	Crystal Amplifier Output Pin (OSC2) . . . . .	75
7.4.3	Oscillator Enable Signal (SIMOSCEN) . . . . .	75
7.4.4	External Clock Source (CGMXCLK) . . . . .	75
7.4.5	Oscillator Out (CGMOUT) . . . . .	76
7.5	Low-Power Modes . . . . .	76
7.5.1	Wait Mode . . . . .	76
7.5.2	Stop Mode . . . . .	76
7.6	Oscillator During Break Mode . . . . .	76

### 7.2 Introduction

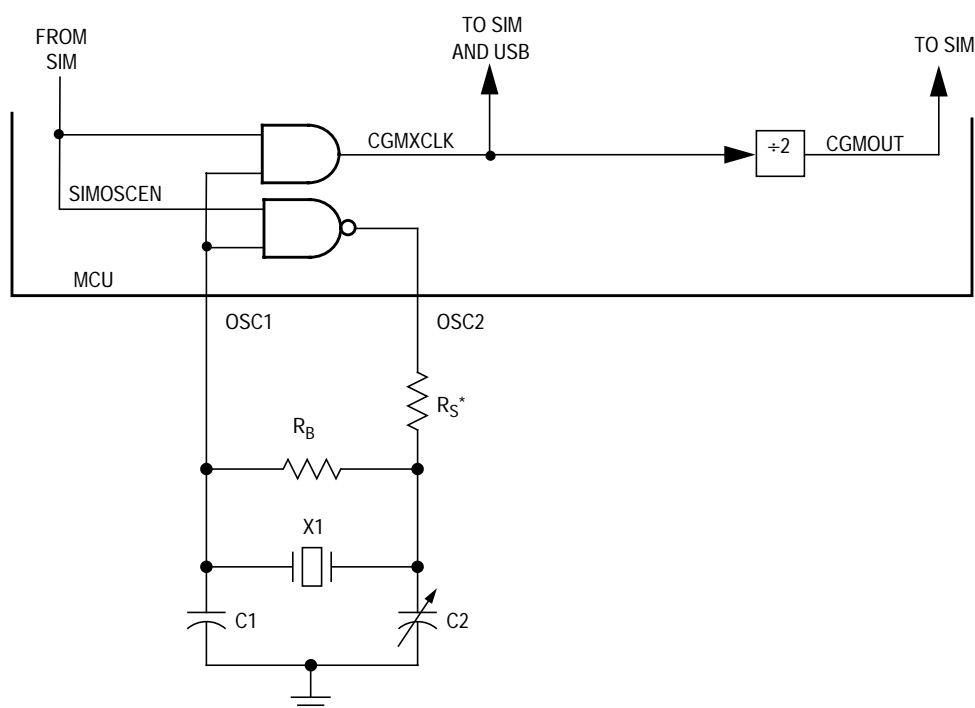
The oscillator circuit is designed for use with crystals or ceramic resonators. The oscillator circuit generates the crystal clock signal, CGMXCLK, at the frequency of the crystal. This signal is divided by two before being passed on to the system integration module (SIM) for bus clock generation.

**Figure 7-1** shows the structure of the oscillator. The oscillator requires various external components.

## 7.3 Oscillator External Connections

In its typical configuration, the oscillator requires five external components. The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in **Figure 7-1**. This figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

- Crystal,  $X_1$
- Fixed capacitor,  $C_1$
- Tuning capacitor,  $C_2$  (can also be a fixed capacitor)
- Feedback resistor,  $R_B$
- Series resistor,  $R_S$  (optional)



\*  $R_S$  can be 0 (shorted) when used with higher-frequency crystals. Refer to manufacturer's data.

**Figure 7-1. Oscillator External Connections**

The series resistor ( $R_S$ ) is included in the diagram to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high-frequency crystals. Refer to the crystal manufacturer's data for more information.

## 7.4 I/O Signals

The following paragraphs describe the oscillator input/output (I/O) signals.

### 7.4.1 Crystal Amplifier Input Pin (OSC1)

The OSC1 pin is an input to the crystal oscillator amplifier.

### 7.4.2 Crystal Amplifier Output Pin (OSC2)

The OSC2 pin is the output of the crystal oscillator inverting amplifier.

### 7.4.3 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal comes from the system integration module (SIM) and enables the oscillator.

### 7.4.4 External Clock Source (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ( $f_{XCLK}$ ) and comes directly from the crystal oscillator circuit. **Figure 7-1** shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at startup.

### 7.4.5 Oscillator Out (CGMOUT)

The clock driven to the SIM is the crystal frequency divided by two. This signal is driven to the SIM for generation of the bus clocks used by the CPU and other modules on the MCU. CGMOUT will be divided again in the SIM and results in the internal bus frequency being one fourth of the CGMXCLK frequency.

## 7.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power standby modes.

### 7.5.1 Wait Mode

The WAIT instruction has no effect on the oscillator logic. CGMXCLK continues to drive to the SIM module.

### 7.5.2 Stop Mode

The STOP instruction disables the CGMXCLK output.

## 7.6 Oscillator During Break Mode

The oscillator continues to drive CGMXCLK when the chip enters the break state.

## Section 8. System Integration Module (SIM)

### 8.1 Contents

8.2	Introduction . . . . .	78
8.3	SIM Bus Clock Control and Generation . . . . .	81
8.3.1	Bus Timing . . . . .	82
8.3.2	Clock Startup from POR . . . . .	82
8.3.3	Clocks in Stop Mode and Wait Mode . . . . .	82
8.4	Reset and System Initialization. . . . .	82
8.4.1	External Pin Reset. . . . .	83
8.4.2	Active Resets from Internal Sources. . . . .	84
8.4.2.1	Power-On Reset. . . . .	85
8.4.2.2	Computer Operating Properly (COP) Reset. . . . .	86
8.4.2.3	Illegal Opcode Reset . . . . .	86
8.4.2.4	Illegal Address Reset . . . . .	86
8.4.2.5	Universal Serial Bus Reset . . . . .	87
8.5	SIM Counter . . . . .	87
8.5.1	SIM Counter During Power-On Reset. . . . .	87
8.5.2	SIM Counter During Stop Mode Recovery . . . . .	88
8.5.3	SIM Counter and Reset States . . . . .	88
8.6	Exception Control . . . . .	88
8.6.1	Interrupts . . . . .	88
8.6.1.1	Hardware Interrupts . . . . .	91
8.6.1.2	SWI Instruction. . . . .	92
8.6.2	Interrupt Status Registers . . . . .	92
8.6.2.1	Interrupt Status Register 1 . . . . .	93
8.6.2.2	Interrupt Status Register 2 . . . . .	94
8.6.3	Reset. . . . .	95
8.6.4	Break Interrupts. . . . .	95
8.6.5	Status Flag Protection in Break Mode. . . . .	95

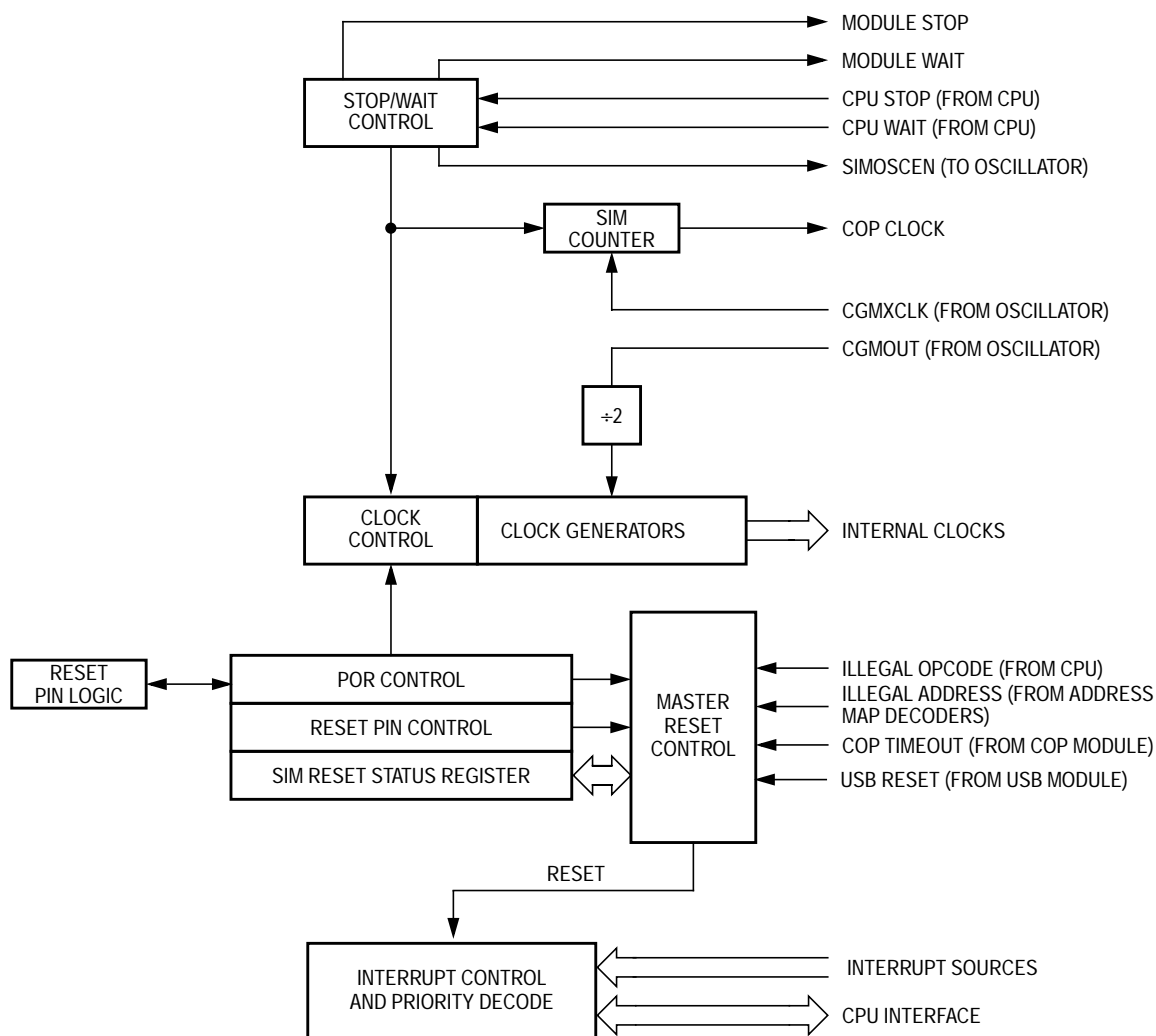
8.7	Low-Power Modes . . . . .	96
8.7.1	Wait Mode . . . . .	96
8.7.2	Stop Mode . . . . .	97
8.8	SIM Registers . . . . .	99
8.8.1	Break Status Register . . . . .	99
8.8.2	Reset Status Register . . . . .	101
8.8.3	Break Flag Control Register . . . . .	102

## 8.2 Introduction

This section describes the system integration module (SIM), which supports up to 16 external and/or internal interrupts. Together with the CPU, the SIM controls all MCU activities. The SIM is a system state controller that coordinates CPU and exception timing. A block diagram of the SIM is shown in [Figure 8-1](#). [Figure 8-2](#) is a summary of the SIM I/O registers.

The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals
  - Stop/wait/reset/break entry and recovery
  - Internal clock control
- Master reset control, including power-on reset (POR) and computer operating properly (COP) timeout
- Interrupt control:
  - Acknowledge timing
  - Arbitration control timing
  - Vector address generation
- CPU enable/disable timing
- Modular architecture expandable to 128 interrupt sources



**Figure 8-1. SIM Block Diagram**

# System Integration Module (SIM)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	0	
\$FE00	Break Status Register (BSR)	Read:							SBSW		
		Write:	R	R	R	R	R		Note 1	R	
		Reset:	0								
\$FE01	Reset Status Register (RSR)	Read:	POR	PIN	COP	ILOP	ILAD	USB	0	0	
		Write:									
		POR:	1	0	0	0	0	0	0	0	0
\$FE03	Break Flag Control Register (BFCR)	Read:									
		Write:	BCFE	R	R	R	R	R	R	R	R
		Reset:	0								
\$FE04	Interrupt Status Register 1 (INT1)	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0	
		Write:	R	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0	0
\$FE05	Interrupt Status Register 2 (INT2)	Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7	
		Write:	R	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0	0
Note 1. Writing a logic 0 clears SBSW.				= Unimplemented		R	= Reserved				

**Figure 8-2. SIM Register Summary**

**NOTE:** Writing to a reserved address can have unpredictable effects on MCU operation.



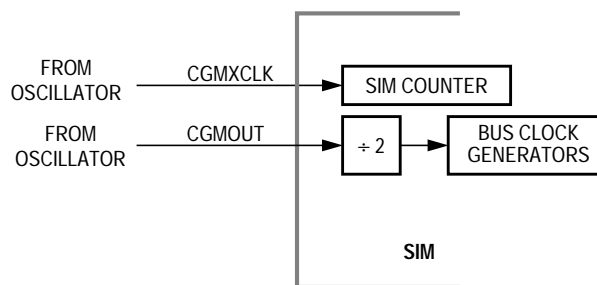
**Table 8-1** shows the internal signal names used in this section.

**Table 8-1. Signal Name Conventions**

Signal Name	Description
CGMXCLK	Buffered OSC1 from the oscillator
CGMOUT	The CGMXCLK frequency divided by two. This signal is again divided by two in the SIM to generate the internal bus clocks. (Bus clock = CGMXCLK divided by four)
IAB	Internal address bus
IDB	Internal data bus
PORRST	Signal from the power-on reset module to the SIM
IRST	Internal reset signal
R/ $\overline{W}$	Read/write signal

### 8.3 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in **Figure 8-3**.



**Figure 8-3. SIM Clock Signals**

### 8.3.1 Bus Timing

In user mode, the internal bus frequency is the oscillator frequency (CGMXCLK) divided by four.

### 8.3.2 Clock Startup from POR

When the power-on reset (POR) module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 CGMXCLK cycle POR timeout has completed. The  $\overline{\text{RST}}$  pin is driven low by the SIM during this entire period. The IBUS clocks start upon completion of the timeout.

### 8.3.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See [8.7.2 Stop Mode](#).)

In wait mode, the CPU clocks are inactive. The SIM also produces two sets of clocks for other modules. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

## 8.4 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ( $\overline{\text{RST}}$ )
- Computer operating properly module (COP)
- Illegal opcode
- Illegal address
- Universal serial bus module (USB)

All of these resets produce the vector \$FFFE–FFFF (\$FEFE–FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

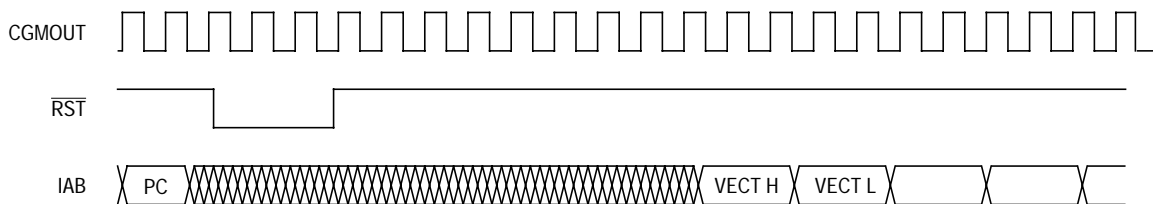
An internal reset clears the SIM counter (see [8.5 SIM Counter](#)), but an external reset does not. Each of the resets sets a corresponding bit in the reset status register (RSR). (See [8.8 SIM Registers](#).)

#### 8.4.1 External Pin Reset

The  $\overline{\text{RST}}$  pin circuits include an internal pullup device. Pulling the asynchronous  $\overline{\text{RST}}$  pin low halts all processing. The PIN bit of the reset status register (RSR) is set as long as  $\overline{\text{RST}}$  is held low for a minimum of 67 CGMXCLK cycles, assuming that the POR was not the source of the reset. See [Table 8-2](#) for details. [Figure 8-4](#) shows the relative timing.

**Table 8-2. PIN Bit Set Timing**

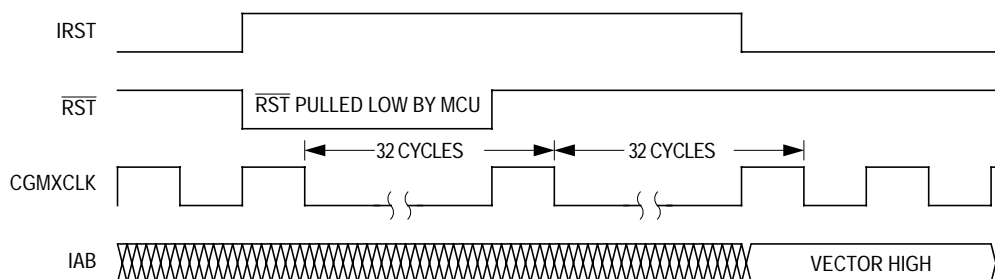
Reset Type	Number of Cycles Required to Set PIN
POR	4163 (4096 + 64 + 3)
All Others	67 (64 + 3)



**Figure 8-4. External Reset Timing**

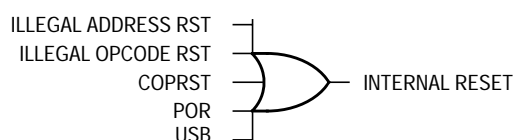
## 8.4.2 Active Resets from Internal Sources

All internal reset sources actively pull the  $\overline{\text{RST}}$  pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles. (See [Figure 8-5](#).) An internal reset can be caused by an illegal address, illegal opcode, COP timeout, the USB module, or POR. (See [Figure 8-6](#).) Note that for POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the  $\overline{\text{RST}}$  pin low. The internal reset signal then follows the sequence from the falling edge of  $\overline{\text{RST}}$  shown in [Figure 8-5](#).



**Figure 8-5. Internal Reset Timing**

The COP reset is asynchronous to the bus clock.



**Figure 8-6. Sources of Internal Reset**

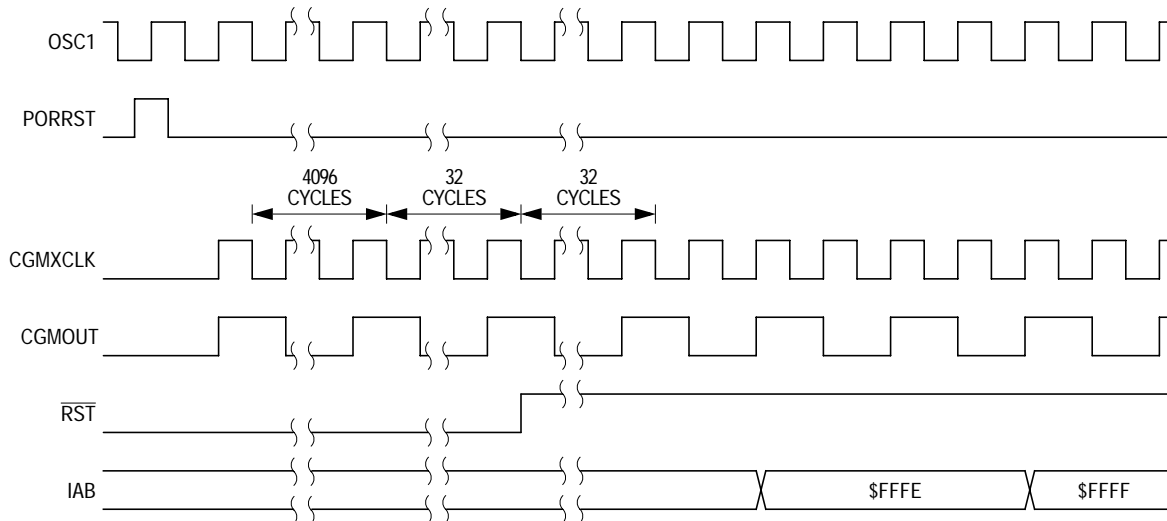
The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

### 8.4.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ( $\overline{\text{RST}}$ ) is held low while the SIM counter counts out 4096 CGMXCLK cycles. Sixty-four CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, these events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables the oscillator to drive CGMXCLK.
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the oscillator.
- The  $\overline{\text{RST}}$  pin is driven low during the oscillator stabilization time.
- The POR bit of the reset status register (RSR) is set and all other bits in the register are cleared.



**Figure 8-7. POR Recovery**

### 8.4.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the reset status register (RSR). The SIM actively pulls down the  $\overline{\text{RST}}$  pin for all internal reset sources.

To prevent a COP module timeout, write any value to location \$FFFF. Writing to location \$FFFF clears the COP counter and stages 12 through 5 of the SIM counter. The SIM counter output, which occurs at least every  $2^{12} - 2^4$  CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

The COP module is disabled if the  $\overline{\text{RST}}$  pin or the  $\overline{\text{IRQ1}}$  pin is held at  $V_{\text{TST}}$  while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high voltage signal on the  $\overline{\text{RST}}$  or the  $\overline{\text{IRQ1}}$  pin. This prevents the COP from becoming disabled as a result of external noise. During a break state,  $V_{\text{TST}}$  on the  $\overline{\text{RST}}$  pin disables the COP module.

### 8.4.2.3 Illegal Opcode Reset

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the reset status register (RSR) and causes a reset.

If the stop enable bit, STOP, in the mask option register is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the  $\overline{\text{RST}}$  pin for all internal reset sources.

### 8.4.2.4 Illegal Address Reset

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the reset status register (RSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the  $\overline{\text{RST}}$  pin for all internal reset sources.

#### 8.4.2.5 Universal Serial Bus Reset

The USB module will detect a reset signaled on the bus by the presence of an extended SE0 at the USB data pins of a device. The reset signaling is specified to be present for a minimum of 10 ms. An active device (powered and not in the suspend state) seeing a single-ended 0 on its USB data inputs for more than 2.5  $\mu$ s may treat that signal as a reset, but must have interpreted the signaling as a reset within 5.5  $\mu$ s. For a low-speed device, an SE0 condition between 4 and 8 low-speed bit times represents a valid USB reset. After the reset is removed, the device will be in the attached, but not yet addressed or configured, state (refer to Section 9.1 USB Devices of the *Universal Serial Bus Specification* Rev. 1.0). The device must be able to accept the device address via a SET\_ADDRESS command (refer to Section 9.4 of the *Universal Serial Bus Specification* Rev. 1.0) no later than 10 ms after the reset is removed.

Reset can wake a device from the suspended mode. A device may take up to 10 ms to wake up from the suspended state.

### 8.5 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter uses 12 stages for counting, followed by a 13th stage that triggers a reset of SIM counters and supplies the clock for the COP module. The SIM counter is clocked by the falling edge of CGMXCLK.

#### 8.5.1 SIM Counter During Power-On Reset

The power-on reset module (POR) detects power applied to the MCU. At power-on, the POR circuit asserts the signal PORRST. Once the SIM is initialized, it enables the oscillator to drive the bus clock state machine.

### 8.5.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the mask option register. If the SSREC bit is a logic 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long startup times from stop mode. External crystal applications should use the full stop recovery time by having SSREC cleared in the mask option register.

### 8.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See [8.7.2 Stop Mode](#) for details.) The SIM counter is free-running after all reset states. (See [8.4.2 Active Resets from Internal Sources](#) for counter control and internal reset recovery sequences.)

## 8.6 Exception Control

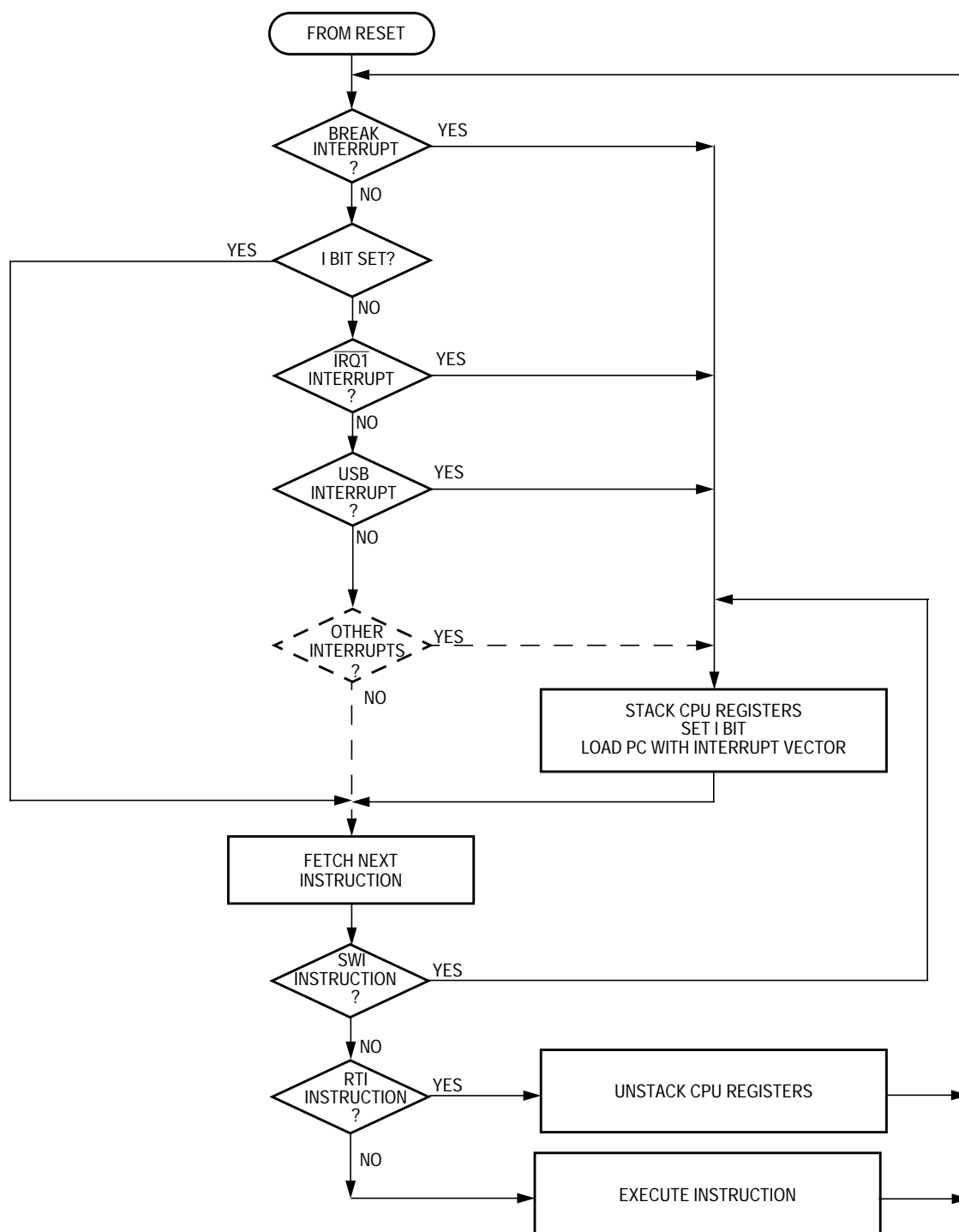
Normal, sequential program execution can be changed in three different ways:

- Interrupts
  - Maskable hardware CPU interrupts
  - Non-maskable software interrupt instruction (SWI)
- Reset
- Break interrupts

### 8.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. The flowchart in [Figure 8-8](#) shows the handling of system interrupts.

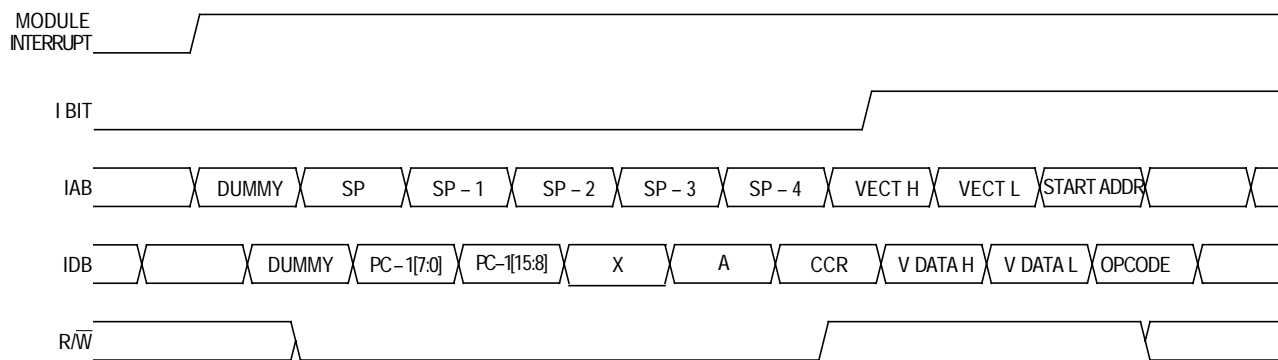




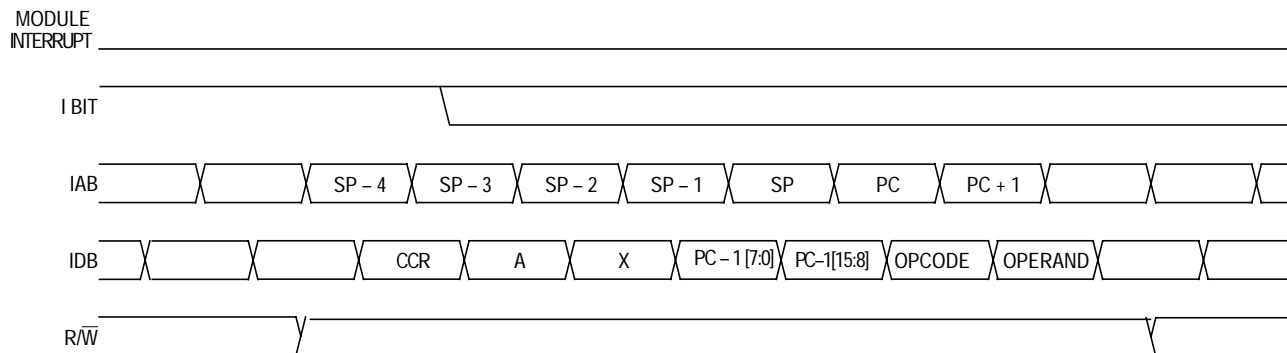
**Figure 8-8. Interrupt Processing**

Interrupts are latched and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced or the I bit is cleared.

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 8-9** shows interrupt entry timing. **Figure 8-10** shows interrupt recovery timing.



**Figure 8-9. Interrupt Entry**

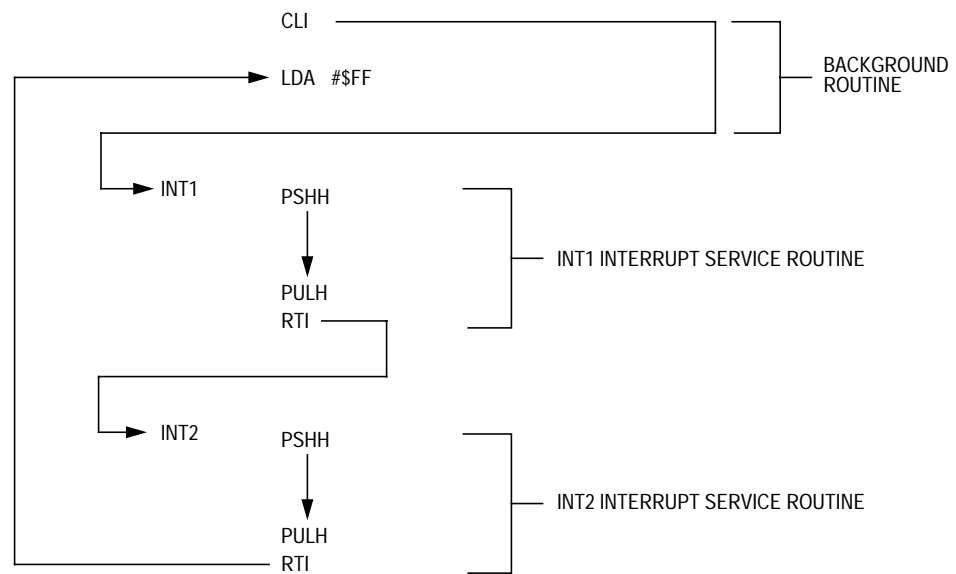


**Figure 8-10. Interrupt Recovery**

### 8.6.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register) and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. **Figure 8-11** demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.



**Figure 8-11. Interrupt Recognition Example**

The LDA opcode is prefetched by both the INT1 and INT2 RTI instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

**NOTE:** *To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.*

## 8.6.1.2 SWI Instruction

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I bit) in the condition code register.

**NOTE:** A software interrupt pushes PC onto the stack. A software interrupt does *not* push PC-1, as a hardware interrupt does.

## 8.6.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. [Table 8-3](#) summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

**Table 8-3. Interrupt Sources**

Source	Flags	Mask <sup>(1)</sup>	INT Register Flag	Priority <sup>(2)</sup>	Vector Address
SWI Instruction	—	—	—	0	\$FFFC-\$FFFD
$\overline{\text{IRQ1}}$ Pin	IRQF1	IMASK1	IF1	1	\$FFFA-\$FFFB
USB Endpoint 0 Transmit	TXD0F	TXD0IE	IF2	2	\$FFF8-\$FFF9
USB Endpoint 0 Receive	RXD0F	RXD0IE			
USB Endpoint 1/ Endpoint 2 Transmit	TXD1F	TXD1IE			
USB End of Packet	EOPF	EOPIE			
USB Resume Interrupt	RESUMF	—			
TIM Channel 0	CH0F	CH0IE	IF3	3	\$FFF6-\$FFF7
TIM Channel 1	CH1F	CH1IE	IF4	4	\$FFF4-\$FFF5
TIM Overflow	TOF	TOIE	IF5	5	\$FFF2-\$FFF3
Keyboard Pins	KEYF	IMASKK	IF6	6	\$FFF0-\$FFF1

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.

2. 0 = highest priority

### 8.6.2.1 Interrupt Status Register 1

Address: \$FE04

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

**Figure 8-12. Interrupt Status Register 1 (INT1)**

**NOTE:** Writing to a reserved address can have unpredictable effects on MCU operation.

IF6–IF1 — Interrupt Flags 1–6

These flags indicate the presence of interrupt requests from the sources shown in [Table 8-3](#).

1 = Interrupt request present

0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

## 8.6.2.2 Interrupt Status Register 2

Address: \$FE05

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

**Figure 8-13. Interrupt Status Register 2 (INT2)**

**NOTE:** Writing to a reserved address can have unpredictable effects on MCU operation.

IF14–IF7 — Interrupt Flags 14–7

Since the MC68HC08KL8 does not use these interrupt flags, these bits will always read 0.

### 8.6.3 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

### 8.6.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See [Section 16. Break Module \(BREAK\)](#).) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

### 8.6.5 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

## 8.7 Low-Power Modes

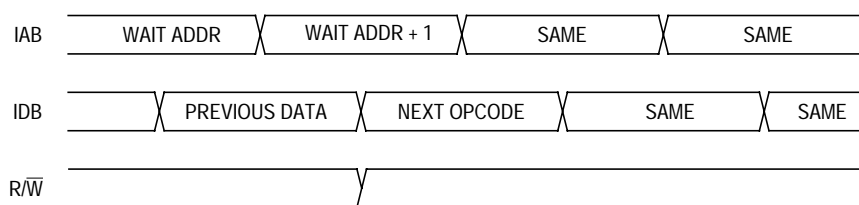
Executing the WAIT or STOP instruction puts the MCU in a low-power mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described here. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

### 8.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. **Figure 8-14** shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break status register (BSR). If the COP disable bit, COPD, in the mask option register is logic 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

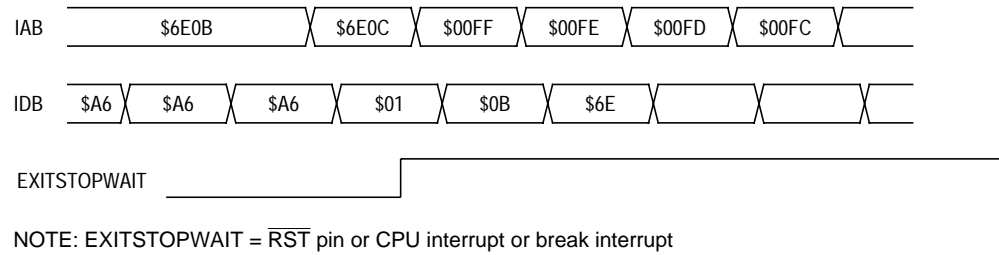


NOTE: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

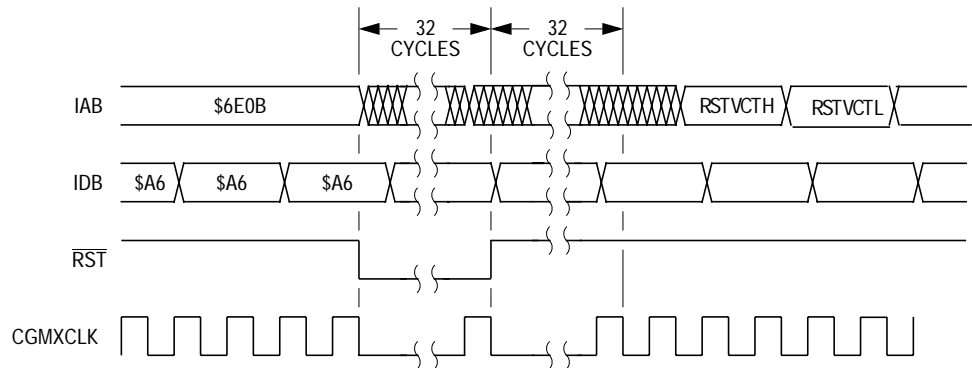
**Figure 8-14. Wait Mode Entry Timing**



**Figure 8-15** and **Figure 8-16** show the timing for WAIT recovery.



**Figure 8-15. Wait Recovery from Interrupt or Break**



**Figure 8-16. Wait Recovery from Internal Reset**

## 8.7.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

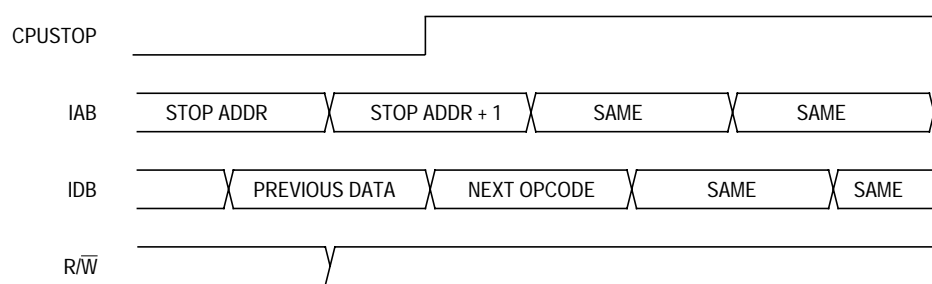
The SIM disables the oscillator signals (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the mask option register. If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long startup times from stop mode.

**NOTE:** External crystal applications should use the full stop recovery time by not selecting the SSREC option.

A break interrupt during stop mode sets the SIM break stop/wait bit (SBSW) in the break status register (BSR).

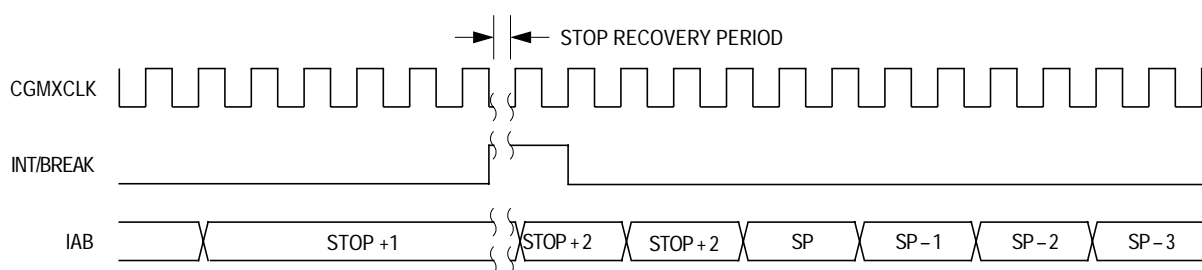
The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. **Figure 8-17** shows stop mode entry timing.

**NOTE:** To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



NOTE: Previous data can be operand data or the STOP opcode, depending on the last instruction.

**Figure 8-17. Stop Mode Entry Timing**



**Figure 8-18. Stop Mode Recovery from Interrupt or Break**

## 8.8 SIM Registers

The SIM has two break registers and one reset register.

### 8.8.1 Break Status Register

The break status register contains a flag to indicate that a break caused an exit from stop mode or wait mode.

Address: \$FE00

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note 1	
Reset:							0	

Note 1. Writing a logic 0 clears SBSW.

R
---

 = Reserved

**Figure 8-19. Break Status Register (BSR)**

**NOTE:** *Writing to a reserved address can have unpredictable effects on MCU operation.*

#### SBSW — SIM Break Stop/Wait

This status bit is useful in applications requiring a return to wait mode or stop mode after exiting from a break interrupt. Clear SBSW by writing a logic 0 to it. Reset clears SBSW.

1 = Stop mode or wait mode exited by break interrupt

0 = Stop mode or wait mode not exited by break interrupt

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

The following code is an example of this. Writing 0 to the SBSW bit clears it.

This code works if the H register has been pushed onto the stack in the break service routine software. This code should be executed at the end of the break service routine software.


```
HIBYTE EQU 5
LOBYTE EQU 6
;      If not SBSW, do RTI
      BRCLR SBSW,BSR, RETURN ; See if wait mode or stop mode was exited
                                ; by break.
      TST   LOBYTE,SP         ; If RETURNLO is not zero,
      BNE   DOLO              ; then just decrement low byte.
      DEC   HIBYTE,SP         ; Else deal with high byte, too.
DOLO    DEC   LOBYTE,SP       ; Point to WAIT/STOP opcode.
RETURN  PULH                  ; Restore H register.
      RTI
```

## 8.8.2 Reset Status Register

This register contains six flags that show the source of the last reset. All flag bits are cleared automatically following a read of the register. The register is initialized on power-up as shown with the POR bit set and all other bits cleared. However, during a POR or any other internal reset, the  $\overline{\text{RST}}$  pin is pulled low. After the pin is released, it will be sampled 32 CGMXCLK cycles later. If the pin is not above a  $V_{IH}$  at that time, then the PIN bit in the RSR may be set in addition to whatever other bits are set.

Address: \$FE01

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	ILAD	USB	0	0
Write:								
POR:	1	0	0	0	0	0	0	0

 = Unimplemented

**Figure 8-20. Reset Status Register (RSR)**

**POR** — Power-On Reset Bit  
1 = A POR has occurred.  
0 = Read of RSR

**PIN** — External Reset Bit  
1 = An external reset has occurred since the last read of the RSR.  
0 = Read of RSR

**COP** — Computer Operating Properly Reset Bit  
1 = A COP reset has occurred since the last read of the RSR.  
0 = POR or read of RSR

**ILOP** — Illegal Opcode Reset Bit  
1 = An illegal opcode reset has occurred since the last read of the RSR.  
0 = POR or read of RSR

ILAD — Illegal Address Reset Bit (opcode fetches only)

1 = An illegal address reset has occurred since the last read of the RSR.

0 = POR or read of RSR

USB — Universal Serial Bus Reset Bit

1 = Last reset caused by an USB module

0 = POR or read of RSR

## 8.8.3 Break Flag Control Register

The break control register contains a bit that enables software to clear status bits while the MCU is in a break state.

Address: \$FE03

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	BCFE	R	R	R	R	R	R	R
Write:								
POR:	0							
		R						

= Reserved

**Figure 8-21. Break Flag Control Register (BFCR)**

**NOTE:** Writing to a reserved address can have unpredictable effects on MCU operation.

BCFE — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break

0 = Status bits not clearable during break

## Section 9. Universal Serial Bus Module (USB)

### 9.1 Contents

9.2	Features .....	105
9.3	Overview .....	106
9.3.1	USB Protocol .....	107
9.3.1.1	Sync Pattern .....	109
9.3.1.2	Packet Identifier Field .....	110
9.3.1.3	Address Field (ADDR) .....	110
9.3.1.4	Endpoint Field (ENDP) .....	110
9.3.1.5	Cyclic Redundancy Check (CRC) .....	111
9.3.1.6	End-of-Packet (EOP) .....	113
9.3.2	Reset Signaling .....	114
9.3.3	Suspend .....	115
9.3.4	Resume After Suspend .....	115
9.3.4.1	Host Initiated Resume .....	115
9.3.4.2	USB Reset Signalling .....	116
9.3.4.3	Remote Wakeup .....	116
9.3.5	Low-Speed Device .....	117
9.4	Clock Requirements .....	117
9.5	Hardware Description .....	118
9.5.1	Voltage Regulator .....	118
9.5.2	Regulator Bypass Option .....	119
9.5.3	USB Transceiver .....	120
9.5.3.1	Output Driver Characteristics .....	120
9.5.3.2	Low Speed (1.5 Mbs) Driver Characteristics .....	120
9.5.3.3	Receiver Data Jitter .....	122
9.5.3.4	Data Source Jitter .....	122
9.5.3.5	Data Signal Rise and Fall Time .....	123

9.5.4	USB Control Logic . . . . .	124
9.5.4.1	Data Encoding/Decoding . . . . .	125
9.5.4.2	Bit Stuffing . . . . .	126
9.6	I/O Register Description . . . . .	127
9.6.1	USB Address Register . . . . .	131
9.6.2	USB Interrupt Register 0 . . . . .	132
9.6.3	USB Interrupt Register 1 . . . . .	134
9.6.4	USB Control Register 0 . . . . .	136
9.6.5	USB Control Register 1 . . . . .	138
9.6.6	USB Control Register 2 . . . . .	140
9.6.7	USB Status Register . . . . .	142
9.6.8	USB Endpoint 0 Data Registers . . . . .	143
9.6.9	USB Endpoint 1/Endpoint 2 Data Registers . . . . .	144
9.7	USB Interrupts . . . . .	145
9.7.1	USB End-of-Transaction Interrupt . . . . .	145
9.7.1.1	Receive Control Endpoint 0 . . . . .	145
9.7.1.2	Transmit Control Endpoint 0 . . . . .	148
9.7.1.3	Transmit Endpoint 1 and Transmit Endpoint 2 . . . . .	149
9.7.2	Resume Interrupt . . . . .	150
9.7.3	End-of-Packet Interrupt . . . . .	150



## 9.2 Features

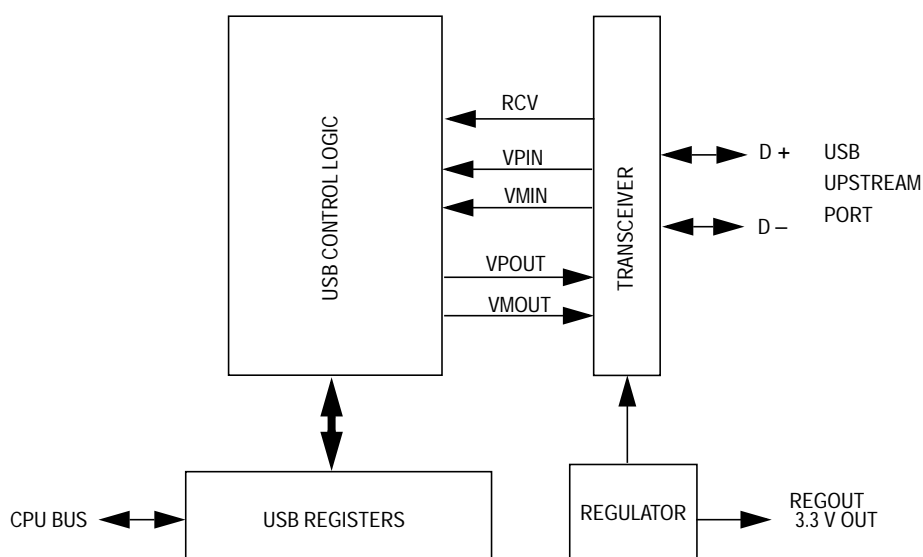
Features of the USB (universal serial bus) module include:

- Integrated 3.3-Volt Regulator with 3.3-Volt Output Pin
- Integrated USB Transceiver Supporting Low-Speed Functions
- USB Data Control Logic
  - Packet Decoding/Generation
  - CRC (Cyclic Redundancy Check) Generation and Checking
  - NRZI (Non-Return-to Zero Inserted) Encoding/Decoding
  - Bit-Stuffing
- USB Reset Support
- Control Endpoint 0 and Interrupt Endpoints 1 and 2
- Two 8-Byte Transmit Buffers
- One 8-Byte Receive Buffer
- Suspend and Resume Operations
  - Remote Wakeup Support
- USB-Generated Interrupts
  - Transaction Interrupt Driven
  - Resume Interrupt
  - End-of-Packet Interrupt
- Stall, Nak, and Ack Handshake Generation

## 9.3 Overview

This section provides an overview of the universal serial bus (USB) module developed for the MC68HC08KL8. This USB module is designed to serve as a low-speed (LS) USB device per the *Universal Serial Bus Specification* Rev 1.0. Three types of USB data transfers are supported: control, interrupt, and bulk (transmit only). Endpoint 0 functions as a receive/transmit control endpoint. Endpoints 1 and 2 can function as a receive/transmit control endpoint. Endpoints 1 and 2 can function as interrupt or bulk, but only in the transmit direction.

A block diagram of the USB module is shown in [Figure 9-1](#). The USB module manages communications between the host and the USB function. The module is partitioned into four functional blocks. These blocks consist of a 3.3-volt regulator, a dual-function transceiver, the USB control logic, and the endpoint registers. The blocks are further detailed later in this section (see [9.5 Hardware Description](#)).



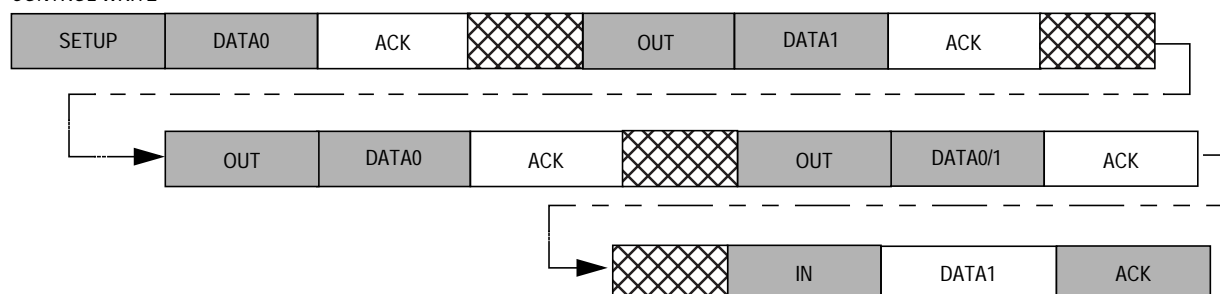
**Figure 9-1. USB Block Diagram**

### 9.3.1 USB Protocol

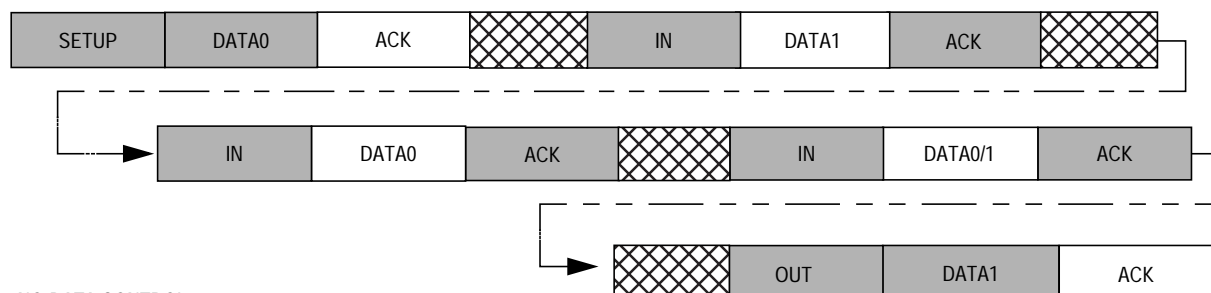
**Figure 9-2** shows the various transaction types supported by the MC68HC08KL8 USB module. The transactions are portrayed as error free. The effect of errors in the data flow are discussed later.

#### ENDPOINT 0 TRANSACTIONS:

##### CONTROL WRITE



##### CONTROL READ



##### NO-DATA CONTROL

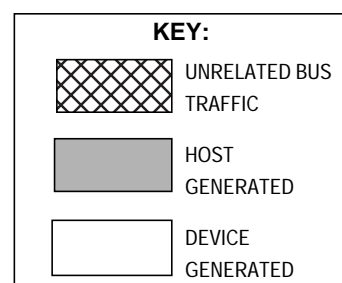


#### ENDPOINTS 1 AND 2 TRANSACTIONS:

##### INTERRUPT



##### BULK TRANSMIT



**Figure 9-2. Supported Transaction Types Per Endpoint**

Each USB transaction is comprised of a series of packets. The MC68HC08KL8 USB module supports the packet types shown in [Figure 9-3](#). Token packets are generated by the USB host and decoded by the USB device. Data and handshake packets are both decoded and generated by the USB device, depending on the type of transaction.

## Token Packet:

*IN*

*OUT*

*SETUP*

SYNC	PID	$\overline{\text{PID}}$	ADDR	ENDP	CRC5	EOP
------	-----	-------------------------	------	------	------	-----

## Data Packet:

*DATA0*

*DATA1*

SYNC	PID	$\overline{\text{PID}}$	DATA	CRC16	EOP
------	-----	-------------------------	------	-------	-----

0 – 8 Bytes

## Handshake Packet:

*ACK*

*NAK*

*STALL*

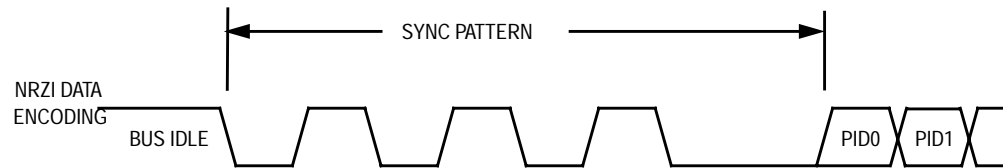
SYNC	PID	$\overline{\text{PID}}$	EOP
------	-----	-------------------------	-----

**Figure 9-3. Supported USB Packet Types**

The following subsections detail each segment used to form a complete USB transaction.

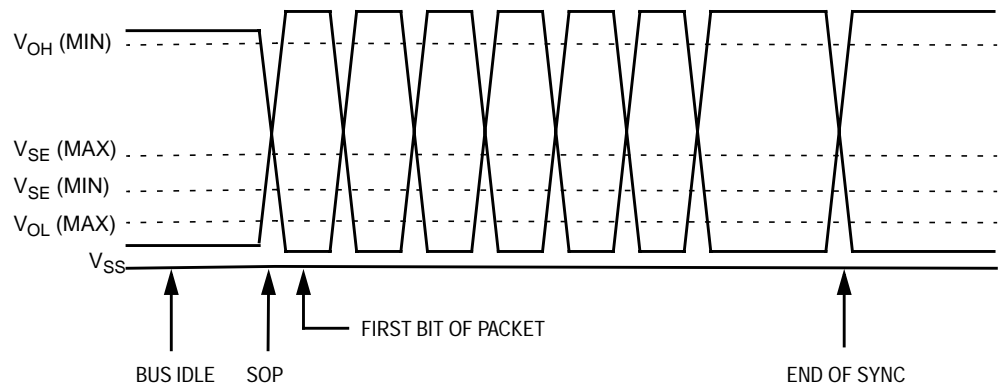
### 9.3.1.1 Sync Pattern

The NRZI (see [9.5.4.1 Data Encoding/Decoding](#)) bit pattern shown in [Figure 9-4](#) is used as a synchronization pattern and is prefixed to each packet. This pattern is equivalent to a data pattern of seven 0s followed by a 1 (\$80).



**Figure 9-4. Sync Pattern**

The start of a packet (SOP) is signaled by the originating port by driving the D+ and D– lines from the idle state (also referred to as the J state) to the opposite logic level (also referred to as the K state). This switch in levels represents the first bit of the sync field. [Figure 9-5](#) shows the data signaling and voltage levels for the start-of-packet and the sync pattern.



**Figure 9-5. SOP, Sync Signaling, and Voltage Levels**

## 9.3.1.2 Packet Identifier Field

The packet identifier field is an 8-bit number comprised of the 4-bit packet identification and its complement. The field follows the sync pattern and determines the direction and type of transaction on the bus.

**Table 9-1** shows the packet identifier values for the supported packet types.

**Table 9-1. Supported Packet Identifiers**

Packet Identifier Value	Packet Identifier Type
%1001	IN Token
%0001	OUT Token
%1101	SETUP Token
%0011	DATA0 Packet
%1011	DATA1 Packet
%0010	ACK Handshake
%1010	NAK Handshake
%1110	STALL Handshake

## 9.3.1.3 Address Field (ADDR)

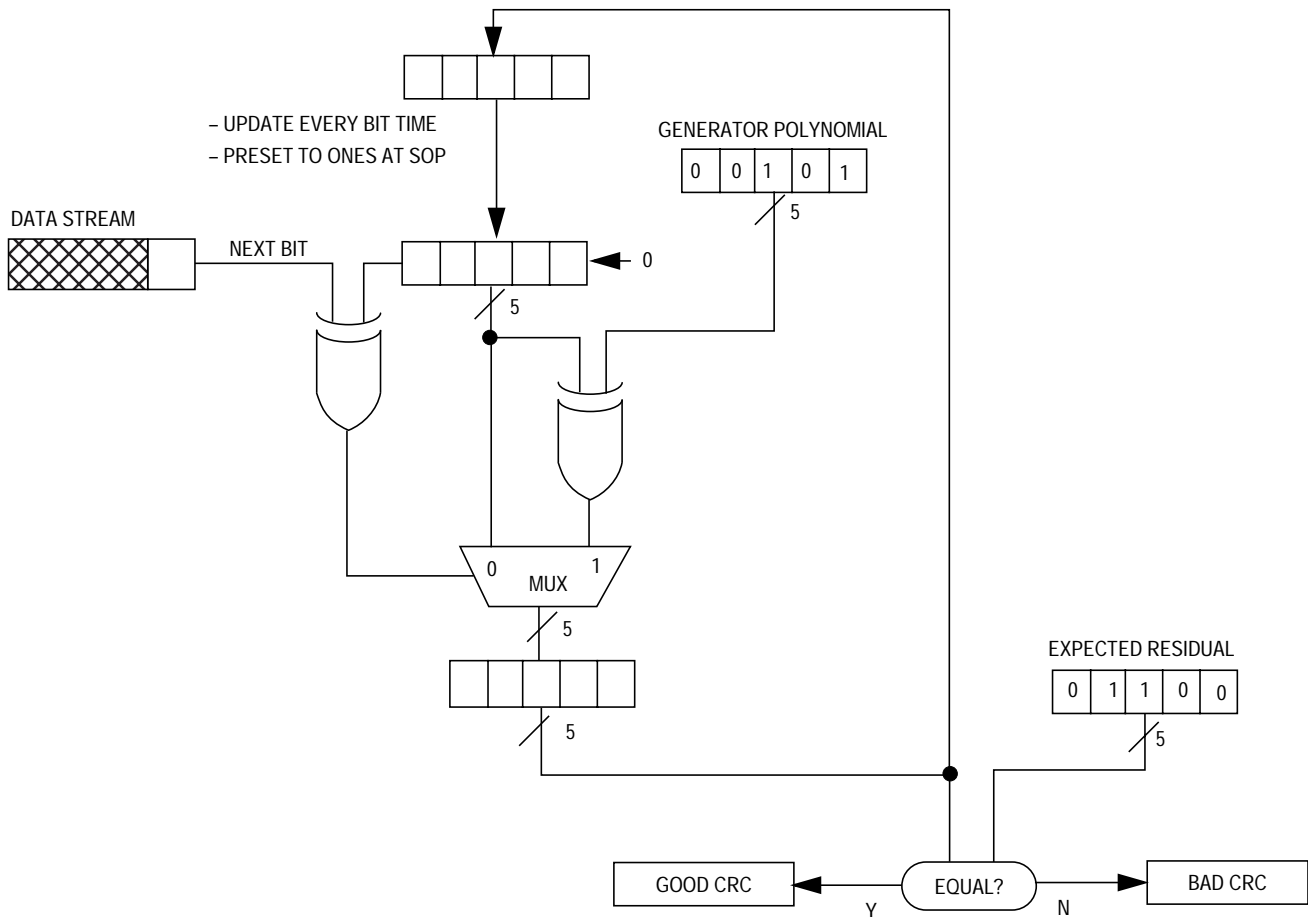
The address field is a 7-bit number that is used to select a particular USB device. This field is compared to the lower seven bits of the UADDR register to determine if a given transaction is targeting the MC68HC08KL8 USB device.

## 9.3.1.4 Endpoint Field (ENDP)

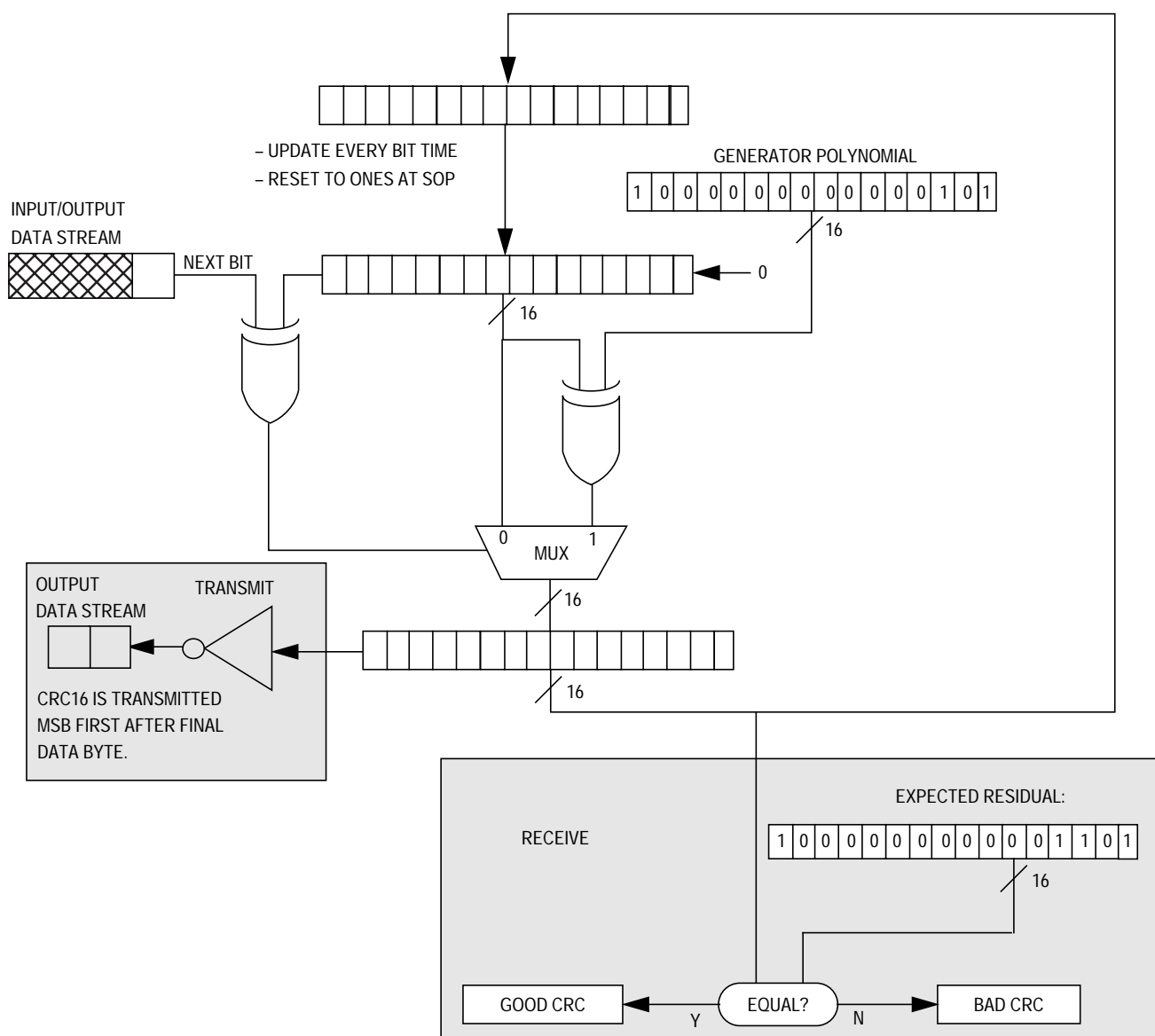
The endpoint field is a 4-bit number that is used to select a particular endpoint within a USB device. For the MC68HC08KL8, this will be a binary number between 0 and 2 inclusive. Any other value will cause the transaction to be ignored.

### 9.3.1.5 Cyclic Redundancy Check (CRC)

Cyclic redundancy checks are used to verify the address and data stream of a USB transaction. This field is five bits wide for token packets and 16 bits wide for data packets. CRCs are generated in the transmitter and sent on the USB data lines after both the endpoint field and the data field. **Figure 9-6** shows how the 5-bit CRC value is calculated from the data stream and verified for the address and endpoint fields of a token packet. **Figure 9-7** shows how the 16-bit CRC value is calculated and either transmitted or verified for the data packet of a given transaction.



**Figure 9-6. CRC Block Diagram for Token Packets**

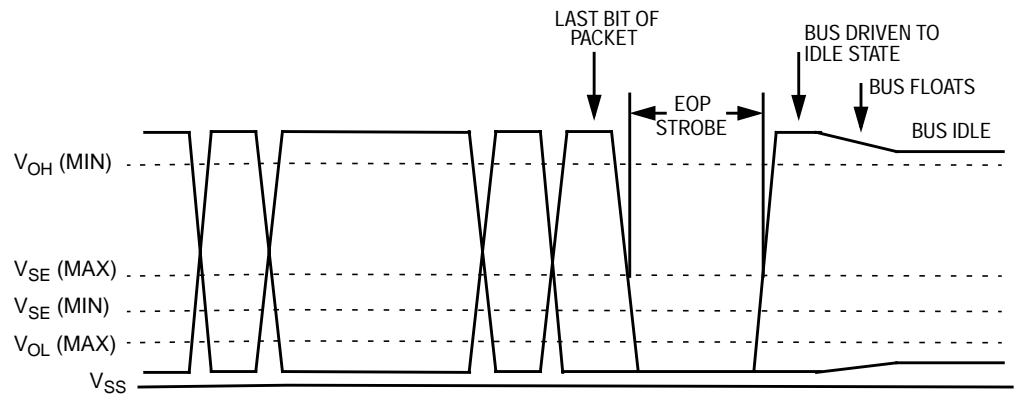


**Figure 9-7. CRC Block Diagram for Data Packets**



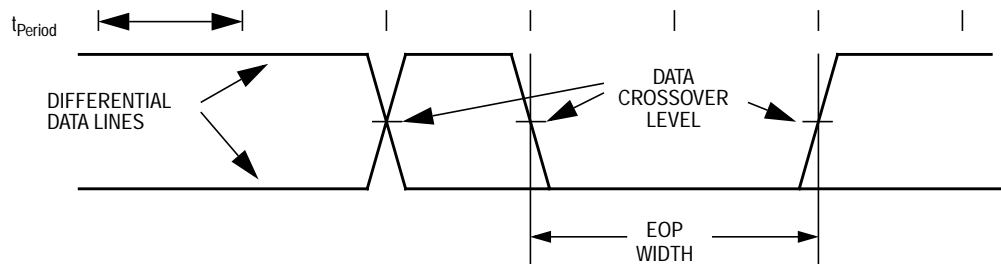
### 9.3.1.6 End-of-Packet (EOP)

The single-ended 0 (SE0) state is used to signal an end-of-packet (EOP). The single-ended 0 state is indicated by both D+ and D– being below 0.8 V. EOP will be signaled by driving D+ and D– to the single-ended 0 state for two bit times followed by driving the lines to the idle state for one bit time. The transition from the single-ended 0 to the idle state defines the end of the packet. The idle state is asserted for one bit time and then both the D+ and D– output drivers are placed in their high-impedance state. The bus termination resistors hold the bus in the idle state. **Figure 9-8** shows the data signaling and voltage levels for an end-of-packet transaction.



**Figure 9-8. EOP Transaction Voltage Levels**

The width of the SE0 in the EOP is about two bit times. The EOP width is measured with the same capacitive load used for maximum rise and fall times and is measured at the same level as the differential signal crossover points of the data lines.



**Figure 9-9. EOP Width Timing**

### 9.3.2 Reset Signaling

A reset is signaled on the bus by the presence of an extended SE0 at the USB data pins of a device. The reset signaling is specified to be present for a minimum of 10 ms. An active device (powered and not in the suspend state) seeing a single-ended 0 on its USB data inputs for more than 2.5  $\mu$ s may treat that signal as a reset, but must have interpreted the signaling as a reset within 5.5  $\mu$ s. For a low-speed device, an SE0 condition between four and eight low-speed bit times represents a valid USB reset.

A USB sourced reset will hold the 68HC6808KL8 in reset for the duration of the reset on the USB bus. The USB bit in the reset status register (RSR) will be set after the internal reset is removed. Refer to [8.8.2 Reset Status Register](#) for more detail. The MCU's reset recovery sequence is detailed in [Section 8. System Integration Module \(SIM\)](#).

The reset flag bit (RSTF) in the USB interrupt register 0 (UIR0) also will be set after the internal reset is removed. Refer to [9.6.2 USB Interrupt Register 0](#) for more detail.

After a reset is removed, the device will be in the attached, but not yet addressed or configured state (refer to Section 9.1 USB Device States of the *Universal Serial Bus Specification* Rev. 1.0). The device must be able to accept a device address via a SET\_ADDRESS command (refer to Section 9.4 Standard Device Request in the *Universal Serial Bus Specification* Rev. 1.0) no later than 10 ms after the reset is removed.

Reset can wake a device from the suspended mode. A device may take up to 10 ms to wake up from the suspended state.

### 9.3.3 Suspend

The MC68HC08KL8 supports suspend mode for low power. Suspend mode should be entered when the USB data lines are in the idle state for more than 3.0 ms. Entry into suspend mode is controlled by the SUSPND bit in the USB interrupt register. Any low-speed bus activity should keep the device out of the suspend state. Low-speed devices are kept awake by periodic low-speed EOP signals from the host. This is referred to as low speed keep alive (refer to Section 11.2.5.1 of the *Universal Serial Bus Specification* Rev. 1.0).

Firmware should monitor the EOPF flag and enter suspend mode by setting the SUSPND bit if an EOP is not detected for 3 ms.

Per the USB specification, the MC68HC08KL8 is required to draw less than 500  $\mu\text{A}$  from the  $V_{\text{DD}}$  supply when it is in the suspend state. This includes the current supplied by the voltage regulator to the 15 k $\Omega$  to ground termination resistors placed at the host end of the USB bus. This low-current requirement means that firmware is responsible for entering stop mode once the USB module has been placed in the suspend state.

### 9.3.4 Resume After Suspend

The MC68HC08KL8 can be activated from the suspend state by normal bus activity, a USB reset signal, or by a forced resume driven from the MC68HC08KL8.

#### 9.3.4.1 Host Initiated Resume

The host signals resume by initiating resume signalling (K state) for at least 20 ms followed by a standard low-speed EOP signal. This 20 ms ensures that all devices in the USB network are awakened.

After resuming the bus, the host must begin sending bus traffic within 3 ms to prevent the device from re-entering suspend mode.

### 9.3.4.2 USB Reset Signalling

Reset can wake a device from the suspended mode. A device may take up to 10 ms to wake up from the suspended state.

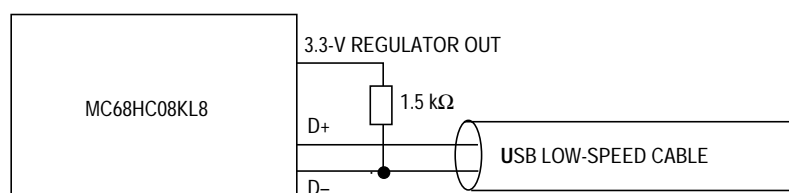
### 9.3.4.3 Remote Wakeup

The MC68HC08KL8 also supports the remote wakeup feature. The firmware has the ability to exit suspend mode by signaling a resume state to the upstream host or hub. A non-idle state (K state) on the USB data lines is accomplished by asserting the FRESUM bit in the UCR1 register.

When using the remote wakeup capability, the firmware must wait for at least 5 ms after the bus is in the idle state before sending the remote wakeup resume signaling. This allows the upstream devices to get into their suspend state and prepare for propagating resume signaling. The FRESUM bit should be asserted to cause the resume state on the USB data lines for at least 10 ms, but not more than 15 ms. Note that the resume signalling is controlled by the FRESUM bit and meeting the timing specifications is dependent on the firmware. When FRESUM is cleared by firmware, the data lines will return to their high-impedance state. Refer to the register definitions (see [9.6.5 USB Control Register 1](#)) for more information about how the force resume (FRESUM) bit can be used to initiate the remote wakeup feature.

### 9.3.5 Low-Speed Device

Externally, low-speed devices are configured by the position of a pullup resistor on the USB D<sup>−</sup> pin of the MC68HC08KL8. Low-speed devices are terminated as shown in **Figure 9-10** with the pullup on the D<sup>−</sup> line.



**Figure 9-10. External Low-Speed Device Configuration**

For low-speed transmissions, the transmitter's EOP width must be between 1.25  $\mu$ s and 1.50  $\mu$ s. These ranges include timing variations due to differential buffer delay and rise/fall time mismatches and to noise and other random effects. A low-speed receiver must accept a 670-ns SE0 followed by a J transition as a valid EOP. An SE0 shorter than 330 ns or an SE0 not followed by a J transition must be rejected as an EOP. An EOP between 330 ns and 670 ns may be rejected or accepted as discussed. Any SE0 that is 2.5  $\mu$ s or longer is automatically a reset.

## 9.4 Clock Requirements

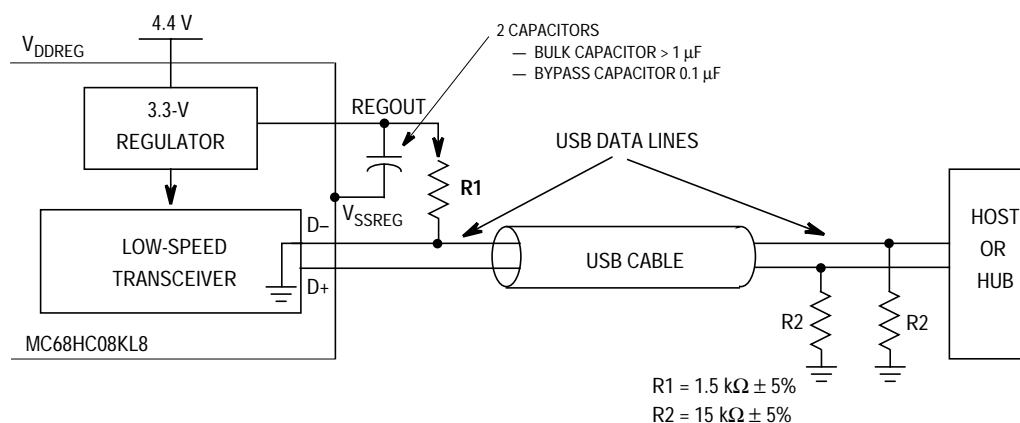
The low-speed data rate is nominally 1.5 Mbs. The CGMXCLK signal driven by the oscillator circuits is the clock source for the USB module and requires that a 6-MHz oscillator circuit be connected to the OSC1 and OSC2 pins. The permitted frequency tolerance for low-speed functions is approximately  $\pm 1.5\%$  (15,000 ppm). This tolerance includes inaccuracies from all sources: initial frequency accuracy, crystal capacitive loading, supply voltage on the oscillator, temperature, and aging. The jitter in the low-speed data rate must be less than 10 ns. This tolerance allows the use of resonators in low-cost, low-speed devices.

## 9.5 Hardware Description

The USB module as previously shown in [Figure 9-1](#) contains four functional blocks: a 3.3-volt regulator, a low-speed USB transceiver, the USB control logic, and the USB registers. The following subsections detail the function of the regulator, transceiver, and control logic. See [9.6 I/O Register Description](#) for the register discussion.

### 9.5.1 Voltage Regulator

The USB data lines are required by the USB specification to have a maximum output voltage between 2.8 V and 3.6 V. The data lines also are required to have an external 1.5-k $\Omega$  pullup resistor connected between data lines and a voltage source between 3.0 V and 3.6 V. Since the power provided by the USB cable is specified to be between 4.4 V and 5.0 V, an on-chip regulator is used to drop the voltage to the appropriate level for sourcing the USB transceiver and external pullup resistor. An output pin driven by the regulator voltage is provided to source the 1.5-k $\Omega$  external resistor. The REGOUT pin requires an external bulk capacitor 1  $\mu$ F or larger and a 0.1- $\mu$ F ceramic bypass capacitor. [Figure 9-11](#) shows the worst case electrical connection for the voltage regulator.



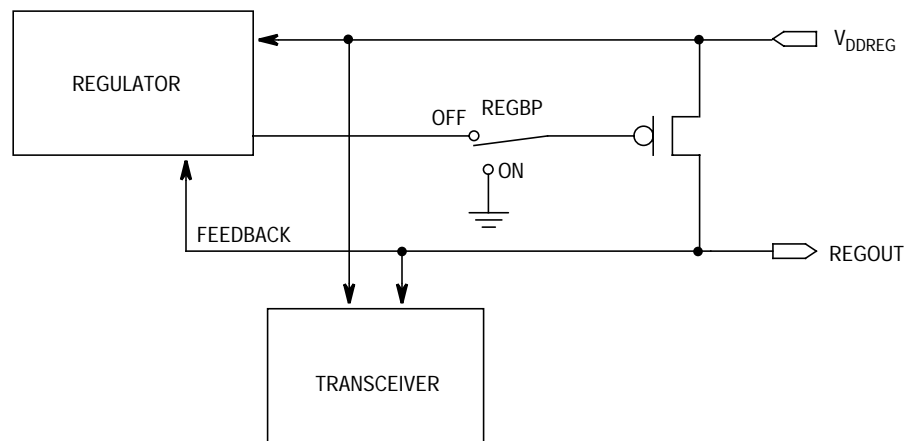
**Figure 9-11. Regulator Electrical Connections**

## 9.5.2 Regulator Bypass Option

Under normal user operation, the REGOUT voltage is sourced from the 3.3-volt regulator. The transceiver is powered by the regulator and the  $V_{DDREG}$  input. For testability of the transceiver, the 3.3-volt regulator has a bypass option that allows the REGOUT and USB transceiver power to be sourced from the  $V_{DDREG}$  input.

**NOTE:** *The bypass mode is to be used for transceiver testing and should not be used in normal operation.*

See [Section 5. Mask Option Register \(MOR\)](#) for setting the bypass option. [Figure 9-12](#) illustrates operation of the REGBP bit in the mask option register.



**Figure 9-12. Regulator Bypass Option**

### 9.5.3 USB Transceiver

The USB transceiver provides the physical interface to the USB D+ and D– data lines. The transceiver is composed of two parts: an output drive circuit and a receiver.

#### 9.5.3.1 Output Driver Characteristics

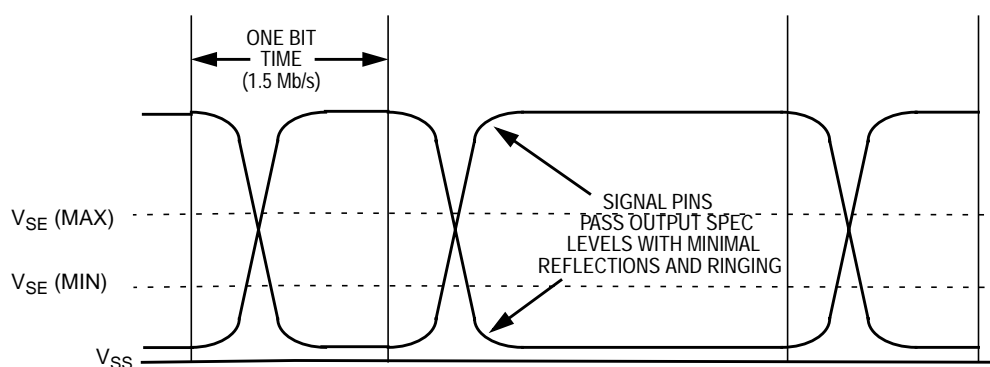
The USB transceiver uses a differential output driver to drive the USB data signal onto the USB cable. The static output swing of the driver in its low state is below the  $V_{OL}$  of 0.3 V with a 1.5-k $\Omega$  load to 3.6 V and in its high state is above the  $V_{OH}$  of 2.8 V with a 15-k $\Omega$  load to ground. The output swings between the differential high and low state are well balanced to minimize signal skew. Slew rate control on the driver is used to minimize the radiated noise and cross talk. The driver's outputs support 3-state operation to achieve bidirectional half duplex operation. The driver can tolerate a voltage on the signal pins of –0.5 V to 3.8 V with respect to local ground reference without damage.

#### 9.5.3.2 Low Speed (1.5 Mbs) Driver Characteristics

The rise and fall time of the signals on this cable are greater than 75 ns to keep RFI (radio frequency interference) emissions under FCC (Federal Communications Commission) class B limits and less than 300 ns to limit timing delays, signaling skews, and distortions. The driver reaches the specified static signal levels with smooth rise and fall times and minimal reflections and ringing when driving the cable. This driver is used only on network segments between low-speed devices and the ports to which they are connected.

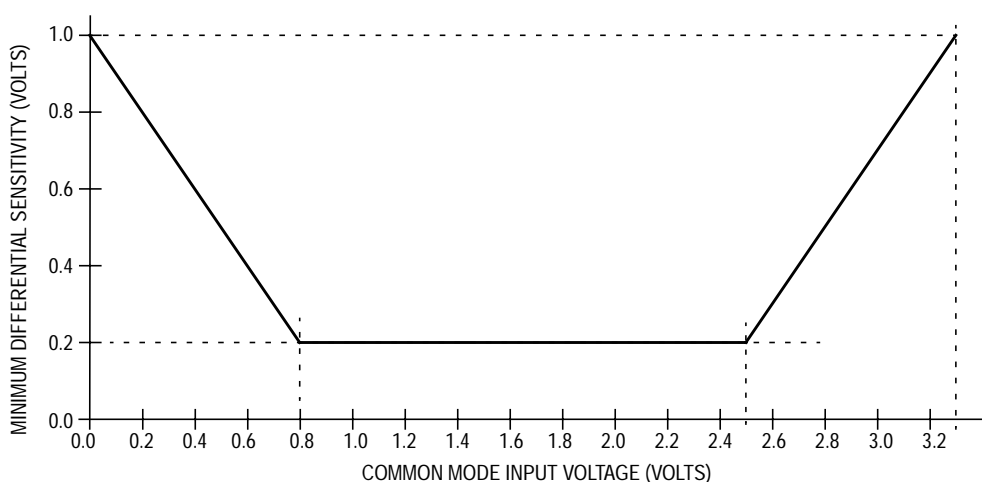
USB data transmission is done with differential signals. A differential input receiver is used to accept the USB data signal. A differential 1 on the bus is represented by D+ being at least 200 mV more positive than D– as seen at the receiver, and a differential 0 is represented by D– being at least 200 mV more positive than D+ as seen at the receiver. The signal crossover point must be between 1.3 V and 2.0 V.





**Figure 9-13. Receiver Characteristics**

The receiver features an input sensitivity of 200 mV when both differential data inputs are in the range of 0.8 V to 2.5 V with respect to the local ground reference. This is called the common mode input voltage range. Proper data reception also is achieved when the differential data lines are outside the common mode range, as shown in [Figure 9-14](#). The receiver can tolerate static input voltages between  $-0.5$  V to 3.8 V with respect to its local ground reference without damage. In addition to the differential receiver, there is a single-ended receiver (Schmitt trigger) for each of the two data lines.



**Figure 9-14. Differential Input Sensitivity  
Over Entire Common Mode Range**

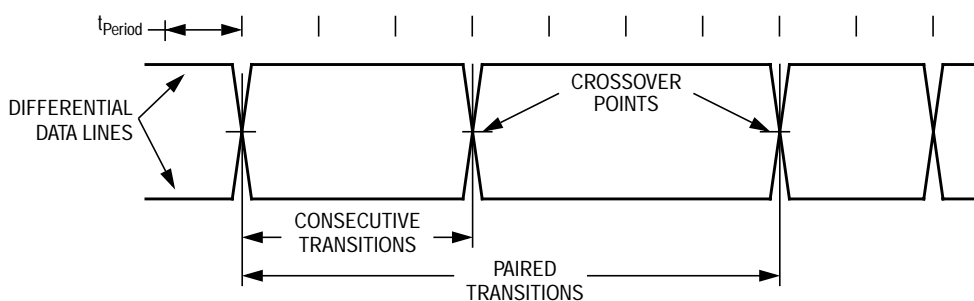
## 9.5.3.3 Receiver Data Jitter

The data receivers for all types of devices must be able to properly decode the differential data in the presence of jitter. The more of the bit time that any data edge can occupy and still be decoded, the more reliable the data transfer will be. Data receivers are required to decode differential data transitions that occur in a window plus and minus a nominal quarter bit time from the nominal (centered) data edge position.

Jitter will be caused by the delay mismatches and by mismatches in the source and destination data rates (frequencies). The receive data jitter budget for low speed is given in [Section 17. Electrical Specifications](#). The specification includes the consecutive (next) and paired transition values for each source of jitter.

## 9.5.3.4 Data Source Jitter

The source of data can have some variation (jitter) in the timing of edges of the data transmitted. The time between any set of data transitions is  $N * t_{\text{Period}} \pm \text{jitter time}$ , where  $N$  is the number of bits between the transitions and  $t_{\text{Period}}$  is defined as the actual period of the data rate. The data jitter is measured with the same capacitive load used for maximum rise and fall times and is measured at the crossover points of the data lines as shown in [Figure 9-15](#).

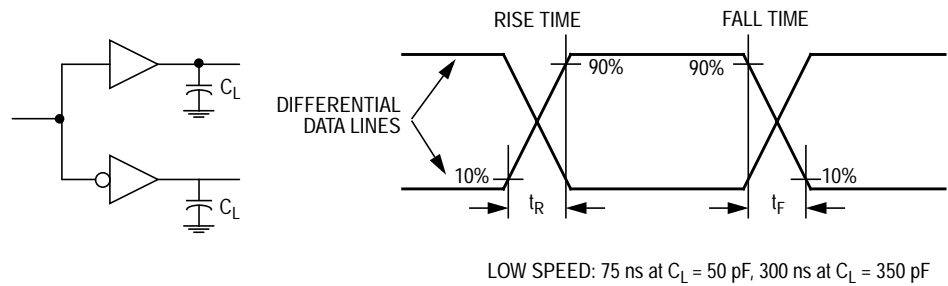


**Figure 9-15. Data Jitter**

For low-speed transmissions, the jitter time for any consecutive differential data transitions must be within  $\pm 25$  ns and within  $\pm 10$  ns for any set of paired differential data transitions. These jitter numbers include timing variations due to differential buffer delay, rise/fall time mismatches, internal clock source jitter, noise and other random effects.

#### 9.5.3.5 Data Signal Rise and Fall Time

The output rise time and fall time are measured between 10% and 90% of the signal. Edge transition time for the rising and falling edges of low-speed signals is 75 ns (minimum) into a capacitive load ( $C_L$ ) of 50 pF and 300 ns (maximum) into a capacitive load of 350 pF. The rising and falling edges should be transitioning (monotonic) smoothly when driving the cable to avoid excessive EMI (electro-magnetic interference).



**Figure 9-16. Data Signal Rise and Fall Time**

### 9.5.4 USB Control Logic

The USB control logic manages data movement between the CPU and the transceiver. The control logic handles both transmit and receive operations on the USB. It contains the logic used to manipulate the transceiver and the endpoint registers.

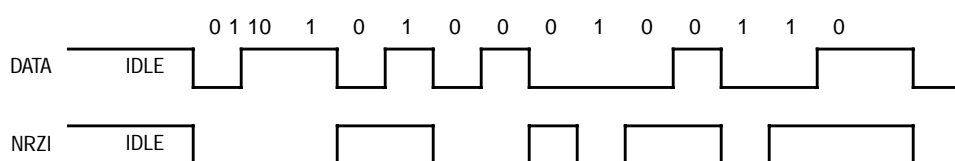
The byte count buffer is loaded with the active transmit endpoints byte count value during transmit operations. This same buffer is used for receive transactions to count the number of bytes received and, upon the end of the transaction, transfer that number to the receive endpoints byte count register.

When transmitting, the control logic handles parallel-to-serial conversion, CRC generation, NRZI encoding, and bit stuffing.

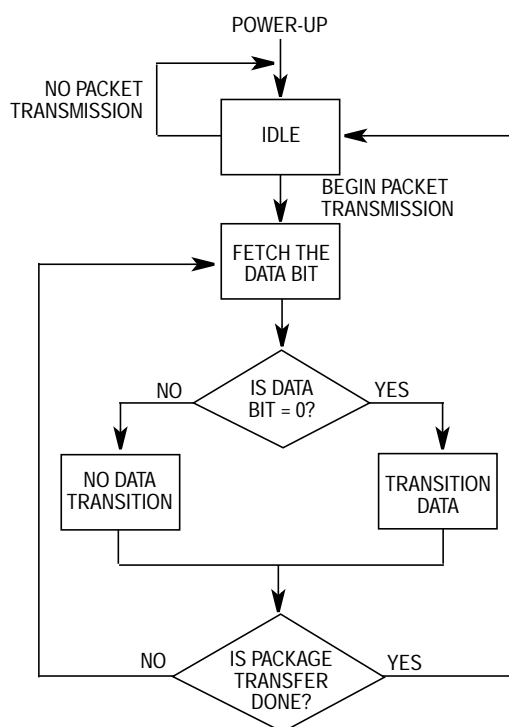
When receiving, the control logic handles sync detection, packet identification, end-of-packet detection, bit (un)stuffing, NRZI decoding, CRC validation, and serial-to-parallel conversion. Errors detected by the control logic include bad CRC, timeout while waiting for EOP, and bit stuffing violations.

### 9.5.4.1 Data Encoding/Decoding

The USB employs NRZI data encoding when transmitting packets. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. **Figure 9-17** shows a data stream and the NRZI equivalent and **Figure 9-18** is a flow diagram for NRZI. The high level represents the J state on the data lines in this and subsequent figures showing NRZI encoding. A string of 0s causes the NRZI data to toggle each bit time. A string of 1s causes long periods with no transitions in the data.



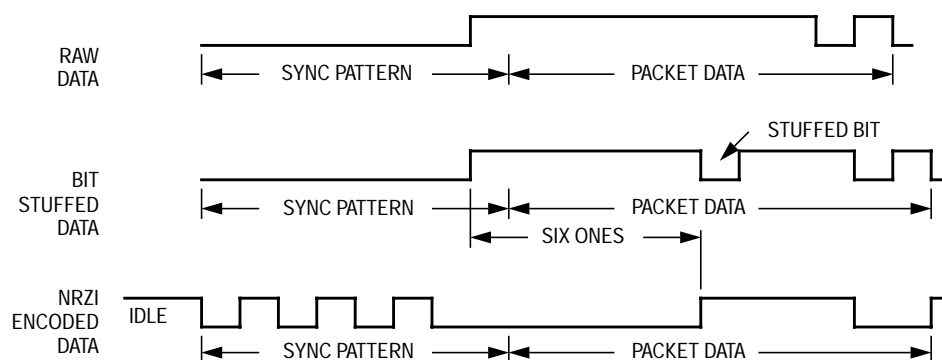
**Figure 9-17. NRZI Data Encoding**



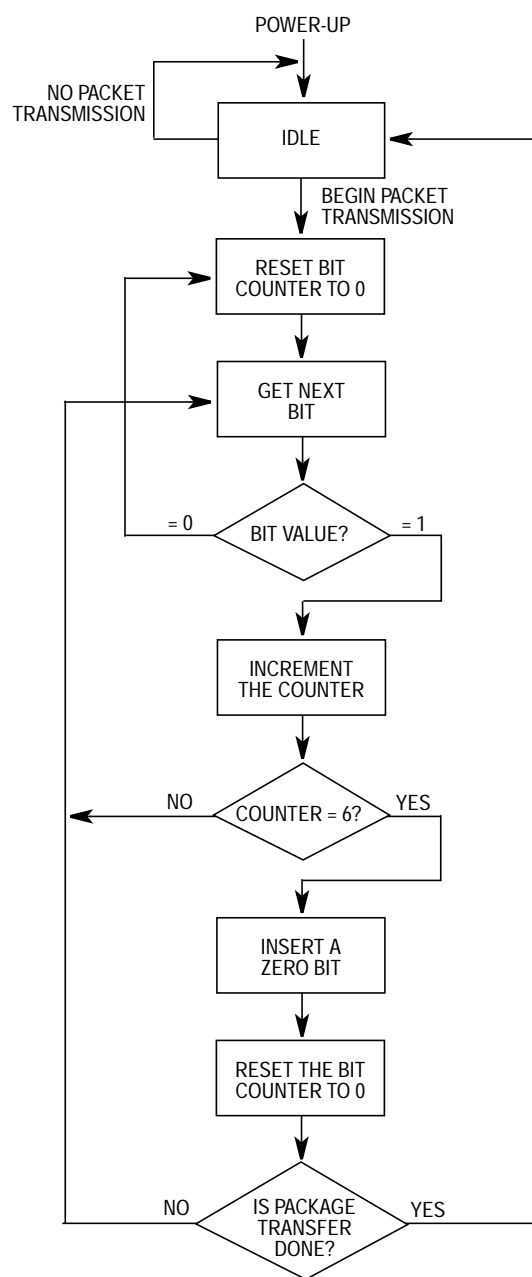
**Figure 9-18. Flow Diagram for NRZI**

## 9.5.4.2 Bit Stuffing

To ensure adequate signal transitions, bit stuffing is employed by the transmitting device when sending a packet on the USB (see [Figure 9-19](#) and [Figure 9-20](#)). A 0 is inserted after every six consecutive 1s in the data stream before the data is NRZI encoded to force a transition in the NRZI data stream. This gives the receiver logic a data transition at least once every seven bit times to guarantee the data and clock lock. The receiver must decode the NRZI data, recognize the stuffed bits, and discard them. Bit stuffing is enabled beginning with the sync pattern and throughout the entire transmission. The data 1 that ends the sync pattern is counted as the first 1 in a sequence. Bit stuffing is always enforced, without exception. If required by the bit stuffing rules, a 0 bit will be inserted even if it is the last bit before the end-of-packet (EOP) signal.



**Figure 9-19. Bit Stuffing**



**Figure 9-20. Flow Diagram for Bit Stuffing**

## 9.6 I/O Register Description

The USB endpoint registers are comprised of a set of control/status registers and 24 data registers that provide storage for the buffering of data between USB and the CPU. See [Figure 9-21](#).

# Universal Serial Bus Module (USB)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0020	USB Endpoint 0 Data Register 0 (UE0D0)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0021	USB Endpoint 0 Data Register 1 (UE0D1)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0022	USB Endpoint 0 Data Register 2 (UE0D2)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0023	USB Endpoint 0 Data Register 3 (UE0D3)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0024	USB Endpoint 0 Data Register 4 (UE0D4)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0025	USB Endpoint 0 Data Register 5 (UE0D5)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0026	USB Endpoint 0 Data Register 6 (UE0D6)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
\$0027	USB Endpoint 0 Data Register 7 (UE0D7)	Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
		Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
		Reset:	Indeterminate after Reset							
			<div></div> = Unimplemented                      X = Indeterminate							

**Figure 9-21. Register Summary (Sheet 1 of 3)**



Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0028	USB Endpoint 1/2 Data Register 0 (UE1D0)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$0029	USB Endpoint 1/2 Data Register 1 (UE1D1)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002A	USB Endpoint 1/2 Data Register 2 (UE1D2)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002B	USB Endpoint 1/2 Data Register 3 (UE1D3)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002C	USB Endpoint 1/2 Data Register 4 (UE1D4)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002D	USB Endpoint 1/2 Data Register 5 (UE1D5)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002E	USB Endpoint 1/2 Data Register 6 (UE1D6)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
\$002F	USB Endpoint 1/2 Data Register 7 (UE1D7)	Read:							
		Write:							
		UE1TD7	UE1TD6	UE1TD5	UE1TD4	UE1TD3	UE1TD2	UE1TD1	UE1TD0
		Reset: Indeterminate after Reset							
		<div></div> = Unimplemented      X = Indeterminate							

**Figure 9-21. Register Summary (Sheet 2 of 3)**

# Universal Serial Bus Module (USB)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0037	USB Control Register 2 (UCR2)	Read:	0	0	TX1ST	0	ENABLE2	ENABLE1	STALL2	STALL1
		Write:	RSTFR	TX1STR						
		Reset:	0	0	0	0	0	0	0	0
\$0038	USB Address Register (UADDR)	Read:	USBEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0039	USB Interrupt Register 0 (UIR0)	Read:	TXD0F	RXD0F	RSTF	SUSPND	TXD0IE	RXD0IE	0	0
		Write:							TXD0FR	RXD0FR
		Reset:	0	0	0	0	0	0	0	0
\$003A	USB Interrupt Register 1 (UIR1)	Read:	TXD1F	EOPF	RESUMF	0	TXD1IE	EOPIE	0	0
		Write:				RESUMFR			TXD1FR	EOPFR
		Reset:	0	0	0	0	0	0	0	0
\$003B	USB Control Register 0 (UCR0)	Read:	T0SEQ	STALL0	TX0E	RX0E	TP0SIZ3	TP0SIZ2	TP0SIZ1	TP0SIZ0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$003C	USB Control Register 1 (UCR1)	Read:	T1SEQ	ENDADD	TX1E	FRESUM	TP1SIZ3	TP1SIZ2	TP1SIZ1	TP1SIZ0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$003D	USB Status Register (USR)	Read:	RSEQ	SETUP			RPSIZ3	RPSIZ2	RPSIZ1	RPSIZ0
		Write:								
		Reset:	X	X			X	X	X	X
				= Unimplemented			X = Indeterminate			

**Figure 9-21. Register Summary (Sheet 3 of 3)**

### 9.6.1 USB Address Register

Address:	\$0038							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	USBEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 9-22. USB Address Register (UADDR)**

#### USBEN — USB Module Enable

This read/write bit enables and disables the USB module and the USB pins. When USBEN is clear, the USB module will not respond to any traffic. Reset clears this bit.

- 1 = USB function enabled
- 0 = USB function disabled

**NOTE:** *The user must set this bit before the USB module will recognize USB reset signalling.*


#### UADD6–UADD0 — USB Function Address

These bits specify the USB address of the device. Reset clears these bits.

## 9.6.2 USB Interrupt Register 0

Address: \$0039

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXD0F	RXD0F	RSTF	SUSPND	TXD0IE	RXD0IE	0	0
Write:							TXD0FR	RXD0FR
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 9-23. USB Interrupt Register 0 (UIR0)**

### TXD0F — Endpoint 0 Data Transmit Flag

This read-only bit is set after the data stored in endpoint 0 transmit buffers has been sent and an ACK handshake packet from the host is received. Once the next set of data is ready in the transmit buffers, software must clear this flag by writing a logic 1 to the TXD0FR bit. To enable the next data packet transmission, TX0E also must be set. If the TXD0F bit is not cleared, a NAK handshake will be returned in the next IN transaction.

Reset clears this bit. Writing to TXD0F has no effect.

1 = Transmit on endpoint 0 has occurred.

0 = Transmit on endpoint 0 has not occurred.

### RXD0F — Endpoint 0 Data Receive Flag

This read-only bit is set after the USB module has received a data packet and responded with an ACK handshake packet. Software must clear this flag by writing a logic 1 to the RXD0FR bit after all of the received data has been read. Software also must set the RX0E bit to 1 to enable the next data packet reception. If the RXD0F bit is not cleared, a NAK handshake will be returned in the next OUT transaction.

Reset clears this bit. Writing to RXD0F has no effect.

1 = Receive on endpoint 0 has occurred.

0 = Receive on endpoint 0 has not occurred.

### RSTF — USB Reset Flag

This read-only bit is set when a valid reset signal state is detected on the D+ and D– lines. This reset detection will also generate an internal reset signal to reset the CPU and other peripherals including the USB module. This bit is cleared by writing a logic 1 to the RSTFR bit in the UCR2 register. This bit also is cleared by a POR reset.

**NOTE:** *The RSTF bit is included to maintain backward compatibility with the 68HC705JB2 USB implementation. The USB bit in the RSR register (see [8.8.2 Reset Status Register](#)) is also a USB reset indicator.*

### SUSPND — USB Suspend Flag

To save power, this read/write bit should be set by the software if a 3-ms constant idle state is detected on the USB bus. Setting this bit puts the transceiver and regulator into a power-saving mode. This bit is automatically cleared by hardware when the resume flag (RESUMF) is set.

### TXD0IE — Endpoint 0 Transmit Interrupt Enable

This read/write bit enables the transmit endpoint 0 to generate CPU interrupt requests when the TXD0F bit becomes set. Reset clears the TXD0IE bit.

- 1 = Transmit endpoint 0 can generate a CPU interrupt request.
- 0 = Transmit endpoint 0 cannot generate a CPU interrupt request.

### RXD0IE — Endpoint 0 Receive Interrupt Enable

This read/write bit enables the receive endpoint 0 to generate CPU interrupt requests when the RXD0F bit becomes set. Reset clears the RXD0IE bit.

- 1 = Receive endpoint 0 can generate a CPU interrupt request.
- 0 = Receive endpoint 0 cannot generate a CPU interrupt request.

### TXD0FR — Endpoint 0 Transmit Flag Reset

Writing a logic 1 to this write-only bit will clear the TXD0F bit if it is set. Writing a logic 0 to TXD0FR has no effect. Reset clears this bit.


### RXD0FR — Endpoint 0 Receive Flag Reset

Writing a logic 1 to this write-only bit will clear the RXD0F bit if it is set. Writing a logic 0 to RXD0FR has no effect. Reset clears this bit.

## 9.6.3 USB Interrupt Register 1

Address: \$003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXD1F	EOPF	RESUMF	0	TXD1IE	EOPIE	0	0
Write:				RESUMFR			TXD1FR	EOPFR
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 9-24. USB Interrupt Register 1 (UIR1)**

### TXD1F — Endpoint 1/Endpoint 2 Data Transmit Flag

This read-only bit is shared by endpoint 1 and endpoint 2. It is set after the data stored in the shared endpoint 1/endpoint 2 transmit buffer has been sent and an ACK handshake packet from the host is received. Once the next set of data is ready in the transmit buffers, software must clear this flag by writing a logic 1 to the TXD1FR bit. To enable the next data packet transmission, TX1E also must be set. If the TXD1F bit is not cleared, a NAK handshake will be returned in the next IN transaction.

Reset clears this bit. Writing to TXD1F has no effect.

1 = Transmit on endpoint 1 or endpoint 2 has occurred.

0 = Transmit on endpoint 1 or endpoint 2 has not occurred.

### EOPF — End-of-Packet Detect Flag

This read-only bit is set when a valid end-of-packet sequence is detected on the D+ and D– lines. Software must clear this flag by writing a logic 1 to the EOPFR bit.

Reset clears this bit. Writing to EOPF has no effect.

1 = End-of-packet sequence has been detected.

0 = End-of-packet sequence has not been detected.

#### RESUMF — Resume Flag

This read-only bit is set when USB bus activity is detected while the SUSPND bit is set. Software must clear this flag by writing a logic 1 to the RESUMFR bit.

Reset clears this bit. Writing a logic 0 to RESUMF has no effect.

1 = USB bus activity has been detected.

0 = No USB bus activity has been detected.

#### RESUMFR — Resume Flag Reset

Writing a logic 1 to this write-only bit will clear the RESUMF bit if it is set. Writing to RESUMFR has no effect. Reset clears this bit.

#### TXD1IE — Endpoint 1/Endpoint 2 Transmit Interrupt Enable

This read/write bit enables the USB to generate CPU interrupt requests when the shared transmit endpoint 1/endpoint 2 interrupt flag (TXD1F) bit becomes set. Reset clears the TXD1IE bit.

1 = Transmit endpoints 1 and 2 can generate a CPU interrupt request.

0 = Transmit endpoints 1 and 2 cannot generate a CPU interrupt request.

#### EOPIE — End-of-Packet Detect Interrupt Enable

This read/write bit enables the USB to generate CPU interrupt requests when the EOPF bit becomes set. Reset clears the EOPIE bit.

1 = End-of-packet sequence detection can generate a CPU interrupt request.

0 = End-of-packet sequence detection cannot generate a CPU interrupt request.

#### TXD1FR — Endpoint 1/Endpoint 2 Transmit Flag Reset

Writing a logic 1 to this write-only bit will clear the TXD1F bit if it is set. Writing a logic 0 to TXD1FR has no effect. Reset clears this bit.

#### EOPFR — End-of-Packet Flag Reset

Writing a logic 1 to this write-only bit will clear the EOPF bit if it is set. Writing a logic 0 to the EOPFR has no effect. Reset clears this bit.

## 9.6.4 USB Control Register 0

Address:	\$003B							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	T0SEQ	STALL0	TX0E	RX0E	TP0SIZ3	TP0SIZ2	TP0SIZ1	TP0SIZ0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 9-25. USB Control Register 0 (UCR0)**

### T0SEQ — Endpoint 0 Transmit Sequence Bit

This read/write bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction directed at endpoint 0. Toggling of this bit must be controlled by software. Reset clears this bit.

1 = DATA1 token active for next endpoint 0 transmit

0 = DATA0 token active for next endpoint 0 transmit

### STALL0 — Endpoint 0 Force Stall Bit

This read/write bit causes endpoint 0 to return a STALL handshake when polled by either an IN or OUT token by the USB host controller. The USB hardware clears this bit when a SETUP token is received. Reset clears this bit.

1 = Send STALL handshake.

0 = Default

### TX0E — Endpoint 0 Transmit Enable

This read/write bit enables a transmit to occur when the USB host controller sends an IN token to endpoint 0. Software should set this bit when data is ready to be transmitted. It must be cleared by software when no more endpoint 0 data needs to be transmitted.

If this bit is 0 or the TXD0F is set, the USB will respond with a NAK handshake to any endpoint 0 IN tokens. Reset clears this bit.

1 = Data is ready to be sent.

0 = Data is not ready. Respond with NAK.



#### RX0E — Endpoint 0 Receive Enable

This read/write bit enables a receive to occur when the USB host controller sends an OUT token to endpoint 0. Software should set this bit when data is ready to be received. It must be cleared by software when data cannot be received.

If this bit is 0 or the RXD0F is set, the USB will respond with a NAK handshake to any endpoint 0 OUT tokens. Reset clears this bit.

1 = Data is ready to be received.

0 = Not ready for data. Respond with NAK.

#### TP0SIZ3–TP0SIZ0 — Endpoint 0 Transmit Data Packet Size

These read/write bits store the number of transmit data bytes for the next IN token request for endpoint 0. These bits are cleared by reset.

## 9.6.5 USB Control Register 1

Address:	\$003C							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	T1SEQ	ENDADD	TX1E	FRESUM	TP1SIZ3	TP1SIZ2	TP1SIZ1	TP1SIZ0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 9-26. USB Control Register 1 (UCR1)**

### T1SEQ — Endpoint1/Endpoint 2 Transmit Sequence Bit

This read/write bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction directed to endpoint 1 or endpoint 2. Toggling of this bit must be controlled by software. Reset clears this bit.

1 = DATA1 token active for next endpoint 1/endpoint 2 transmit

0 = DATA0 token active for next endpoint 1/endpoint 2 transmit

### ENDADD — Endpoint Address Select

This read/write bit specifies whether the data inside the registers UE1D0–UE1D7 are used for endpoint 1 or endpoint 2.

If all the conditions for a successful endpoint 2 USB response to a host IN token are satisfied (TXD1F = 0, TX1E = 1, STALL2 = 0, and ENABLE2 = 1) except that the ENDADD bit is configured for endpoint 1, the USB responds with a NAK handshake packet.

1 = Data buffers used for endpoint 2

0 = Data buffers used for endpoint 1

#### TX1E — Endpoint 1/Endpoint 2 Transmit Enable

This read/write bit enables a transmit to occur when the USB host controller sends an IN token to endpoint 1 or endpoint 2. The appropriate endpoint enable bit, ENABLE1 or ENABLE2 bit in the UCR2 register, also should be set. Software should set the TX1E bit when data is ready to be transmitted. It must be cleared by software when no more data needs to be transmitted.

If this bit is 0 or the TXD1F is set, the USB will respond with a NAK handshake to any endpoint 1 or endpoint 2 directed IN tokens. Reset clears this bit.

1 = Data is ready to be sent.

0 = Data is not ready. Respond with NAK.

#### FRESUM — Force Resume

This read/write bit forces a resume state (K or non-idle state) onto the USB data lines to initiate a remote wakeup. Software should control the timing of the forced resume to be between 10 and 15 ms. Setting this bit will not cause the RESUMF bit to be set.

1 = Force data lines to K state

0 = Default


#### TP1SIZ3–TP1SIZ0 — Endpoint 1/Endpoint 2 Transmit Data Packet Size

These read/write bits store the number of transmit data bytes for the next IN token request for endpoint 1 or endpoint 2. These bits are cleared by reset.

## 9.6.6 USB Control Register 2

Address: \$0037

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	TX1ST	0	ENABLE2	ENABLE1	STALL2	STALL1
Write:	RSTFR	TX1STR						
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 9-27. USB Control Register 2 (UCR2)**

### RSTFR — Clear Reset Indicator Bit

Writing a logic 1 to this write-only bit will clear the RSTF bit in the UIR0 register if it is set. Writing a logic 0 to the RSTFR has no effect. Reset clears this bit.

### TX1STR — Clear Transmit First Flag

Writing a logic 1 to this write-only bit will clear the TX1ST bit if it is set. Writing a logic 0 to the TX1STR has no effect. Reset clears this bit.

### TX1ST — Transmit First Flag

This read-only bit is set if the endpoint 0 data transmit flag (TXD0F) is set when the USB control logic is setting the endpoint 0 data receive flag (RXD0F). In other words, if an unserviced endpoint 0 transmit flag is still set at the end of an endpoint 0 reception, then this bit will be set. This bit lets the firmware know that the endpoint 0 transmission happened before the endpoint 0 reception.

Reset clears this bit.

1 = IN transaction occurred before SETUP/OUT.

0 = IN transaction occurred after SETUP/OUT.

### ENABLE2 — Endpoint 2 Enable

This read/write bit enables endpoint 2 and allows the USB to respond to IN packets addressed to endpoint 2. Reset clears this bit.

1 = Endpoint 2 is enabled and can respond to an IN token.

0 = Endpoint 2 is disabled.

#### ENABLE1 — Endpoint 1 Enable

This read/write bit enables endpoint 1 and allows the USB to respond to IN packets addressed to endpoint 1. Reset clears this bit.

1 = Endpoint 1 is enabled and can respond to an IN token.

0 = Endpoint 1 is disabled.

#### STALL2 — Endpoint 2 Force Stall Bit

This read/write bit causes endpoint 2 to return a STALL handshake when polled by either an IN or OUT token by the USB host controller. Reset clears this bit.

1 = Send STALL handshake.

0 = Default

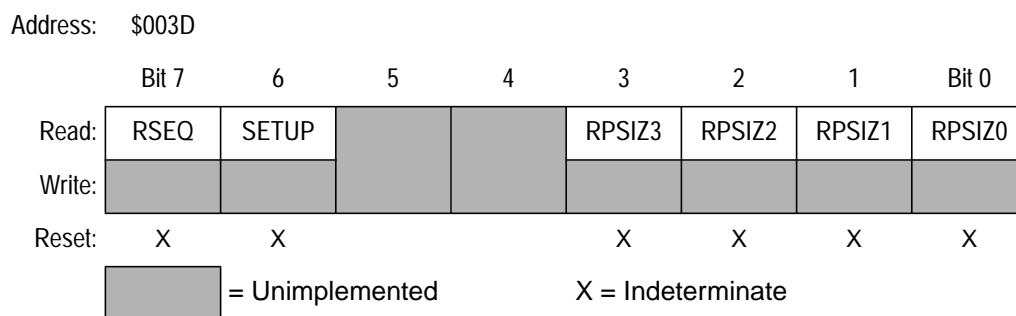
#### STALL1 — Endpoint 1 Force Stall Bit

This read/write bit causes endpoint 1 to return a STALL handshake when polled by either an IN or OUT token by the USB host controller. Reset clears this bit.

1 = Send STALL handshake.

0 = Default

## 9.6.7 USB Status Register



**Figure 9-28. USB Status Register (USR)**

### RSEQ — Endpoint 0 Receive Sequence Bit

This read-only bit indicates the type of data packet last received for endpoint 0 (DATA0 or DATA1).

1 = DATA1 token received in last endpoint 0 receive.

0 = DATA0 token received in last endpoint 0 receive.

### SETUP — SETUP Token Detect Bit

This read-only bit indicates that a valid SETUP token has been received.

1 = Last token received for endpoint 0 was a SETUP token.

0 = Last token received for endpoint 0 was not a SETUP token.

### RPSIZ3–RPSIZ0 — Endpoint 0 Receive Data Packet Size

These read-only bits store the number of data bytes received for the last OUT or SETUP transaction for endpoint 0. These bits are not affected by reset.

### 9.6.8 USB Endpoint 0 Data Registers

UE0D0 Address: \$0020

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
Reset:	X	X	X	X	X	X	X	X
↓								↓

UE0D7 Address: \$0027

Read:	UE0RD7	UE0RD6	UE0RD5	UE0RD4	UE0RD3	UE0RD2	UE0RD1	UE0RD0
Write:	UE0TD7	UE0TD6	UE0TD5	UE0TD4	UE0TD3	UE0TD2	UE0TD1	UE0TD0
Reset:	X	X	X	X	X	X	X	X

X = Indeterminate

**Figure 9-29. USB Endpoint 0 Data Register (UE0D0–UE0D7)**

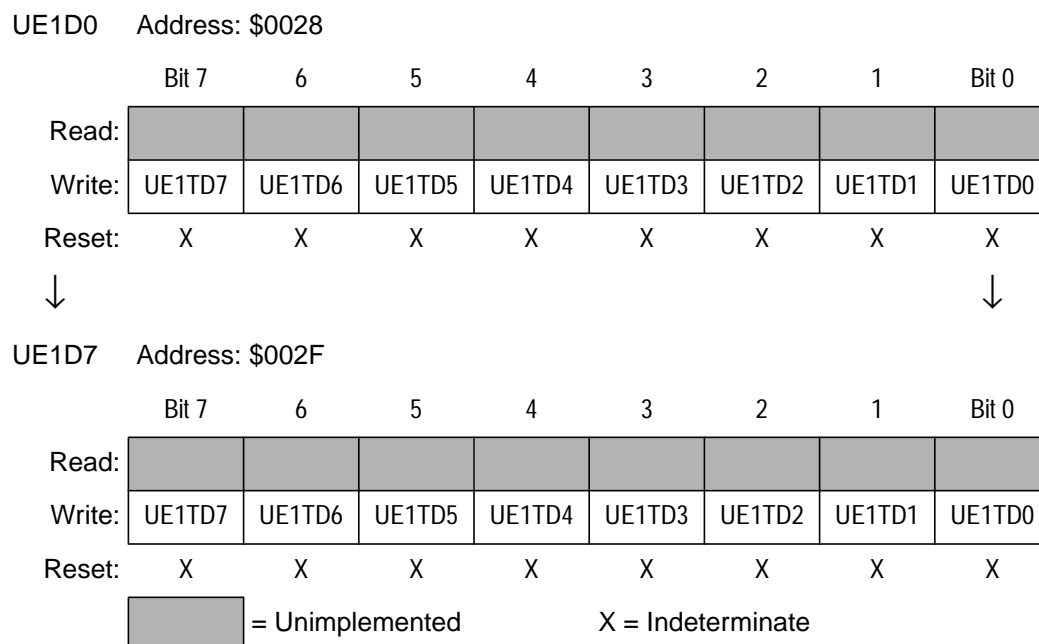
#### UE0RD7–UE0RD0 — Endpoint 0 Receive Data Buffer

These read-only bits are serially loaded with OUT token or SETUP token data directed at endpoint 0. The data is received over the USB's D+ and D– pins.

#### UE0TD7–UE0TD0 — Endpoint 0 Transmit Data Buffer

These write-only buffers are loaded by software with data to be sent on the USB bus on the next IN token directed at endpoint 0.

## 9.6.9 USB Endpoint 1/Endpoint 2 Data Registers



**Figure 9-30. USB Endpoint 1/Endpoint 2 Data Register (UE1D0–UE1D7)**

### UE1TD7–UE1TD0 — Endpoint 1/ Endpoint 2 Transmit Data Buffer

These write-only buffers are loaded by software with data to be sent on the USB bus on the next IN token directed at endpoint 1 or endpoint 2. These buffers are shared by endpoints 1 and 2 and depend on proper configuration of the ENDADD bit.



## 9.7 USB Interrupts

The USB module is capable of generating interrupts and causing the CPU to execute the USB interrupt service routine. The three types of USB interrupts are:

- End-of-transaction interrupts — signify either a completed receive or transmit transaction
- Resume interrupts — signify that the USB bus is reactivated after having been suspended
- End-of-packet interrupts — signify that a low-speed end-of-packet signal was detected

All USB interrupts share the same interrupt vector. Firmware is responsible for determining which interrupt is active.

### 9.7.1 USB End-of-Transaction Interrupt

There are three possible end-of-transaction interrupts:

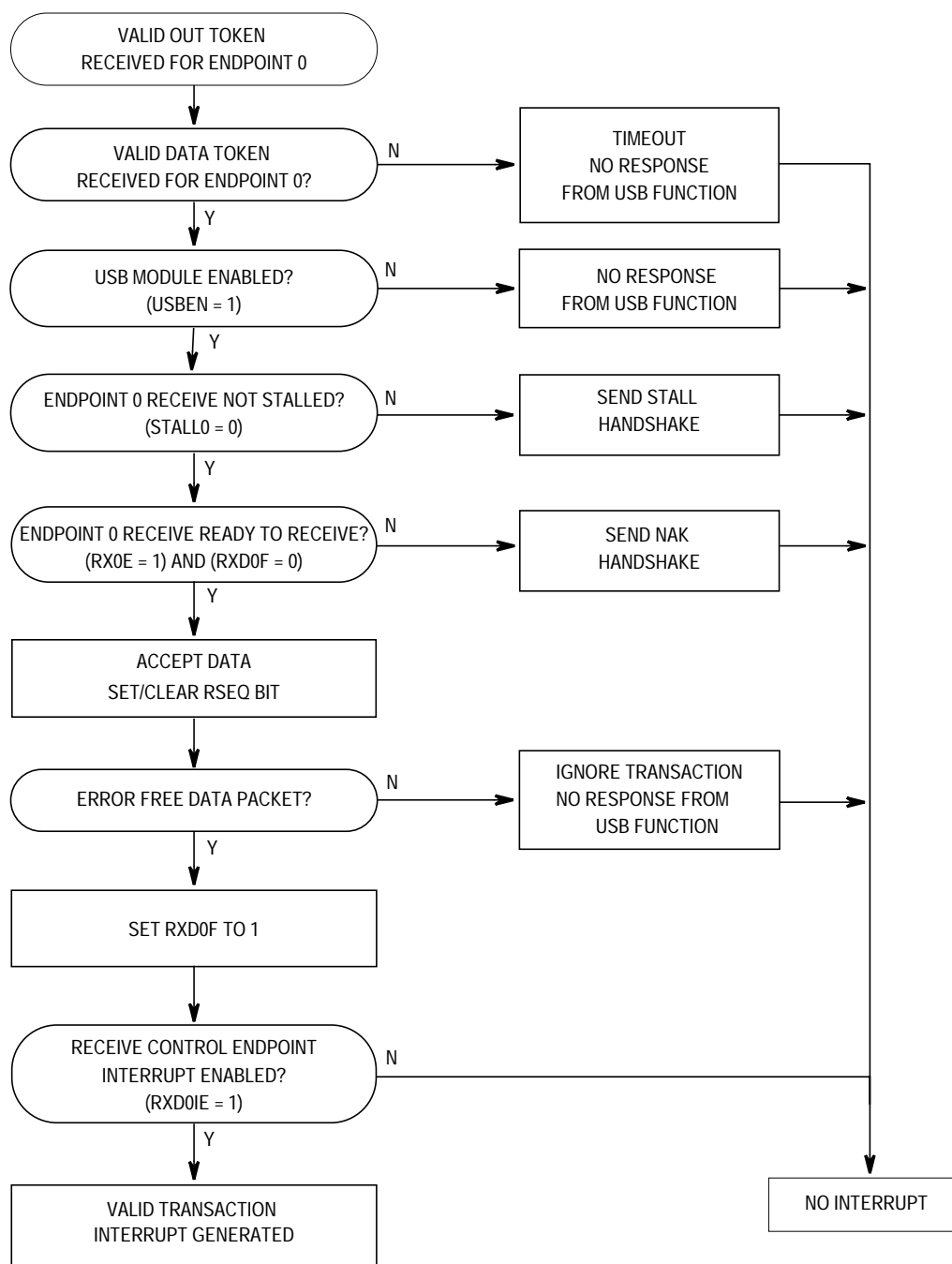
- Endpoint 0 receive
- Endpoint 0 transmit
- Shared endpoint 1 or endpoint 2 transmit

End-of-transaction interrupts occur as detailed in the following sections.

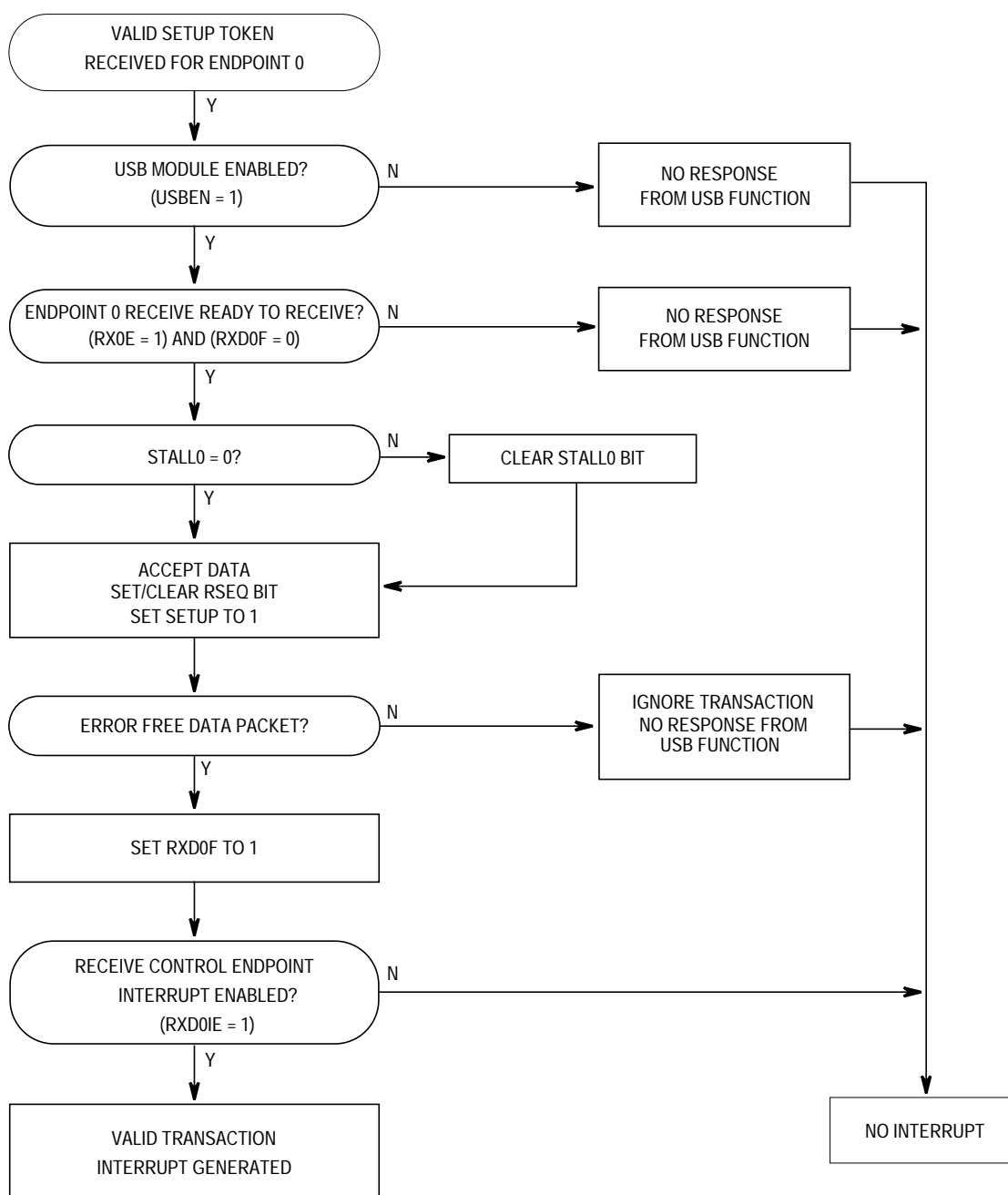
#### 9.7.1.1 Receive Control Endpoint 0

For a control OUT transaction directed at endpoint 0, the USB module will generate an interrupt by setting the RXD0F flag in the UIR0 register. The conditions necessary for the interrupt to occur are shown in the flowchart in [Figure 9-31](#).

SETUP transactions cannot be stalled by the USB function. A SETUP received by a control endpoint will clear the STALL0 bit if it is set. The conditions for receiving a SETUP interrupt are shown in [Figure 9-32](#).



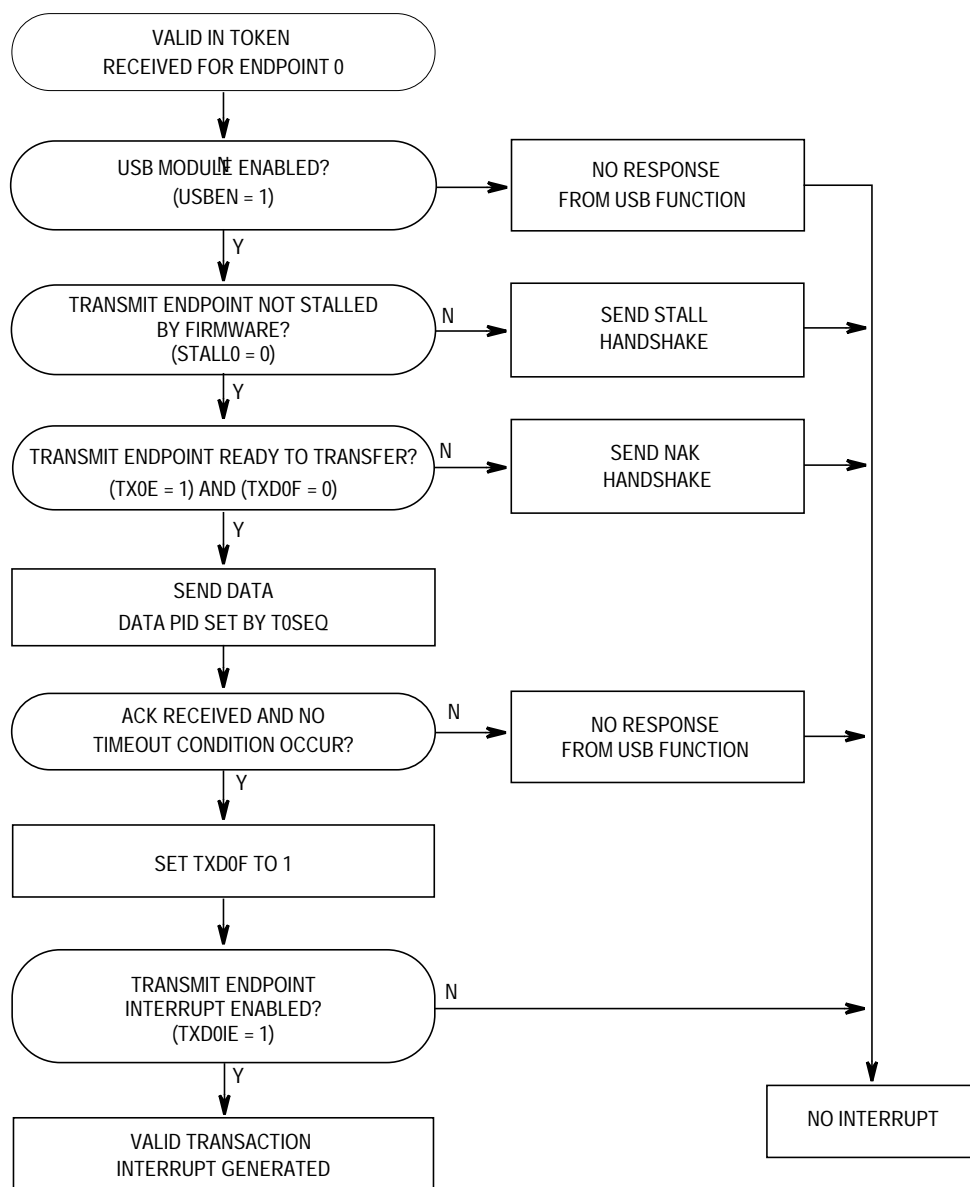
**Figure 9-31. OUT Token Data Flow for Receive Endpoint 0**



**Figure 9-32. SETUP Token Data Flow for Receive Endpoint 0**

## 9.7.1.2 Transmit Control Endpoint 0

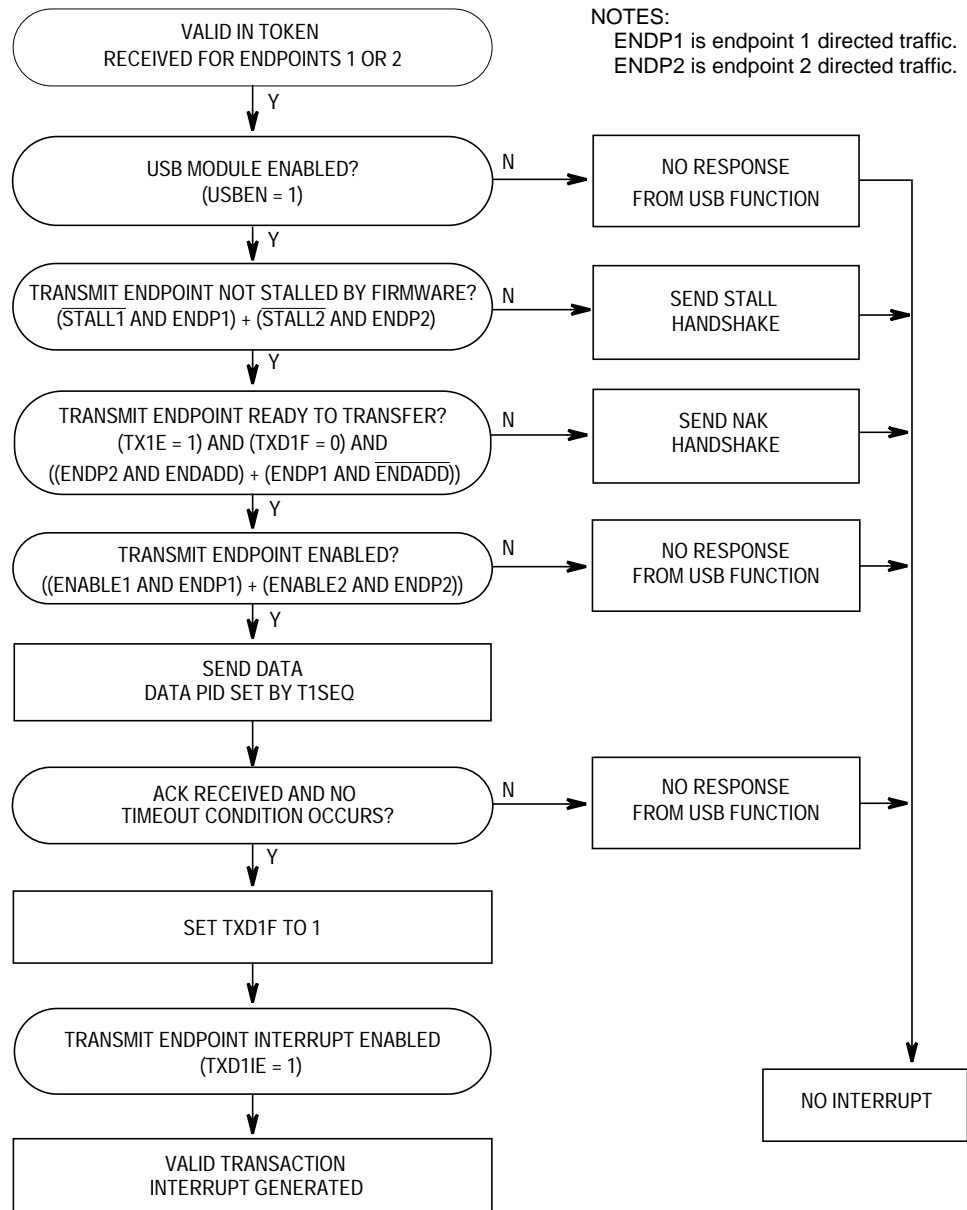
For a control IN transaction directed at endpoint 0, the USB module will generate an interrupt by setting the TXD0F flag in the UIR0 register. The conditions necessary for the interrupt to occur are shown in the flowchart in [Figure 9-33](#).



**Figure 9-33. IN Token Data Flow for Transmit Endpoint 0**

### 9.7.1.3 Transmit Endpoint 1 and Transmit Endpoint 2

Transmit endpoints 1 and 2 share their interrupt flag. For an IN transaction directed at endpoint 1 or 2, the USB module will generate an interrupt by setting the TXD1F flag in the UIR1 register. The conditions necessary for the interrupt to occur are shown in **Figure 9-34**.



**Figure 9-34. IN Token Data Flow for Transmit Endpoint 1/Endpoint 2**

### 9.7.2 Resume Interrupt

The USB module will generate a CPU interrupt if low-speed bus activity is detected after entering the suspend state. A transition of the USB data lines to the non-idle state (K state) while in the suspend mode will set the RESUMF flag in the UIR1 register. There is no interrupt enable bit for this interrupt source and an interrupt will be executed if the I bit in the CCR is cleared. A resume interrupt can only occur while the MC68HC08KL8 is in the suspend mode.

### 9.7.3 End-of-Packet Interrupt

The USB module can generate a USB interrupt upon detection of an end-of-packet signal for low-speed devices. Upon detection of an end-of-packet signal, the USB module sets the EOPF bit and will generate a CPU interrupt if the EOPIE bit in the UIR1 register is set.

## Section 10. Monitor ROM (MON)

### 10.1 Contents

10.2	Introduction .....	151
10.3	Features .....	152
10.4	Functional Description .....	152
10.4.1	Entering Monitor Mode .....	154
10.4.2	Data Format .....	156
10.4.3	Break Signal .....	156
10.4.4	Baud Rate .....	157
10.4.5	Commands .....	157
10.5	Security .....	163

### 10.2 Introduction

This section describes the monitor ROM. The monitor ROM allows complete testing of the MCU through a single-wire interface with a host computer.

### 10.3 Features

Features of the monitor ROM include:

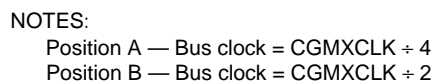
- Normal User-Mode Pin Functionality
- One Pin Dedicated to Serial Communication between Monitor ROM and Host Computer
- Standard Mark/Space Non-Return-to-Zero (NRZ) Communication with Host Computer
- Execution of Code in RAM or ROM
- ROM Security

### 10.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 10-1** shows an example circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute host-computer code in RAM while all MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.





### Figure 10-1. Monitor Mode Circuit

## 10.4.1 Entering Monitor Mode

**Table 10-1** shows the pin conditions for entering monitor mode.

**Table 10-1. Monitor Mode Entry**

$\overline{\text{IRQ1}}$ Pin	PTA7 Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	CGMOUT	Bus Frequency
$V_{\text{TST}}$	0	1	0	1	1	$\frac{\text{CGMXCLK}}{2}$	$\frac{\text{CGMOUT}}{2}$
					0	CGMXCLK	

If PTC3 is low upon monitor mode entry, CGMOUT is equal to the crystal frequency. The bus frequency in this case is a divide-by-two of the input clock. If PTC3 is high upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock.

**NOTE:** Holding the PTC3 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.

Enter monitor mode with the pin configuration shown above by pulling  $\overline{\text{RST}}$  low and then high. The rising edge of  $\overline{\text{RST}}$  latches monitor mode. Once monitor mode is latched, the values on the specified pins can change.

**NOTE:** The PTA7 pin must remain at logic 0 for 24 bus cycles after the  $\overline{\text{RST}}$  pin goes high.

Once out of reset, the MCU waits for the host to send eight security bytes. (See [10.5 Security](#).) After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host, indicating that it is ready to receive a command.

In monitor mode, the MCU uses different vectors for reset, SWI, and break interrupt than those for user mode. The alternate vectors are in the \$FE page instead of the \$FF page and allow code execution from the internal monitor firmware instead of user code.

The COP module is disabled in monitor mode as long as  $V_{TST}$  is applied to either the  $\overline{IRQ}$  pin or the  $\overline{RST}$  pin.

**Table 10-2** summarizes the differences between user mode and monitor mode.

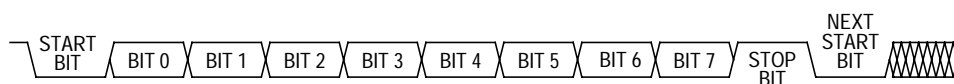
**Table 10-2. Mode Differences**

Modes	Functions						
	COP	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	Enabled	\$FFFE	\$FFFF	\$FFFC	\$FFFD	\$FFFC	\$FFFD
Monitor	Disabled <sup>(1)</sup>	\$FEFE	\$FEFF	\$FEFC	\$FEFD	\$FEFC	\$FEFD

1. If the high voltage ( $V_{TST}$ ) is removed from the  $\overline{IRQ1}$  pin or the  $\overline{RST}$  pin, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the mask option register.

## 10.4.2 Data Format

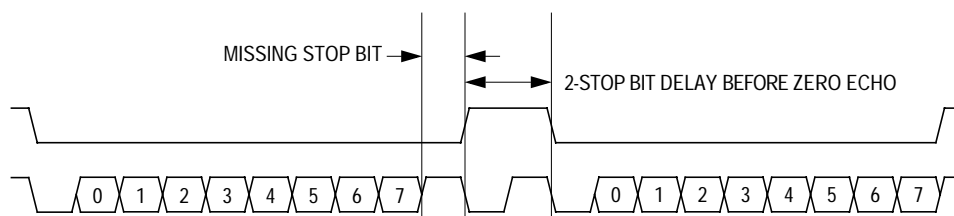
Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.



**Figure 10-2. Monitor Data Format**

## 10.4.3 Break Signal

A start bit (logic 0) followed by nine logic 0 bits is a break signal. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits and then echoes back the break signal.



**Figure 10-3. Break Transaction**

#### 10.4.4 Baud Rate

The communication baud rate is controlled by the crystal frequency and the state of the PTC3 pin upon entry into monitor mode. When PTC3 is high, the divide by ratio is 1024. If the PTC3 pin is at logic 0 upon entry into monitor mode, the divide by ratio is 512. [Table 10-3](#) lists crystal frequencies required to achieve standard baud rates. Other standard baud rates can be accomplished using higher crystal frequencies.

**Table 10-3. Monitor Baud Rate Selection**

Crystal Frequency (MHz)	PTC3 Pin	Baud Rate
4.9152	0	9600
4.9152	1	4800

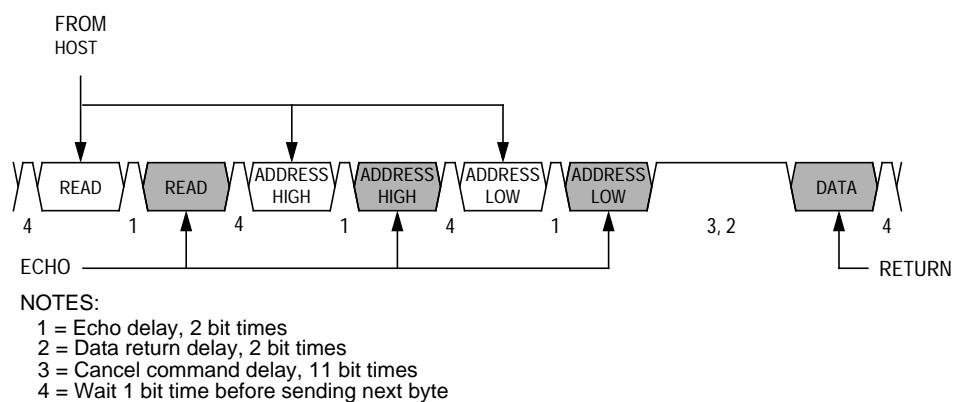
#### 10.4.5 Commands

The monitor ROM firmware uses these commands:

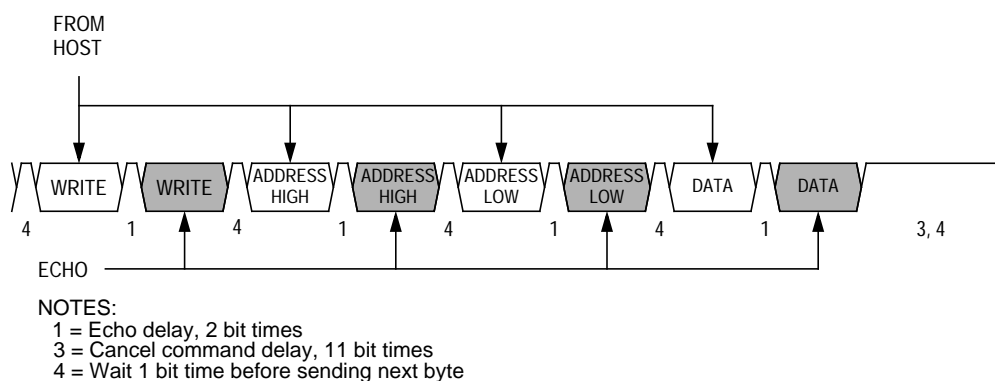
- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

**NOTE:** *Wait one bit time after each echo before sending the next byte.*



**Figure 10-4. Read Transaction**



**Figure 10-5. Write Transaction**

A brief description of each monitor mode command is shown in [Table 10-4](#), [Table 10-5](#), [Table 10-6](#), and [Table 10-7](#).

**Table 10-4. READ (Read Memory) Command**

<b>Description</b>	Read Byte from Memory
<b>Operand</b>	2-Byte Address in High Byte:Low Byte Order
<b>Data Returned</b>	Returns Contents of Specified Address
<b>Opcode</b>	\$4A
<p style="text-align: center;"><b>Command Sequence</b></p>	

**Table 10-5. WRITE (Write Memory) Command**

<b>Description</b>	Write Byte to Memory
<b>Operand</b>	2-Byte Address in High Byte:Low Byte Order; Low Byte Followed by Data Byte
<b>Data Returned</b>	None
<b>Opcode</b>	\$49
<p style="text-align: center;"><b>Command Sequence</b></p>	

**Table 10-6. IREAD (Indexed Read) Command**

<b>Description</b>	Read Next 2 Bytes in Memory from Last Address Accessed
<b>Operand</b>	2-Byte Address in High Byte:Low Byte Order
<b>Data Returned</b>	Returns Contents of Next Two Addresses
<b>Opcode</b>	\$1A
<p style="text-align: center;"><b>Command Sequence</b></p>	

**Table 10-7. IWRITE (Indexed Write) Command**

<b>Description</b>	Write to Last Address Accessed + 1
<b>Operand</b>	Single Data Byte
<b>Data Returned</b>	None
<b>Opcode</b>	\$19
<p style="text-align: center;"><b>Command Sequence</b></p>	



A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.

**Table 10-8. READSP (Read Stack Pointer) Command**

<b>Description</b>	Reads Stack Pointer
<b>Operand</b>	None
<b>Data Returned</b>	Returns Incremented Stack Pointer Value (SP + 1) in High Byte:Low Byte Order
<b>Opcode</b>	\$0C
<p style="text-align: center;"><b>Command Sequence</b></p>	

**Table 10-9. RUN (Run User Program) Command**

<b>Description</b>	Executes PULH and RTI Instructions
<b>Operand</b>	None
<b>Data Returned</b>	None
<b>Opcode</b>	\$28
<p style="text-align: center;"><b>Command Sequence</b></p>	

The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value,  $SP + 1$ . The high and low bytes of the program counter are at addresses  $SP + 5$  and  $SP + 6$ .

	SP
HIGH BYTE OF INDEX REGISTER	SP + 1
CONDITION CODE REGISTER	SP + 2
ACCUMULATOR	SP + 3
LOW BYTE OF INDEX REGISTER	SP + 4
HIGH BYTE OF PROGRAM COUNTER	SP + 5
LOW BYTE OF PROGRAM COUNTER	SP + 6
	SP + 7

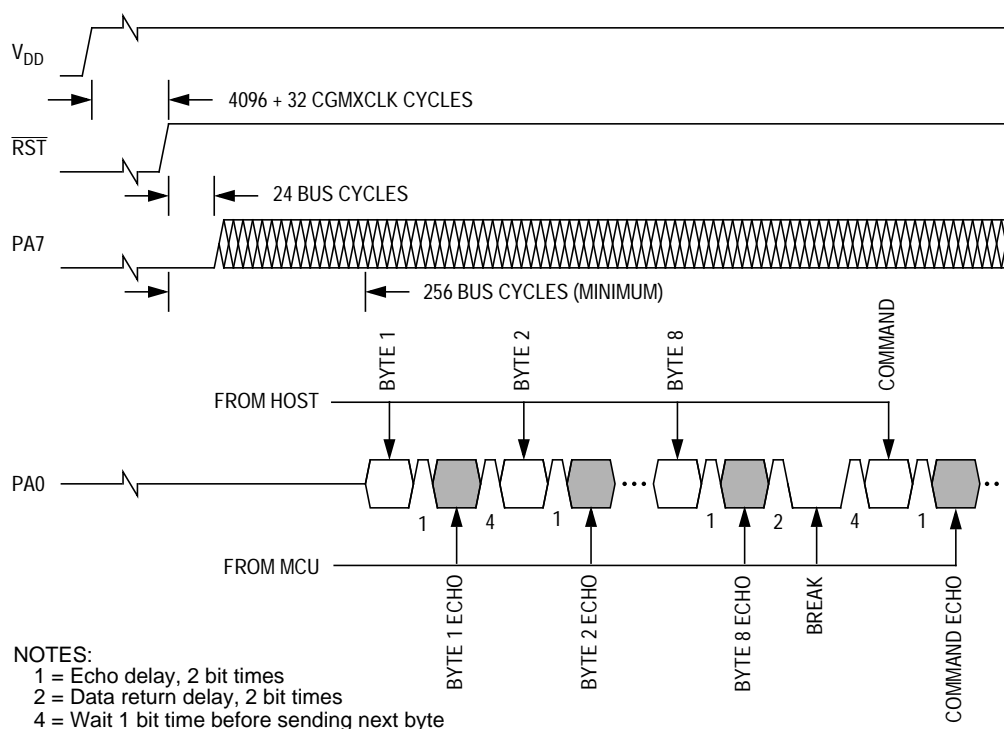
**Figure 10-6. Stack Pointer at Monitor Mode Entry**

## 10.5 Security

A security feature<sup>1</sup> discourages unauthorized reading of ROM locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations \$FFF6–\$FFFD. Locations \$FFF6–\$FFFD contain user-defined data.

**NOTE:** Do not leave locations \$FFF6–\$FFFD blank. For security reasons, program locations \$FFF6–\$FFFD even if they are not used for vectors.

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PA0.



**Figure 10-7. Monitor Mode Entry Timing**

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the ROM data difficult for unauthorized users.

If the received bytes match those at locations \$FFF6–\$FFFD, the host bypasses the security feature and can read all ROM locations and execute code from ROM. Security remains bypassed until a power-on reset occurs. After the host bypasses security, any reset other than a power-on reset requires the host to send another eight bytes. If the reset was not a power-on reset, security remains bypassed regardless of the data that the host sends.

If the received bytes do not match the data at locations \$FFF6–\$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading ROM locations returns undefined data, and trying to execute code from ROM causes an illegal address reset. After the host fails to bypass security, any reset other than a power-on reset causes an endless loop of illegal address resets.

After receiving the eight security bytes from the host, the MCU transmits a break character signalling that it is ready to receive a command.

**NOTE:** *The MCU does not transmit a break character until after the host sends the eight security bytes.*

## Section 11. Timer Interface Module (TIM)

### 11.1 Contents

11.2	Introduction	166
11.3	Features	166
11.4	Functional Description	166
11.4.1	TIM Counter Prescaler	170
11.4.2	Input Capture	170
11.4.3	Output Compare	170
11.4.3.1	Unbuffered Output Compare	170
11.4.3.2	Buffered Output Compare	171
11.4.4	Pulse Width Modulation (PWM)	172
11.4.4.1	Unbuffered PWM Signal Generation	173
11.4.4.2	Buffered PWM Signal Generation	174
11.4.4.3	PWM Initialization	174
11.5	Interrupts	176
11.6	Wait Mode	176
11.7	TIM During Break Interrupts	177
11.8	I/O Signals	177
11.8.1	TIM Clock Pin (PTE0/TCLK)	178
11.8.2	TIM Channel I/O Pins (PTE1/TCH0:PTE2/TCH1)	178
11.9	I/O Registers	178
11.9.1	TIM Status and Control Register	179
11.9.2	TIM Counter Registers	181
11.9.3	TIM Counter Modulo Registers	182
11.9.4	TIM Channel Status and Control Registers	183
11.9.5	TIM Channel Registers	187

### 11.2 Introduction

This section describes the timer interface module (TIM2, Version B). The TIM is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions.

**Figure 11-1** is a block diagram of the TIM.

**NOTE:** *The TIM module is available only on the 52-pin QFP package.*

### 11.3 Features

Features of the TIM include:

- Two Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIM Clock Input
  - 7-Frequency Internal Bus Clock Prescaler Selection
  - External TIM Clock Input (Bus Frequency  $\div 2$  Maximum)
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIM Counter Stop and Reset Bits
- Modular Architecture Expandable to Eight Channels

### 11.4 Functional Description

**Figure 11-1** shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

The two TIM channels are programmable independently as input capture or output compare channels.

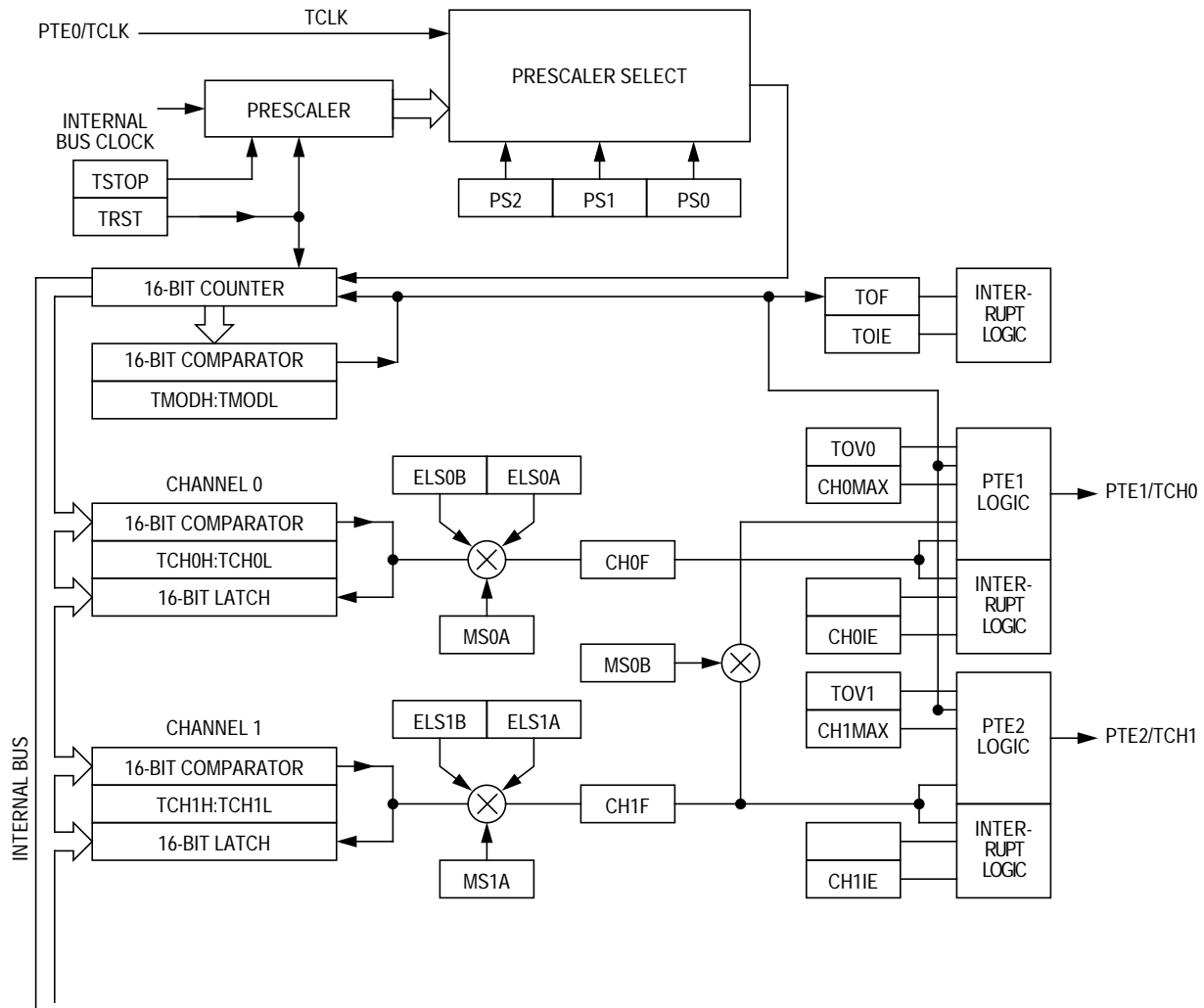



Figure 11-1. TIM Block Diagram

# Timer Interface Module (TIM)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0010	TIM Status and Control Register (TSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
\$0012	TIM Counter Register High (TCNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0013	TIM Counter Register Low (TCNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0014	TIM Counter Modulo Register High (TMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
\$0015	TIM Counter Modulo Register Low (TMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
\$0016	TIM Channel 0 Status and Control Register (TSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
\$0017	TIM Channel 0 Register High (TCH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after Reset							
\$0018	TIM Channel 0 Register Low (TCH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							

 = Unimplemented

**Figure 11-2. TIM I/O Register Summary**



Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0	
\$0019	TIM Channel 1 Status and Control Register (TSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX	
		Write:	0								
		Reset:	0	0	0	0	0	0	0	0	
\$001A	TIM Channel 1 Register High (TCH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8	
		Write:									
		Reset:	Indeterminate after Reset								
\$001B	TIM Channel 1 Register Low (TCH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0	
		Write:									
		Reset:	Indeterminate after Reset								
				= Unimplemented							

**Figure 11-2. TIM I/O Register Summary**

### 11.4.1 TIM Counter Prescaler

The TIM clock source can be one of the seven prescaler outputs or the TIM clock pin, PTE0/TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM status and control register (TSC) select the TIM clock source.

### 11.4.2 Input Capture

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the TIM counter into the TIM channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input captures can generate TIM CPU interrupt requests.

### 11.4.3 Output Compare

With the output compare function, the TIM can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIM can set, clear, or toggle the channel pin. Output compares can generate TIM CPU interrupt requests.

#### 11.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in [11.4.3 Output Compare](#). The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIM overflow interrupt routine to write a new, smaller output

compare value may cause the compare to be missed. The TIM may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable channel x TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

#### 11.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE1/TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

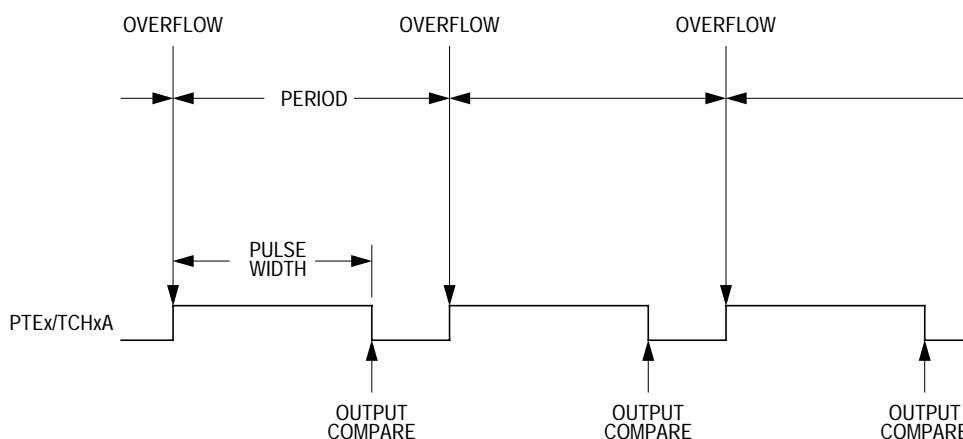
Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the PTE1/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE2/TCH1, is available as a general-purpose I/O pin.

**NOTE:** *In buffered output compare operation, do not write new output compare values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered output compares.*

## 11.4.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM can generate a PWM signal. The value in the TIM counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM counter modulo registers. The time between overflows is the period of the PWM signal.

As [Figure 11-3](#) shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIM to set the pin if the state of the PWM pulse is logic 0.



**Figure 11-3. PWM Period and Pulse Width**

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000 (see [11.9.1 TIM Status and Control Register](#)).

The value in the TIM channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256

increments. Writing 00080 (128) to the TIM channel registers produces a duty cycle of 128/256 or 50%.

#### 11.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in [11.4.4 Pulse Width Modulation \(PWM\)](#). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable channel x TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output*

*compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

### 11.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE1/TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the PTE1/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE2/TCH1, is available as a general-purpose I/O pin.

**NOTE:** *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

### 11.4.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use this initialization procedure:

1. In the TIM status and control register (TSC):
  - a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
  - b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.

4. In TIM channel x status and control register (TSCx):
  - a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See [Table 11-2](#).)
  - b. Write 1 to the toggle-on-overflow bit, TOVx.
  - c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See [Table 11-2](#).)

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

5. In the TIM status control register (TSC), clear the TIM stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. TIM status control register 0 (TSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. (See [11.9.4 TIM Channel Status and Control Registers](#).)

### 11.5 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to \$0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH1F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM CPU interrupt requests are enabled when CHxIE = 1. CHxF and CHxIE are in the TIM channel x status and control register.

### 11.6 Wait Mode

The WAIT instruction puts the MCU in low-power standby mode.

The TIM remains active after the execution of a WAIT instruction. In wait mode the TIM registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIM can bring the MCU out of wait mode.

If TIM functions are not required during wait mode, reduce power consumption by stopping the TIM before executing the WAIT instruction.



## 11.7 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See [8.8.3 Break Flag Control Register](#).)

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

## 11.8 I/O Signals

Port E shares three of its pins with the TIM. PTE0/TCLK is an external clock input to the TIM prescaler. The two TIM channel I/O pins are PTE1/TCH0 and PTE2/TCH1.

## 11.8.1 TIM Clock Pin (PTE0/TCLK)

PTE0/TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTE0/TCLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See [11.9.1 TIM Status and Control Register](#).) The minimum TCLK pulse width,  $TCLK_{L_{MIN}}$  or  $TCLK_{H_{MIN}}$ , is:

$$\frac{1}{\text{bus frequency}} + t_{SU}$$

The maximum TCLK frequency is:

$$\text{bus frequency} \div 2$$

PTE0/TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the PTE0/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE0 bit in data direction register E.

## 11.8.2 TIM Channel I/O Pins (PTE1/TCH0:PTE2/TCH1)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. **PTE1/TCH0** can be configured as buffered output compare or buffered PWM pins.

## 11.9 I/O Registers

These I/O registers control and monitor TIM operation:

- TIM status and control register (TSC)
- TIM control registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0 and TSC1)
- TIM channel registers (TCH0H:TCH0L and TCH1H:TCH1L)


### 11.9.1 TIM Status and Control Register

The TIM status and control register:

- Enables TIM overflow interrupts
- Flags TIM overflows
- Stops the TIM counter
- Resets the TIM counter
- Prescales the TIM counter clock

Address: \$0010

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

 = Unimplemented

**Figure 11-4. TIM Status and Control Register (TSC)**

#### TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter resets to \$0000 after reaching the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a logic 0 to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIM counter has reached modulo value.

0 = TIM counter has not reached modulo value.

#### TOIE — TIM Overflow Interrupt Enable Bit

This read/write bit enables TIM overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIM overflow interrupts enabled

0 = TIM overflow interrupts disabled

## TSTOP — TIM Stop Bit

This read/write bit stops the TIM counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIM counter until software clears the TSTOP bit.

1 = TIM counter stopped

0 = TIM counter active

**NOTE:** *Do not set the TSTOP bit before entering wait mode if the TIM is required to exit wait mode.*

## TRST — TIM Reset Bit

Setting this write-only bit resets the TIM counter and the TIM prescaler. Setting TRST has no effect on any other registers. Counting resumes from \$0000. TRST is cleared automatically after the TIM counter is reset and always reads as logic 0. Reset clears the TRST bit.

1 = Prescaler and TIM counter cleared

0 = No effect

**NOTE:** *Setting the TSTOP and TRST bits simultaneously stops the TIM counter at a value of \$0000.*

## PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTE0/TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as [Table 11-1](#) shows. Reset clears the PS[2:0] bits.

**Table 11-1. Prescaler Selection**

PS[2:0]	TIM Clock Source
000	Internal Bus Clock ÷ 1
001	Internal Bus Clock ÷ 2
010	Internal Bus Clock ÷ 4
011	Internal Bus Clock ÷ 8
100	Internal Bus Clock ÷ 16
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	PTE0/TCLK

### 11.9.2 TIM Counter Registers

The two read-only TIM counter registers contain the high and low bytes of the value in the TIM counter. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL) into a buffer. Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. Reset clears the TIM counter registers. Setting the TIM reset bit (TRST) also clears the TIM counter registers.


**NOTE:** *If you read TCNTH during a break interrupt, be sure to unlatch TCNTL by reading TCNTL before exiting the break interrupt. Otherwise, TCNTL retains the value latched during the break.*

TCNTH Address: \$0012

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

TCNTL Address: \$0013

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 11-5. TIM Counter Registers (TCNTH:TCNTL)**

## 11.9.3 TIM Counter Modulo Registers

The read/write TIM modulo registers contain the modulo value for the TIM counter. When the TIM counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIM counter resumes counting from \$0000 at the next clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIM counter modulo registers.

TMODH Address: \$0014

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	1	1	1	1	1	1	1	1

TMODL Address: \$0015

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	1	1	1	1	1	1	1	1

**Figure 11-6. TIM Counter Modulo Registers (TMODH:TMODL)**

**NOTE:** Reset the TIM counter before writing to the TIM counter modulo registers.

### 11.9.4 TIM Channel Status and Control Registers

Each of the TIM channel status and control registers:


- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIM overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

TSC0 Address: \$0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC1 Address: \$0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 11-7. TIM Channel Status and Control Registers  
(TSC0:TSC1)**

### CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIM counter registers matches the value in the TIM channel x registers.

When TIM CPU interrupt requests are enabled (CHxIE = 1), clear CHxF by reading the TIM channel x status and control register with CHxF set and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x

0 = No input capture or output compare on channel x

### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupt service requests on channel x. Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled

0 = Channel x CPU interrupt requests disabled

### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM channel 0 status and control register.

Setting MS0B disables the channel 1 status and control register and reverts TCH1 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled

0 = Buffered output compare/PWM operation disabled

### MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation.

See [Table 11-2](#).

1 = Unbuffered output compare/PWM operation

0 = Input capture operation



When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. (See [Table 11-2](#).) Reset clears the MSxA bit.

1 = Initial output level low

0 = Initial output level high

**NOTE:** Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM status and control register (TSC).

#### ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to port E, and pin PTE<sub>x</sub>/TCH<sub>x</sub> is available as a general-purpose I/O pin. [Table 11-2](#) shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 11-2. Mode, Edge, and Level Selection**

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
X0	00	Output Preset	Pin under Port Control; Initial Output Level High
X1	00		Pin under Port Control; Initial Output Level Low
00	01	Input Capture	Capture on Rising Edge Only
00	10		Capture on Falling Edge Only
00	11		Capture on Rising or Falling Edge
01	01	Output Compare or PWM	Toggle Output on Compare
01	10		Clear Output on Compare
01	11		Set Output on Compare
1X	01	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1X	10		Clear Output on Compare
1X	11		Set Output on Compare

**NOTE:** Before enabling a TIM channel register for input capture operation, make sure that the PTE<sub>x</sub>/TCH<sub>x</sub> pin is stable for at least two bus clocks.

## TOV<sub>x</sub> — Toggle-On-Overflow Bit

When channel *x* is an output compare channel, this read/write bit controls the behavior of the channel *x* output when the TIM counter overflows. When channel *x* is an input capture channel, TOV<sub>x</sub> has no effect. Reset clears the TOV<sub>x</sub> bit.

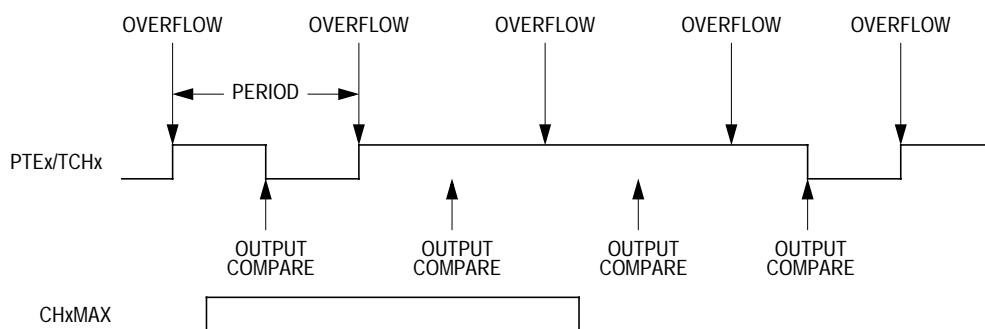
1 = Channel *x* pin toggles on TIM counter overflow.

0 = Channel *x* pin does not toggle on TIM counter overflow.

**NOTE:** When TOV<sub>x</sub> is set, a TIM counter overflow takes precedence over a channel *x* output compare if both occur at the same time.

## CH<sub>x</sub>MAX — Channel *x* Maximum Duty Cycle Bit

When the TOV<sub>x</sub> bit is at logic 0, setting the CH<sub>x</sub>MAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As [Figure 11-8](#) shows, the CH<sub>x</sub>MAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CH<sub>x</sub>MAX is cleared.



**Figure 11-8. CH<sub>x</sub>MAX Latency**

### 11.9.5 TIM Channel Registers

These read/write registers contain the captured TIM counter value of the input capture function or the output compare value of the output compare function. The state of the TIM channel registers after reset is unknown.

In input capture mode ( $MSxB:MSxA = 0:0$ ), reading the high byte of the TIM channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode ( $MSxB:MSxA \neq 0:0$ ), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

TCH0H	Address:	\$0017								
			Bit 7	6	5	4	3	2	1	Bit 0
Read:			Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:										
Reset:			Indeterminate after Reset							

TCH0L	Address:	\$0018								
			Bit 7	6	5	4	3	2	1	Bit 0
Read:			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:										
Reset:			Indeterminate after Reset							

TCH1H	Address:	\$001A								
			Bit 7	6	5	4	3	2	1	Bit 0
Read:			Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:										
Reset:			Indeterminate after Reset							

TCH1L	Address:	\$001B								
			Bit 7	6	5	4	3	2	1	Bit 0
Read:			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:										
Reset:			Indeterminate after Reset							

**Figure 11-9. TIM Channel Registers (TCH0H/L:TCH1H/L)**



## Section 12. Input/Output Ports (I/O)

### 12.1 Contents

12.2	Introduction . . . . .	190
12.3	Port A . . . . .	192
12.3.1	Port A Data Register . . . . .	192
12.3.2	Data Direction Register A . . . . .	193
12.4	Port B . . . . .	195
12.4.1	Port B Data Register . . . . .	195
12.4.2	Data Direction Register B . . . . .	196
12.5	Port C . . . . .	198
12.5.1	Port C Data Register . . . . .	198
12.5.2	Data Direction Register C . . . . .	199
12.6	Port D . . . . .	201
12.6.1	Port D Data Register . . . . .	201
12.6.2	Data Direction Register D . . . . .	202
12.7	Port E . . . . .	204
12.7.1	Port E Data Register . . . . .	204
12.7.2	Data Direction Register E . . . . .	206
12.8	Port Options . . . . .	208
12.8.1	Port Option Control Register . . . . .	208


## 12.2 Introduction

Thirty-nine bidirectional input-output (I/O) pins form five parallel ports. All I/O pins are programmable as inputs or outputs.

**NOTE:** Connect any unused I/O pins to an appropriate logic level, either  $V_{DD}$  or  $V_{SS}$ . Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0000	Port A Data Register (PTA)	Read:								
		Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by Reset							
\$0001	Port B Data Register (PTB)	Read:								
		Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by Reset							
\$0002	Port C Data Register (PTC)	Read:								
		Write:	PTC7 <sup>(1)</sup>	PTC6 <sup>(1)</sup>	PTC5 <sup>(1)</sup>	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by Reset							
\$0003	Port D Data Register (PTD)	Read:								
		Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by Reset							
\$0004	Data Direction Register A (DDRA)	Read:								
		Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
\$0005	Data Direction Register B (DDRB)	Read:								
		Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0


Note 1. Only available on 52-pin QFP.

 = Unimplemented

**Figure 12-1. I/O Port Register Summary**

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0006	Data Direction Register C (DDRC)	Read:	DDRC7 <sup>(1)</sup>	DDRC6 <sup>(1)</sup>	DDRC5 <sup>(1)</sup>	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0007	Data Direction Register D (DDR D)	Read:	DDR D7	DDR D6	DDR D5	DDR D4	DDR D3	DDR D2	DDR D1	DDR D0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0008	Port E Data Register <sup>(1)</sup> (PTE)	Read:	0	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by Reset							
\$000C	Data Direction Register E <sup>(1)</sup> (DDRE)	Read:	0	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$000F	Port Option Control Register (POC)	Read:	0	0	LDD	0	0	PCP	PBP	PAP
		Write:								
		Reset:	0	0	1	0	0	0	0	0

Note 1. Only available on 52-pin QFP.

 = Unimplemented

**Figure 12-1. I/O Port Register Summary (Continued)**

## 12.3 Port A

Port A is an 8-bit general-purpose bidirectional I/O port with software configurable pullups.

### 12.3.1 Port A Data Register

The port A data register contains a data latch for each of the eight port A pins.

Address: \$0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Write:								
Reset:	Unaffected by Reset							

**Figure 12-2. Port A Data Register (PTA)**

#### PTA[7:0] — Port A Data Bits

These read/write bits are software programmable. Data direction of each port A pin is under the control of the corresponding bit in data direction register A. Reset has no effect on port A data.

The port A pullup enable bit, PAP, in the port option control register (POC) enables pullups on port A pins if the respective pin is configured as an input. (See [12.8 Port Options](#).)



### 12.3.2 Data Direction Register A

Data direction register A determines whether each port A pin is an input or an output. Writing a logic 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a logic 0 disables the output buffer.

Address: \$0004

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 12-3. Data Direction Register A (DDRA)**

#### DDRA[7:0] — Data Direction Register A Bits

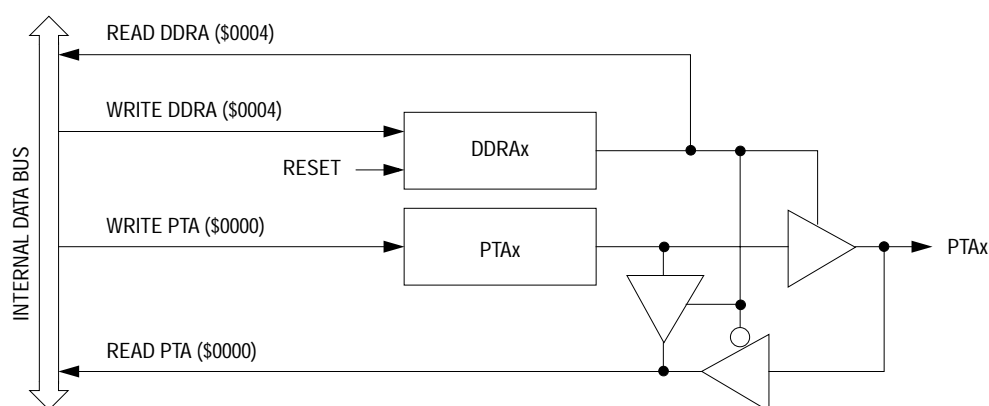
These read/write bits control port A data direction. Reset clears DDRA[7:0], configuring all port A pins as inputs.

1 = Corresponding port A pin configured as output

0 = Corresponding port A pin configured as input

**NOTE:** Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.

Figure 12-4 shows the port A I/O logic.



**Figure 12-4. Port A I/O Circuit**

When bit DDRAx is a logic 1, reading address \$0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address \$0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 12-1](#) summarizes the operation of the port A pins.

**Table 12-1. Port A Pin Functions**

DDRA Bit	PTA Bit	I/O Pin Mode	Accesses to DDRA	Accesses to PTA	
			Read/Write	Read	Write
0	X <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRA[7:0]	Pin	PTA[7:0] <sup>(3)</sup>
1	X	Output	DDRA[7:0]	PTA[7:0]	PTA[7:0]

1. X = don't care

2. Hi-Z = high impedance

3. Writing affects data register, but does not affect input.

## 12.4 Port B

Port B is an 8-bit, general-purpose, bidirectional I/O port with software configurable pullups.

### 12.4.1 Port B Data Register

The port B data register contains a data latch for each of the eight port B pins.

Address: \$0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Write:								
Reset:	Unaffected by Reset							

**Figure 12-5. Port B Data Register (PTB)**

#### PTB[7:0] — Port B Data Bits

These read/write bits are software-programmable. Data direction of each port B pin is under the control of the corresponding bit in data direction register B. Reset has no effect on port B data.

The port B pullup enable bit, PBP, in the port option control register (POC) enables pullups on port B pins if the respective pin is configured as an input. (See [12.8 Port Options](#).)

## 12.4.2 Data Direction Register B

Data direction register B determines whether each port B pin is an input or an output. Writing a logic 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a logic 0 disables the output buffer.

Address: \$0005

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 12-6. Data Direction Register B (DDRB)**

### DDRB[7:0] — Data Direction Register B Bits

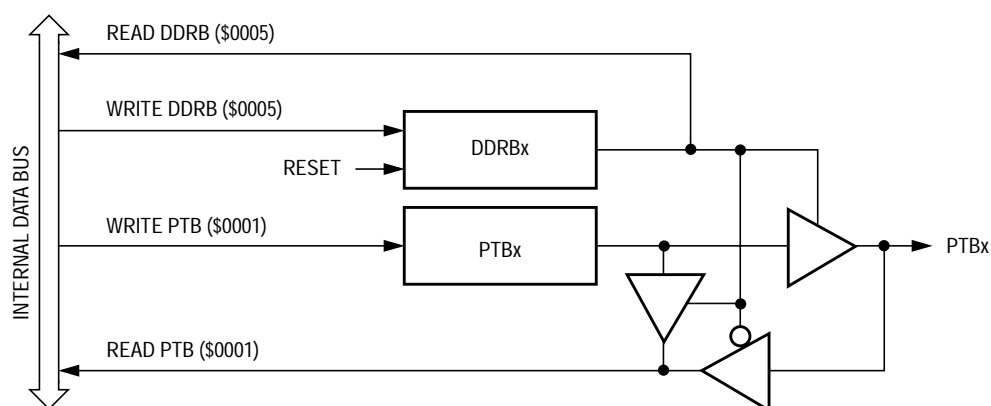
These read/write bits control port B data direction. Reset clears DDRB[7:0], configuring all port B pins as inputs.

1 = Corresponding port B pin configured as output

0 = Corresponding port B pin configured as input

**NOTE:** Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.

**Figure 12-7** shows the port B I/O logic.



**Figure 12-7. Port B I/O Circuit**

When bit DDRBx is a logic 1, reading address \$0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address \$0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 12-2](#) summarizes the operation of the port B pins.

**Table 12-2. Port B Pin Functions**

DDRB Bit	PTB Bit	I/O Pin Mode	Accesses to DDRB	Accesses to PTB	
			Read/Write	Read	Write
0	X <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRB[7:0]	Pin	PTB[7:0] <sup>(3)</sup>
1	X	Output	DDRB[7:0]	PTB[7:0]	PTB[7:0]

1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

## 12.5 Port C

Port C is an 8-bit, general-purpose, bidirectional I/O port with software configurable pullups and current drive options.

**NOTE:** *On the 42-lead SDIP package the PTC7–PTC5 pads are not bonded out. Set these ports to output to avoid floating inputs.*

### 12.5.1 Port C Data Register

The port C data register contains a data latch for each of the eight port C pins.

Address: \$0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
Write:								
Reset:	Unaffected by Reset							

**Figure 12-8. Port C Data Register (PTC)**

#### PTC[7:0] — Port C Data Bits

These read/write bits are software-programmable. Data direction of each port C pin is under the control of the corresponding bit in data direction register C. Reset has no effect on port C data.

The LED direct drive bit, LDD, in the port option control register (POC) controls the drive options for port C.

The port C pullup enable bit, PCP, in the port option control register (POC) enables pullups on PTC[7:0] if the respective pin is configured as an input. (See [12.8 Port Options](#).)

## 12.5.2 Data Direction Register C

Data direction register C determines whether each port C pin is an input or an output. Writing a logic 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a logic 0 disables the output buffer.

Address: \$0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 12-9. Data Direction Register C (DDRC)**

### DDRC[7:0] — Data Direction Register C Bits

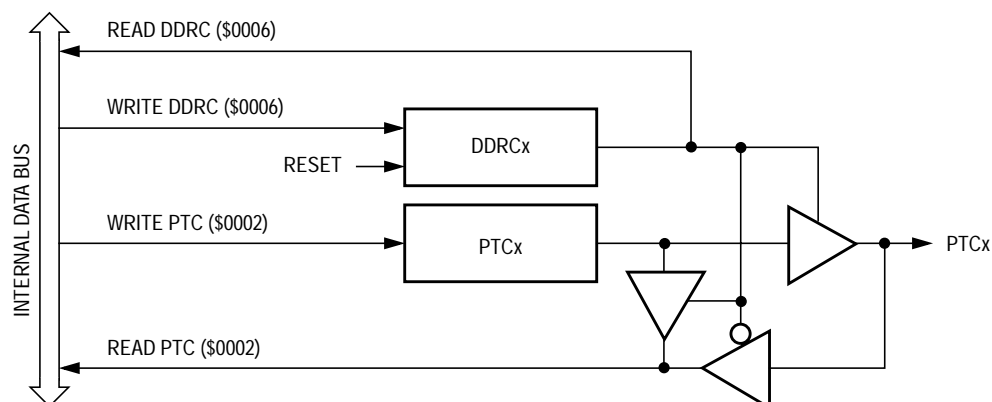
These read/write bits control port C data direction. Reset clears DDRC[7:0], configuring all port C pins as inputs.

1 = Corresponding port C pin configured as output

0 = Corresponding port C pin configured as input

**NOTE:** Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.

Figure 12-10 shows the port C I/O logic.



**Figure 12-10. Port C I/O Circuit**

When bit DDRCx is a logic 1, reading address \$0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address \$0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 12-3](#) summarizes the operation of the port C pins.

**Table 12-3. Port C Pin Functions**

DDRC Bit	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	X <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRC[7:0]	Pin	PTC[7:0] <sup>(3)</sup>
1	X	Output	DDRC[7:0]	PTC[7:0]	PTC[7:0]

1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

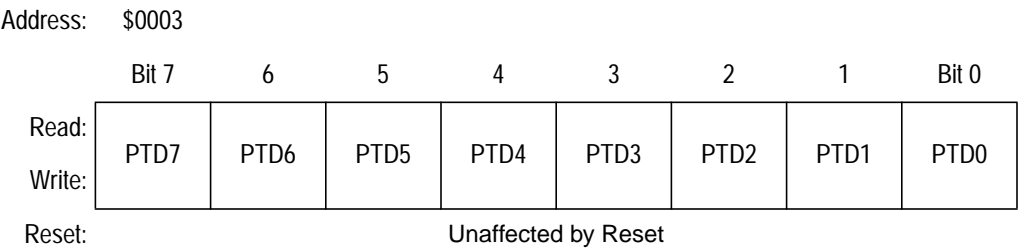


## 12.6 Port D

Port D is an 8-bit, general-purpose, bidirectional I/O port that shares its pins with the keyboard interrupt module (KBI).

### 12.6.1 Port D Data Register

The port D data register contains a data latch for each of the eight port D pins.



**Figure 12-11. Port D Data Register (PTD)**

#### PTD[7:0] — Port D Data Bits

These read/write bits are software programmable. Data direction of each port D pin is under control of the corresponding bit in data direction register D. Reset has no effect on port D data.

The keyboard interrupt enable bits, KBIE7–KBIE0, in the keyboard interrupt control register (KBICR) enable the port D pins as external interrupt pins. (See [Section 15. Keyboard Interrupt Module \(KBI\)](#).)

## 12.6.2 Data Direction Register D

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

Address: \$0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 12-12. Data Direction Register D (DDRD)**

### DDRD[7:0] — Data Direction Register D Bits

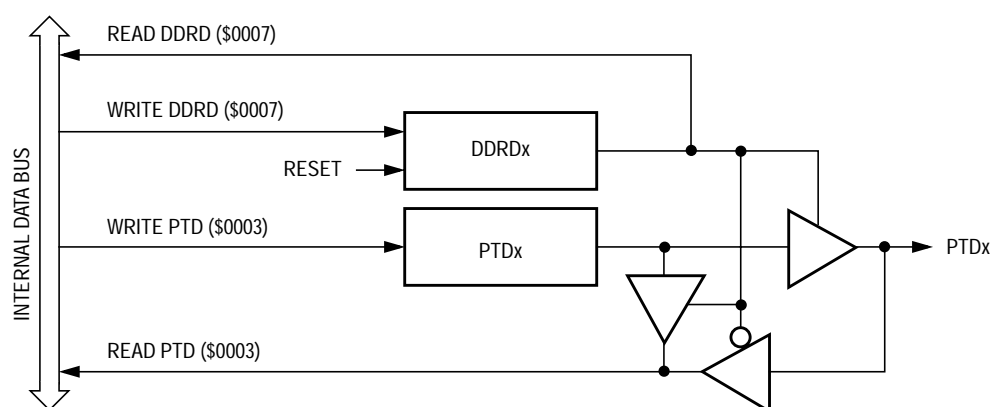
These read/write bits control port D data direction. Reset clears DDRD[7:0], configuring all port D pins as inputs.

1 = Corresponding port D pin configured as output

0 = Corresponding port D pin configured as input

**NOTE:** Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.

**Figure 12-13** shows the port D I/O circuit logic.



**Figure 12-13. Port D I/O Circuit**

When bit DDRDx is a logic 1, reading address \$0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address \$0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 12-4](#) summarizes the operation of the port D pins.

**Table 12-4. Port D Pin Functions**

DDRD Bit	PTD Bit	I/O Pin Mode	Accesses to DDRD	Accesses to PTD	
			Read/Write	Read	Write
0	X <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRD[7:0]	Pin	PTD[7:0] <sup>(3)</sup>
1	X	Output	DDRD[7:0]	PTD[7:0]	PTD[7:0]

1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

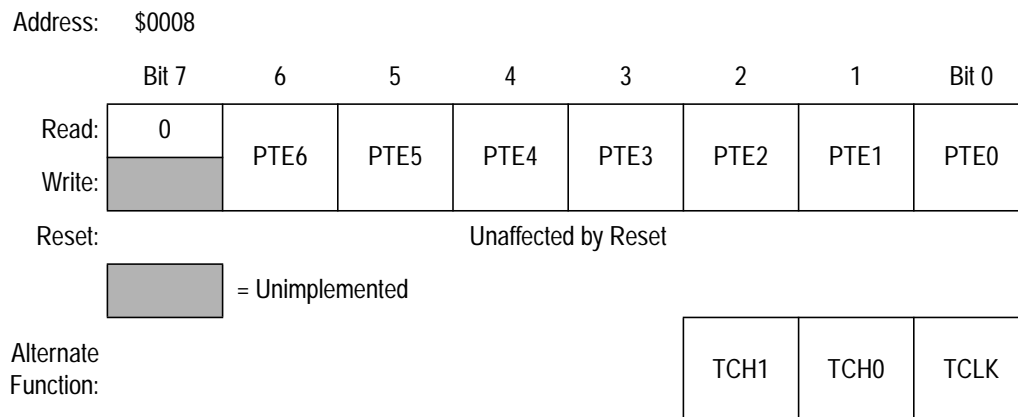
## 12.7 Port E

Port E is a 7-bit special function port that shares three of its pins with the timer interface module (TIM).

**NOTE:** *On the 42-lead SDIP package the port E pads are not bonded out. Set these ports to output to avoid floating inputs.*

### 12.7.1 Port E Data Register

The port E data register contains a data latch for each of the seven port E pins.



**Figure 12-14. Port E Data Register (PTE)**

#### PTE[6:0] — Port E Data Bits

PTE[6:0] are read/write, software-programmable bits. Data direction of each port E pin is under the control of the corresponding bit in data direction register E.

#### TCH1–TCH0 — Timer Channel I/O Bits

The PE2/TCH1–PE1/TCH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB and ELSxA, determine whether the PE2/TCH1–PE1/TCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See [Section 11. Timer Interface Module \(TIM\)](#).)

**NOTE:** *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins.*

#### TCLK — Timer Clock Input

The PE0/TCLK pin is the external clock input for the TIM. The prescaler select bits, PS2–PS0, selects PE0/TCLK as the TIM clock input. When not selected as the TIM clock, PE0/TCLK is available for general-purpose I/O. (See [Section 11. Timer Interface Module \(TIM\)](#).)

## 12.7.2 Data Direction Register E

Data direction register E determines whether each port E pin is an input or an output. Writing a logic 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a logic 0 disables the output buffer.

Address: \$000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 12-15. Data Direction Register E (DDRE)**

### DDRE[6:0] — Data Direction Register E Bits

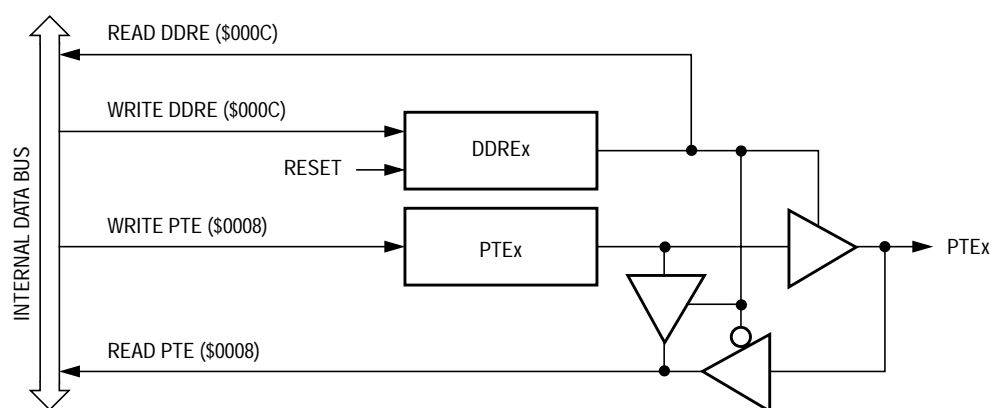
These read/write bits control port E data direction. Reset clears DDRE[6:0], configuring all port E pins as inputs.

1 = Corresponding port E pin configured as output

0 = Corresponding port E pin configured as input

**NOTE:** Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1.

Figure 12-16 shows the port E I/O circuit logic.



**Figure 12-16. Port E I/O Circuit**

When bit DDREx is a logic 1, reading address \$0008 reads the PTE<sub>x</sub> data latch. When bit DDREx is a logic 0, reading address \$0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 12-5** summarizes the operation of the port E pins.

**Table 12-5. Port E Pin Functions**

DDRE Bit	PTE Bit	I/O Pin Mode	Accesses to DDRE	Accesses to PTE	
			Read/Write	Read	Write
0	X <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRE[6:0]	Pin	PTE[6:0] <sup>(3)</sup>
1	X	Output	DDRE[6:0]	PTE[6:0]	PTE[6:0]

1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

## 12.8 Port Options


All pins of port A, port B, and port C have programmable pullup resistors. Port C also has LED drive capability.

### 12.8.1 Port Option Control Register

The pullup option for each port is controlled by one bit in the port option control register. One bit controls the LED drive configuration on port C.

Address: \$000F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	LDD	0	0	PCP	PBP	PAP
Write:								
Reset:	0	0	1	0	0	0	0	0

 = Unimplemented

**Figure 12-17. Port Option Control Register (POC)**

#### LDD — LED Direct Drive Control

This read/write bit controls the output current capability of port C. When set, the port C pins have current limiting ability so that an LED can be connected directly between the port pin and  $V_{DD}$  or  $V_{SS}$  without the need of a series resistor.

- 1 = When respective port is configured as an output, make port C become current, limiting 3 mA source/10 mA sink port pins.
- 0 = Configure port C to become standard I/O port pins.

#### PCP — Port C Pullup Enable

This read/write bit controls the pullup option for port C[7:0] if its respective port pin is configured as an input.

- 1 = Configure port C to have internal pullups.
- 0 = Disconnect port C internal pullups.



**PBP — Port B Pullup Enable**

This read/write bit controls the pullup option for the eight bits of port B if its respective port pin is configured as an input.

1 = Configure port B to have internal pullups.

0 = Disconnect port B internal pullups.

**PAP — Port A Pullup Enable**

This read/write bit controls the pullup option for the eight bits of port A if its respective port pin is configured as an input.

1 = Configure port A to have internal pullups.

0 = Disconnect port A internal pullups.



## Section 13. Computer Operating Properly (COP)

### 13.1 Contents

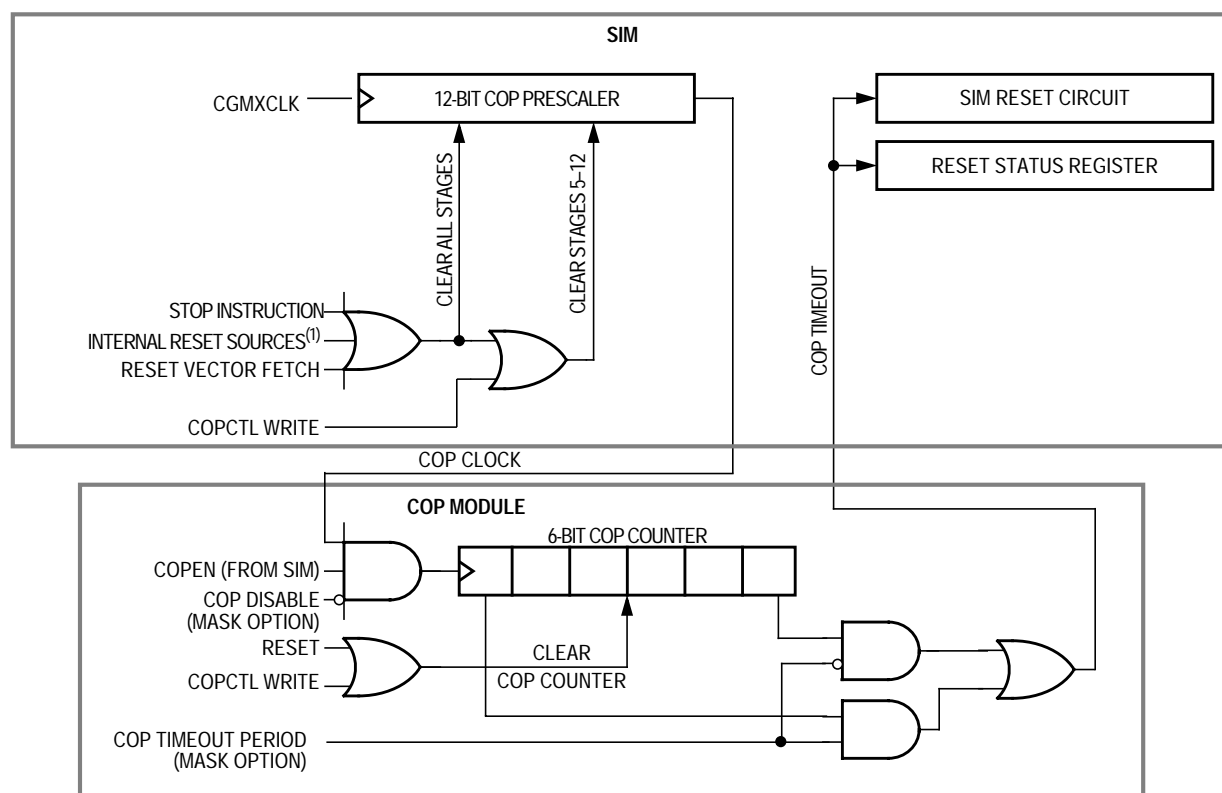
13.2	Introduction .....	211
13.3	Functional Description .....	212
13.4	I/O Signals .....	213
13.4.1	CGMXCLK .....	213
13.4.2	STOP Instruction .....	213
13.4.3	COPCTL Write .....	214
13.4.4	Power-On Reset .....	214
13.4.5	Internal Reset .....	214
13.4.6	Reset Vector Fetch .....	214
13.4.7	COPD (COP Disable) .....	214
13.4.8	COPRS (COP Rate Select) .....	214
13.5	COP Control Register .....	215
13.6	Interrupts .....	215
13.7	Monitor Mode .....	215
13.8	Low-Power Modes .....	215
13.8.1	Wait Mode .....	215
13.8.2	Stop Mode .....	216
13.9	COP Module During Break Mode .....	216

### 13.2 Introduction

This COP module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. A mask option is available to disable the COP module.

## 13.3 Functional Description

**Figure 13-1** shows the structure of the COP module.



NOTE:

1. See [Section 8. System Integration Module \(SIM\)](#) for more details.

**Figure 13-1. COP Block Diagram**

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$FFFF	COP Control Register (COPCTL)	Read: Low Byte of Reset Vector							
		Write: Writing Clears COP Counter (Any Value)							
		Reset: Unaffected by Reset							

**Figure 13-2. COP I/O Register Summary**

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after  $2^{18} - 2^4$  or  $2^{13} - 2^4$  CGMXCLK cycles, depending on the mask option selected for COP timeout period. With a  $2^{18} - 2^4$  CGMXCLK cycle overflow option, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location \$FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12 through 5 of the SIM counter.

**NOTE:** *Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.*

A COP reset pulls the  $\overline{\text{RST}}$  pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the  $\overline{\text{RST}}$  pin or the  $\overline{\text{IRQ1}}$  is held at  $V_{\text{TST}}$ . During the break state,  $V_{\text{TST}}$  on the  $\overline{\text{RST}}$  pin disables the COP.

**NOTE:** *Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.*

## 13.4 I/O Signals

The following paragraphs describe the signals shown in [Figure 13-1](#).

### 13.4.1 CGMXCLK

CGMXCLK is the crystal oscillator output signal. CGMXCLK frequency is equal to the crystal frequency.

### 13.4.2 STOP Instruction

The STOP instruction clears the COP prescaler.

### 13.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see [13.5 COP Control Register](#)) clears the COP counter and clears bits 12 through 4 of the SIM counter. Reading the COP control register returns the low byte of the reset vector.

### 13.4.4 Power-On Reset

The power-on reset (POR) circuit in the SIM clears the COP prescaler 4096 CGMXCLK cycles after power-up.

### 13.4.5 Internal Reset

An internal reset clears the SIM counter and the COP counter.

### 13.4.6 Reset Vector Fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the COP prescaler.

### 13.4.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit mask option.

### 13.4.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select mask option.

### 13.5 COP Control Register

The COP control register is located at address \$FFFF and overlaps the reset vector. Writing any value to \$FFFF clears the COP counter and starts a new timeout period. Reading location \$FFFF returns the low byte of the reset vector.

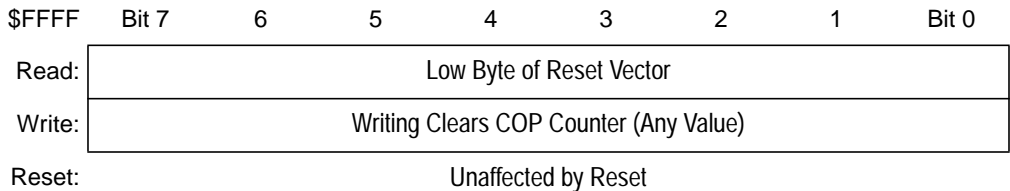


Figure 13-3. COP Control Register (COPCTL)

### 13.6 Interrupts

The COP does not generate CPU interrupt requests.

### 13.7 Monitor Mode

The COP is disabled in monitor mode when  $V_{TST}$  is present on the  $\overline{IRQ1}$  pin or on the  $\overline{RST}$  pin.

### 13.8 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power standby modes.

#### 13.8.1 Wait Mode

The COP remains active during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine.

### 13.8.2 Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the COP prescaler. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

To prevent inadvertently turning off the COP with a STOP instruction, a mask option is available that disables the STOP instruction. When the STOP instruction is disabled, execution of a STOP instruction results in an illegal opcode reset.

### 13.9 COP Module During Break Mode

The COP is disabled during a break interrupt when  $V_{TST}$  is present on the  $\overline{RST}$  pin.



## Section 14. External Interrupt (IRQ)

### 14.1 Contents

14.2	Introduction .....	217
14.3	Features .....	218
14.4	Functional Description .....	218
14.5	$\overline{\text{IRQ1}}$ Pin .....	221
14.6	IRQ Module During Break Interrupts .....	222
14.7	IRQ Status and Control Register .....	222

### 14.2 Introduction

The IRQ module provides an external interrupt input.

### 14.3 Features

Features of the IRQ module include:

- A Dedicated External Interrupt Pin,  $\overline{\text{IRQ1}}$
- IRQ1 Interrupt Control Bits
- Hysteresis Buffer
- Programmable Edge-Only or Edge and Level Interrupt Sensitivity
- Automatic Interrupt Acknowledge
- $\overline{\text{IRQ1}}$  Pin Includes Internal Pullup Resistor

### 14.4 Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. [Figure 14-1](#) shows the structure of the IRQ module.

Interrupt signals on the  $\overline{\text{IRQ1}}$  pin are latched into the IRQ1 latch. An interrupt latch remains set until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the IRQ latch.
- Software clear — Software can clear the interrupt latch by writing to the acknowledge bit in the interrupt status and control register (ISCR). Writing a logic 1 to the ACK1 bit clears the IRQ1 latch.
- Reset — A reset automatically clears the interrupt latch.

The external interrupt pin is falling-edge-triggered and is software-configurable to be either falling-edge or low-level-triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the  $\overline{\text{IRQ1}}$  pin.

When the interrupt pin is edge-triggered only, the CPU interrupt request remains set until a vector fetch, software clear, or reset occurs.

When the interrupt pin is both falling-edge and low-level-triggered, the CPU interrupt request remains set until both of the following occur:

- Vector fetch or software clear
- Return of the interrupt pin to logic 1

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK1 bit in the ISCR mask all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the IMASK1 bit is clear.

**NOTE:** *The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See [8.6 Exception Control](#).)*

# External Interrupt (IRQ)

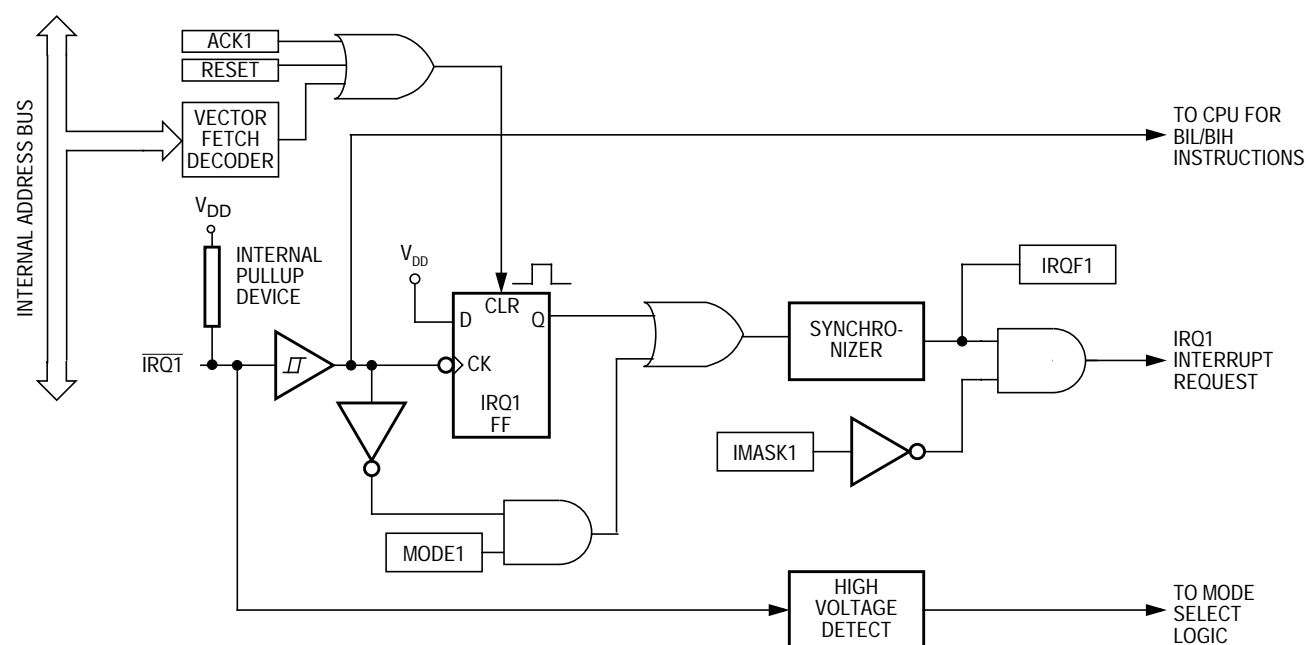


Figure 14-1. IRQ Module Block Diagram

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$001E	IRQ Status/Control Register (ISCR)	Read: 0	0	0	0	IRQF1	0	IMASK1	MODE1
		Write:					ACK1		
		Reset:	0	0	0	0	0	0	0


 = Unimplemented

Figure 14-2. IRQ I/O Register Summary

## 14.5 $\overline{\text{IRQ1}}$ Pin

A logic 0 on the  $\overline{\text{IRQ1}}$  pin can latch an interrupt request into the  $\overline{\text{IRQ1}}$  latch. A vector fetch, software clear, or reset clears the  $\overline{\text{IRQ1}}$  latch.

If the MODE1 bit is set, the  $\overline{\text{IRQ1}}$  pin is both falling-edge-sensitive and low-level-sensitive. With MODE1 set, both of the following actions must occur to clear IRQ1:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK1 bit in the interrupt status and control register (ISCR). The ACK1 bit is useful in applications that poll the  $\overline{\text{IRQ1}}$  pin and require software to clear the IRQ1 latch. Writing to the ACK1 bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK1 does not affect subsequent transitions on the  $\overline{\text{IRQ1}}$  pin. A falling edge that occurs after writing to the ACK1 bit latches another interrupt request. If the IRQ1 mask bit, IMASK1, is clear, the CPU loads the program counter with the vector address at locations \$FFFA and \$FFFB.
- Return of the  $\overline{\text{IRQ1}}$  pin to logic 1 — As long as the  $\overline{\text{IRQ1}}$  pin is at logic 0, IRQ1 remains active.

The vector fetch or software clear and the return of the  $\overline{\text{IRQ1}}$  pin to logic 1 may occur in any order. The interrupt request remains pending as long as the  $\overline{\text{IRQ1}}$  pin is at logic 0. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE1 bit is clear, the  $\overline{\text{IRQ1}}$  pin is falling-edge-sensitive only. With MODE1 clear, a vector fetch or software clear immediately clears the IRQ1 latch.

The IRQF1 bit in the ISCR register can be used to check for pending interrupts. The IRQF1 bit is not affected by the IMASK1 bit, which makes it useful in applications where polling is preferred.

Use the BIH or BIL instruction to read the logic level on the  $\overline{\text{IRQ1}}$  pin.

**NOTE:** *When using the level-sensitive interrupt trigger, avoid false interrupts by masking interrupt requests in the interrupt routine.*

### 14.6 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the  $\overline{\text{IRQ1}}$  latch can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear the latches during the break state. (See [Section 8. System Integration Module \(SIM\)](#).)

To allow software to clear the  $\overline{\text{IRQ1}}$  latch during a break interrupt, write a logic 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latches during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the ACK1 bit in the IRQ status and control register during the break state has no effect on the IRQ latch.


### 14.7 IRQ Status and Control Register

The IRQ status and control register (ISCR) controls and monitors operation of the IRQ module. The ISCR has the following functions:

- Shows the state of the  $\overline{\text{IRQ1}}$  flag
- Clears the  $\overline{\text{IRQ1}}$  latch
- Masks  $\overline{\text{IRQ1}}$  and interrupt request
- Controls triggering sensitivity of the  $\overline{\text{IRQ1}}$  interrupt pin

Address: \$001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	IRQF1	0	IMASK1	MODE1
Write:						ACK1		
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 14-3. IRQ Status and Control Register (ISCR)**

#### IRQF1 — $\overline{\text{IRQ1}}$ Flag

This read-only status bit is high when the  $\overline{\text{IRQ1}}$  interrupt is pending.

1 =  $\overline{\text{IRQ1}}$  interrupt pending

0 =  $\overline{\text{IRQ1}}$  interrupt not pending

#### ACK1 — $\overline{\text{IRQ1}}$ Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the  $\overline{\text{IRQ1}}$  latch. ACK1 always reads as logic 0. Reset clears ACK1.

#### IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables  $\overline{\text{IRQ1}}$  interrupt requests. Reset clears IMASK1.

1 =  $\overline{\text{IRQ1}}$  interrupt requests disabled

0 =  $\overline{\text{IRQ1}}$  interrupt requests enabled

#### MODE1 — $\overline{\text{IRQ1}}$ Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the  $\overline{\text{IRQ1}}$  pin. Reset clears MODE1.

1 =  $\overline{\text{IRQ1}}$  interrupt requests on falling edges and low levels

0 =  $\overline{\text{IRQ1}}$  interrupt requests on falling edges only





## Section 15. Keyboard Interrupt Module (KBI)

### 15.1 Contents

- 15.2 Introduction.....225
- 15.3 Features .....226
- 15.4 Functional Description .....227
- 15.5 Keyboard Initialization.....228
- 15.6 Low-Power Modes .....229
  - 15.6.1 Wait Mode .....229
  - 15.6.2 Stop Mode.....229
- 15.7 Keyboard Module During Break Interrupts .....230
- 15.8 I/O Registers.....230
  - 15.8.1 Keyboard Status and Control Register .....230
  - 15.8.2 Keyboard Interrupt Enable Register .....232

### 15.2 Introduction

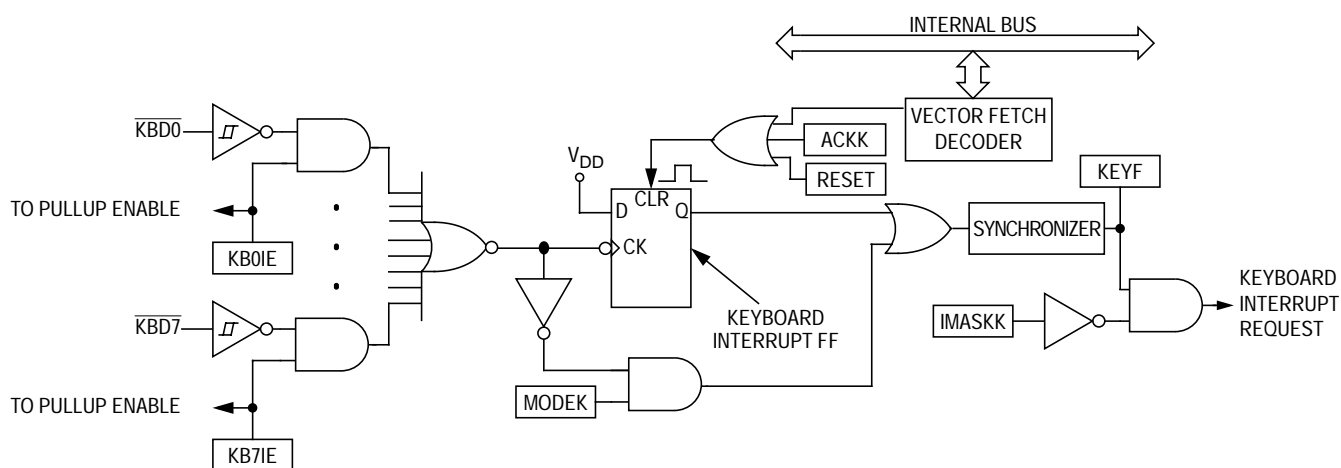
The keyboard module provides eight independently maskable external interrupts.

# Keyboard Interrupt Module (KBI)

## 15.3 Features

Features of the keyboard interrupt module (KBI) include:

- Eight Keyboard Interrupt Pins with Separate Keyboard Interrupt Enable Bits and One Keyboard Interrupt Mask
- Hysteresis Buffers
- Programmable Edge-Only or Edge- and Level-Interrupt Sensitivity
- Exit from Low-Power Modes



**Figure 15-1. Keyboard Module Block Diagram**

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$000D	Keyboard Status and Control Register (KBSCR)	Read: 0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:					ACKK		
		Reset: 0	0	0	0	0	0	0	0
\$000E	Keyboard Interrupt Enable Register (KBIER)	Read: KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:							
		Reset: 0	0	0	0	0	0	0	0

= Unimplemented

**Figure 15-2. I/O Register Summary**

## 15.4 Functional Description

Writing to the KBIE7–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port D pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register (KBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine also can prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations \$FFE0 and \$FFE1.

- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

If the MODEK bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, use the data direction register to configure the pin as an input and read the data register.

**NOTE:** *Setting a keyboard interrupt enable bit (KBIE<sub>x</sub>) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a logic 0 for software to read the pin.*

### 15.5 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a logic 1. Therefore, a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIE<sub>x</sub> bits in the keyboard interrupt enable register.

3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRD bits in data direction register D.
2. Write logic 1s to the appropriate port D data register bits.
3. Enable the KBI pins by setting the appropriate KBIE<sub>x</sub> bits in the keyboard interrupt enable register.

## 15.6 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power standby modes.

### 15.6.1 Wait Mode

The keyboard module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

### 15.6.2 Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

### 15.7 Keyboard Module During Break Interrupts

The system integration module (SIM) controls whether the keyboard interrupt latch can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state.

To allow software to clear the keyboard interrupt latch during a break interrupt, write a logic 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latch during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. (See [15.8.1 Keyboard Status and Control Register](#).)

### 15.8 I/O Registers

These registers control and monitor operation of the keyboard module:

- Keyboard status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)


#### 15.8.1 Keyboard Status and Control Register

The keyboard status and control register:

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard interrupt triggering sensitivity

Address: \$000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
Write:						ACKK		
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 15-3. Keyboard Status and Control Register (KBSCR)**

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

**KEYF — Keyboard Flag Bit**

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending

0 = No keyboard interrupt pending

**ACKK — Keyboard Acknowledge Bit**

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

**IMASKK — Keyboard Interrupt Mask Bit**

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked

0 = Keyboard interrupt requests not masked

**MODEK — Keyboard Triggering Sensitivity Bit**

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODEK.

1 = Keyboard interrupt requests on falling edges and low levels

0 = Keyboard interrupt requests on falling edges only

## 15.8.2 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port D pin to operate as a keyboard interrupt pin.

Address: \$000E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 15-4. Keyboard Interrupt Enable Register (KBIER)**

### KBIE7–KBIE0 — Keyboard Interrupt Enable Bits

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

1 = PDx pin enabled as keyboard interrupt pin

0 = PDx pin not enabled as keyboard interrupt pin



## Section 16. Break Module (BREAK)

### 16.1 Contents

16.2	Introduction .....	233
16.3	Features .....	234
16.4	Functional Description .....	234
16.4.1	Flag Protection During Break Interrupts .....	236
16.4.2	CPU During Break Interrupts .....	236
16.4.3	TIM During Break Interrupts .....	236
16.4.4	COP During Break Interrupts .....	236
16.5	Break Module Registers .....	237
16.5.1	Break Status and Control Register .....	237
16.5.2	Break Address Registers .....	238
16.6	Low-Power Modes .....	238
16.6.1	Wait Mode .....	238
16.6.2	Stop Mode .....	238

### 16.2 Introduction

This section describes the break module. To enter a background program, the break module can generate a break interrupt that stops normal program flow at a defined address.

### 16.3 Features

Features of the break module include:

- Accessible I/O Registers during the Break Interrupt
- CPU-Generated Break Interrupts
- Software-Generated Break Interrupts
- COP Disabling during Break Interrupts

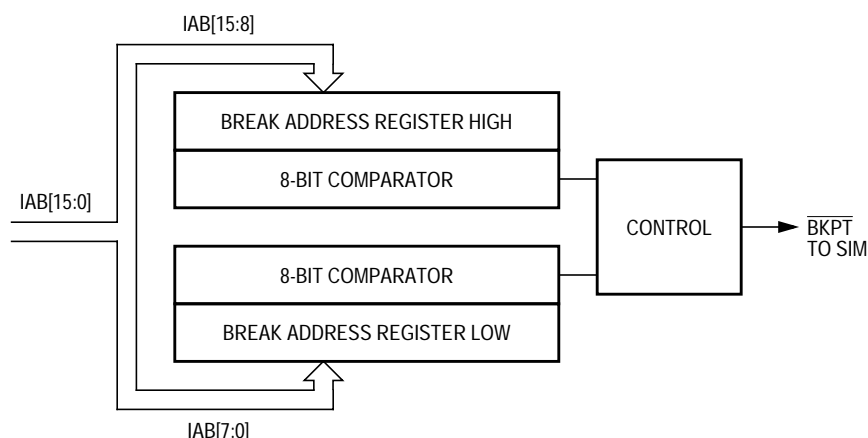
### 16.4 Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal ( $\overline{\text{BKPT}}$ ) to the SIM. The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter vectors to \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode).

These events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 1 to the BRKA bit in the break status and control register.

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. [Figure 16-1](#) shows the structure of the break module.



**Figure 16-1. Break Module Block Diagram**

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$FE0C	Break Address Register High (BRKH)	Read: Bit 15	14	13	12	11	10	9	Bit 8
		Write:							
		Reset:	0	0	0	0	0	0	0
\$FE0D	Break Address Register Low (BRKL)	Read: Bit 7	6	5	4	3	2	1	Bit 0
		Write:							
		Reset:	0	0	0	0	0	0	0
\$FE0E	Break Status and Control Register (BRKSCR)	Read: BRKE	BRKA	0	0	0	0	0	0
		Write:							
		Reset:	0	0	0	0	0	0	0

= Unimplemented

**Figure 16-2. Break I/O Register Summary**

### 16.4.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See [8.8.3 Break Flag Control Register](#) and see the break interrupts subsection for each module.)

### 16.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC:\$FFFD (\$FEFC:\$FEFD in monitor mode)

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

### 16.4.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

### 16.4.4 COP During Break Interrupts

The COP is disabled during a break interrupt when  $V_{TST}$  is present on the  $\overline{RST}$  pin.

## 16.5 Break Module Registers

Three registers control and monitor operation of the break module:


- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)

### 16.5.1 Break Status and Control Register

The break status and control register contains break module enable and status bits.

Address: \$FE0E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	BRKE	BRKA	0	0	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 16-3. Break Status and Control Register (BRKSCR)**

#### BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a logic 0 to bit 7. Reset clears the BRKE bit.

- 1 = Breaks enabled on 16-bit address match
- 0 = Breaks disabled

#### BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a logic 1 to BRKA generates a break interrupt. Clear BRKA by writing a logic 0 to it before exiting the break routine. Reset clears the BRKA bit.

- 1 = Break address match
- 0 = No break address match

## 16.5.2 Break Address Registers

The break address registers contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

BRKH Address: \$FE0C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

BRKL Address: \$FE0D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 16-4. Break Address Registers (BRKH and BRKL)**

## 16.6 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power standby modes.

### 16.6.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract 1 from the return address on the stack if SBSW is set (see [8.7 Low-Power Modes](#)). Clear the SBSW bit by writing logic 0 to it.

### 16.6.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the break status register. See [8.8 SIM Registers](#).

## Section 17. Electrical Specifications

### 17.1 Contents

17.2	Introduction . . . . .	239
17.3	Absolute Maximum Ratings . . . . .	240
17.4	Functional Operating Range . . . . .	241
17.5	Thermal Characteristics . . . . .	241
17.6	DC Electrical Characteristics . . . . .	242
17.7	Control Timing . . . . .	243
17.8	Oscillator Characteristics . . . . .	243
17.9	USB DC Electrical Characteristics . . . . .	244
17.10	USB Low-Speed Source Electrical Characteristics . . . . .	245
17.11	USB Signaling Levels . . . . .	246
17.12	Timer Interface Module Characteristics . . . . .	247
17.13	Memory Characteristics . . . . .	247

### 17.2 Introduction

This section contains electrical and timing specifications. These values are design targets and have not yet been fully tested.

## 17.3 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

The MCU contains circuitry to protect the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in the table. Keep  $V_{In}$  and  $V_{Out}$  within the range  $V_{SS} \leq (V_{In} \text{ or } V_{Out}) \leq V_{DD}$ . Connect unused inputs to the appropriate voltage level, either  $V_{SS}$  or  $V_{DD}$ .

Characteristic <sup>(1)</sup>	Symbol	Value	Unit
Supply Voltage	$V_{DD}, V_{DDREG}$	−0.3 to +6.0	V
Regulator Supply Voltage	$V_{DDREG}$	−0.3 to +6.0	V
Input Voltage	$V_{IN}$	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	V
Programming Voltage	$V_{PP}$	$V_{SS} - 0.3$ to 14.0	V
Maximum Current Per Pin Excluding $V_{DD}$ and $V_{SS}$	I	± 25	mA
Storage Temperature	$T_{STG}$	−55 to +150	°C
Maximum Current Out of $V_{SS}$	$I_{MVSS}$	100	mA
Maximum Current Into $V_{DD}$	$I_{MVDD}$	100	mA

1. Voltages referenced to  $V_{SS}$

**NOTE:** This device is not guaranteed to operate properly at the maximum ratings. Refer to [17.6 DC Electrical Characteristics](#) for guaranteed operating conditions.



## 17.4 Functional Operating Range

Characteristic	Symbol	Value	Unit
Operating Temperature Range	$T_A$	0 to 85	°C
Operating Voltage Range	$V_{DD}$	4.4 to 5.5	V
Operating Regulator Voltage Range Bypass Not Enabled Bypass Enabled	$V_{DDREG}$	4.4 to 5.5 3.0 to 3.6	V

## 17.5 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal Resistance Quad Flat Pack, 52 Pins	$\theta_{JA}$	70	°C/W
I/O Pin Power Dissipation	$P_{I/O}$	User Determined	W
Power Dissipation <sup>(1)</sup>	$P_D$	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} =$ $K/(T_J + 273\text{ °C})$	W
Constant <sup>(2)</sup>	K	$P_D \times (T_A + 273\text{ °C})$ $+ P_D^2 \times \theta_{JA}$	W/°C
Average Junction Temperature	$T_J$	$T_A + (P_D \times \theta_{JA})$	°C
Maximum Junction Temperature	$T_{JM}$	100	°C

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for a known  $T_A$  and measured  $P_D$ .  
With this value of K,  $P_D$  and  $T_J$  can be determined for any value of  $T_A$ .

## 17.6 DC Electrical Characteristics

Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	Max	Unit
Output High Voltage ( $I_{Load} = -2.0$ mA) All I/O Pins	$V_{OH}$	$V_{DD} - 0.8$	—	—	V
Output Low Voltage ( $I_{Load} = 1.6$ mA) All I/O Pins	$V_{OL}$	—	—	0.4	V
Input High Voltage All Ports, $\overline{IRQ1}$ , $\overline{RST}$ , OSC1	$V_{IH}$	$0.7 \times V_{DD}$	—	$V_{DD}$	V
Input Low Voltage All Ports, $\overline{IRQ1}$ , $\overline{RST}$ , OSC1	$V_{IL}$	$V_{SS}$	—	$0.3 \times V_{DD}$	V
Output High Current ( $V_{OH} = 2.1$ V) Port C in LDD Mode	$I_{OH}$	2.5	4.6	7	mA
Output Low Current ( $V_{OL} = 2.3$ V) Port C in LDD Mode	$I_{OL}$	7	13.2	20	mA
$V_{DD}$ Supply Current, $f_{op} = 1.5$ MHz Run, Low Speed USB <sup>(3)</sup> Run, USB Suspended <sup>(3)</sup> Wait, Low Speed USB <sup>(4)</sup> Wait, USB Suspended <sup>(4)</sup> Stop <sup>(5)</sup> 25 °C 0 °C to 85 °C	$I_{DD}$	— — — — — — —	4.25 3.75 2.25 1.75 180 190	6.0 5.0 3.5 3.0 225 250	mA mA mA mA μA μA
I/O Ports Hi-Z Leakage Current	$I_{IL}$	—	—	± 10	μA
Input Current	$I_{In}$	—	—	± 1	μA
Capacitance Ports (as Input or Output)	$C_{Out}$ $C_{In}$	— —	— —	12 8	pF
POR ReArm Voltage <sup>(6)</sup>	$V_{POR}$	0	—	100	mV
POR Rise Time Ramp Rate <sup>(7)</sup>	$R_{POR}$	0.035	—	—	V/ms
Monitor Mode Entry Voltage	$V_{TST}$	$V_{DD} + 2.5$	—	13.5	V
Pullup Resistor PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, $\overline{RST}$ , $\overline{IRQ1}$	$R_{PU}$	20	35	50	kΩ

- $V_{DD} = V_{DDREG} = 4.4$  to  $5.5$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ , unless otherwise noted.
- Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
- Run (operating)  $I_{DD}$  measured using external square wave clock source ( $f_{XCLK} = 6$  MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs.  $C_L = 20$  pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run  $I_{DD}$ . Measured with all modules enabled.
- Wait  $I_{DD}$  measured using external square wave clock source ( $f_{XCLK} = 6$  MHz); all inputs 0.2 V from rail; no dc loads; less than 100 pF on all outputs.  $C_L = 20$  pF on OSC2; 15 kΩ ± 5% termination resistors on D+ and D– pins; all ports configured as inputs; OSC2 capacitance linearly affects wait  $I_{DD}$ .
- STOP  $I_{DD}$  measured with USB in suspend mode; OSC1 grounded; REGOUT, D+, and D– not connected; no port pins sourcing current.
- Maximum is highest voltage that POR is guaranteed.
- If minimum  $V_{DD}$  is not reached before the internal POR reset is released,  $\overline{RST}$  must be driven low externally until minimum  $V_{DD}$  is reached.

## 17.7 Control Timing

Characteristic <sup>(1)</sup>	Symbol	Min	Max	Unit
Internal Operating Frequency <sup>(2)</sup>	$f_{OP}$	—	1.5	MHz
$\overline{RST}$ Input Pulse Width Low <sup>(3)</sup>	$t_{IRL}$	50	—	ns

- $V_{DD} = V_{DDREG} = 4.4$  to  $5.5$  Vdc;  $V_{SS} = 0$  Vdc; timing shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$ , unless otherwise noted.
- Some modules may require a minimum frequency greater than dc for proper operation; see appropriate table for this information.
- Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.

## 17.8 Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Crystal Frequency <sup>(1)</sup>	$f_{XCLK}$	1	—	8	MHz
External Clock Reference Frequency <sup>(1), (2)</sup>	$f_{XCLK}$	dc	—	32	MHz
Crystal Load Capacitance <sup>(3)</sup>	$C_L$	—	—	—	
Crystal Fixed Capacitance <sup>(3)</sup>	$C_1$	—	$2 \times C_L$	—	
Crystal Tuning Capacitance <sup>(3)</sup>	$C_2$	—	$2 \times C_L$	—	
Feedback Bias Resistor	$R_B$	—	10 M $\Omega$	—	
Series Resistor <sup>(3), (4)</sup>	$R_S$	—	—	—	

- The USB module is designed to function at  $f_{XCLK} = 6$  MHz. The values given here are oscillator specifications.
- No more than 10% duty cycle deviation from 50%
- Consult crystal vendor data sheet
- Not required for high-frequency crystals

## 17.9 USB DC Electrical Characteristics

Characteristic <sup>(1)</sup>	Symbol	Conditions	Min	Typ	Max	Unit
Hi-Z State Data Line Leakage	$I_{LO}$	$0\text{ V} < V_{In} < 3.3\text{ V}$	-10	—	+10	$\mu\text{A}$
Differential Input Sensitivity	$V_{DI}$	$ (D+) - (D-) $	0.2	—		V
Differential Common Mode Range	$V_{CM}$	Includes $V_{DI}$ Range	0.8	—	2.5	V
Single Ended Receiver Threshold	$V_{SE}$		0.8	—	2.0	V
Static Output Low	$V_{OL}$	$R_L$ of 1.5 k to 3.6 V	—	—	0.3	V
Static Output High	$V_{OH}$	$R_L$ of 15 k to GND	2.8	—	3.6	V
Regulator Supply Voltage <sup>(2), (3)</sup>	$V_{REGOUT}$	$I_L = 4\text{ mA}$	3.0	3.3	3.6	V
Regulator Bypass Capacitor	$C_{REGBYPASS}$		—	0.1	—	$\mu\text{F}$
Regulator Bulk Capacitor	$C_{REGBULK}$		1.0	—	—	$\mu\text{F}$

1.  $V_{DD} = 4.4 - 5.5\text{ V}$ ,  $V_{SS} = 0\text{ Vdc}$ ,  $T_A = 0\text{ }^\circ\text{C}$  to  $+85\text{ }^\circ\text{C}$ , unless otherwise noted

2. Transceiver pullup resistor of  $1.5\text{ k}\Omega \pm 5\%$  between REGOUT and D- and  $15\text{ k}\Omega \pm 5\%$  to ground termination resistors on D+ and D-.

3. No external current draw besides the USB-required external resistors should be connected to the REGOUT pin.

## 17.10 USB Low-Speed Source Electrical Characteristics

Characteristic <sup>(1)</sup>	Symbol	Conditions	Min	Typ	Max	Unit
Internal Operating Frequency	f <sub>OP</sub>	—	—	1.5	—	MHz
Transition Time <sup>(2)</sup>						
Rise Time	t <sub>R</sub>	C <sub>L</sub> = 50 pF	75	—	—	ns
Fall Time	t <sub>F</sub>	C <sub>L</sub> = 350 pF C <sub>L</sub> = 50 pF C <sub>L</sub> = 350 pF	75	—	300 — 300	
Rise/Fall Time Matching	t <sub>RFM</sub>	t <sub>R</sub> /t <sub>F</sub>	80	—	120	%
Output Signal Crossover Voltage	V <sub>CRS</sub>		1.3	—	2.0	V
Low Speed Data Rate	t <sub>DRATE</sub>	1.5 Mbs ± 1.5%	1.4775 676.8	1.500 666.0	1.5225 656.8	Mbs ns
Source Differential Driver Jitter To Next Transition For Paired Transitions	t <sub>UDJ1</sub> t <sub>UDJ2</sub>	C <sub>L</sub> = 350 pF Measured at Crossover Point	–25 –10	— —	25 10	ns
Receiver Data Jitter Tolerance To Next Transition For Paired Transitions	t <sub>DJR1</sub> t <sub>DJR2</sub>	C <sub>L</sub> = 350 pF Measured at Crossover Point	–75 –45	— —	75 45	ns
Source EOP Width	t <sub>EOP</sub>	Measured at Crossover Point	1.25	—	1.50	μs
Differential to EOP Transition Skew	t <sub>DEOP</sub>	Measured at Crossover Point	–40	—	100	ns
Receiver EOP Width Must Reject as EOP Must Accept	t <sub>EOPR1</sub> t <sub>EOPR2</sub>	Measured at Crossover Point	330 675	— —	— —	ns

1. All voltages are measured from local ground, unless otherwise specified. All timings use a capacitive load of 50 pF, unless otherwise specified. Low-speed timings have a 1.5-kΩ pullup to 2.8 V on the D– data line.
2. Transition times are measured from 10% to 90% of the data signal. The rising and falling edges should be smoothly transitioning (monotonic). Capacitive loading includes 50 pF of tester capacitance.

## 17.11 USB Signaling Levels

Bus State	Signaling Levels	
	From Originating Driver	At Receiver
Differential 1	$(D+) - (D-) > 200 \text{ mV}$ and $D+ \text{ or } D- > V_{SE} \text{ (min)}$	
Differential 0	$(D+) - (D-) < -200 \text{ mV}$ and $D+ \text{ or } D- > V_{SE} \text{ (min)}$	
Data J State Low Speed Full Speed	Differential 0 Differential 1	
Data K State Low Speed Full Speed	Differential 1 Differential 0	
Idle State Low Speed Full Speed	Differential 0 and $D- > V_{SE} \text{ (max)}$ and $D+ < V_{SE} \text{ (min)}$ Differential 1 and $D+ > V_{SE} \text{ (max)}$ and $D- < V_{SE} \text{ (min)}$	
Resume State Low Speed Full Speed	Differential 1 and $D+ > V_{SE} \text{ (max)}$ and $D- < V_{SE} \text{ (min)}$ Differential 0 and $D- > V_{SE} \text{ (max)}$ and $D+ < V_{SE} \text{ (min)}$	
Start of Packet (SOP)	Data lines switch from Idle to K State	
End of Packet (EOP)	$D+ \text{ and } D- < V_{SE} \text{ (min)}$ for 2 Bit Times <sup>(1)</sup> Followed by an Idle for 1 Bit Time	$D+ \text{ and } D- < V_{SE} \text{ (min)}$ for 1 Bit Time <sup>(2)</sup> Followed by a J State
Disconnect Upstream Only		$D+ \text{ and } D- < V_{SE} \text{ (max)}$ for 2.5 $\mu$ s
Connect Upstream Only		$D+ \text{ or } D- > V_{SE} \text{ (max)}$ for 2.5 $\mu$ s
Reset Downstream Only	$D+ \text{ and } D- < V_{SE}$ for 10 ms	$D+ \text{ and } D- < V_{SE} \text{ (min)}$ for 2.5 $\mu$ s; Must be Recognized within 5.5 $\mu$ s <sup>(3)</sup>

1. The width of EOP is defined in bit times relative to the speed of transmission.

2. The width of EOP is defined in bit times relative to the device type receiving the EOP.

3. These times apply to an active device that is not in the suspend state.

## 17.12 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Input Capture Pulse Width	$t_{TIH}, t_{TIL}$	125	—	ns
Input Clock Pulse Width	$t_{TCH}, t_{TCL}$	$(1/f_{OP}) + 5$	—	ns

## 17.13 Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
RAM Data Retention Voltage	$V_{RM}$	1.3	—	—	V





## Section 18. Mechanical Specifications

### 18.1 Contents

18.2 Introduction.....249

18.3 Quad Flat Pack (Case 848B-04).....250

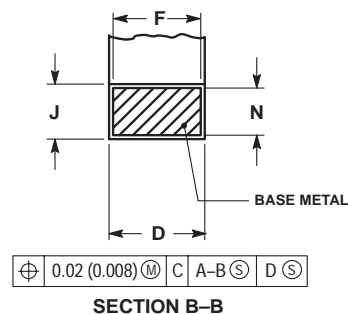
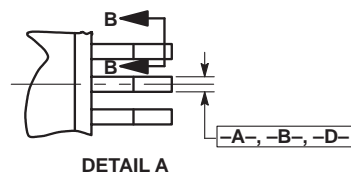
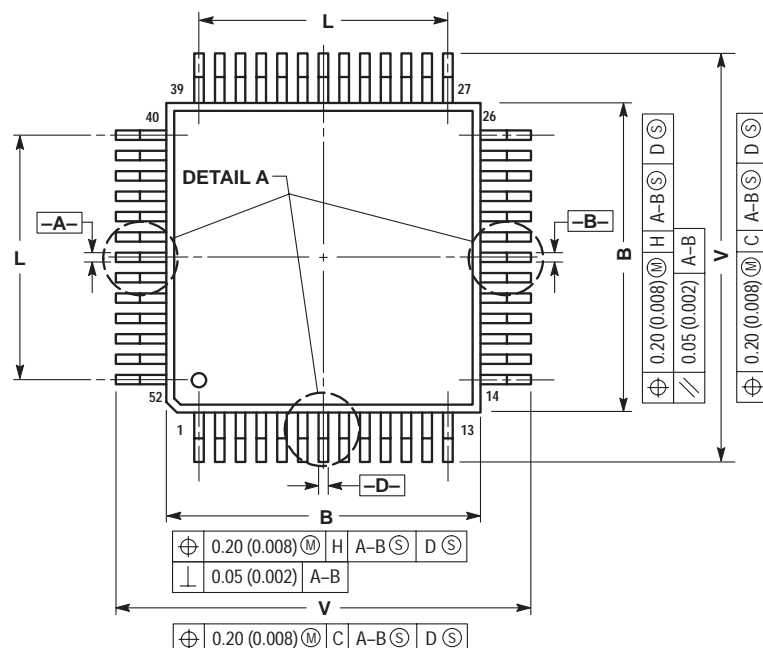
18.4 Shrink Dual In-Line Package (Case 858-01) .....251

### 18.2 Introduction

The MC68HC08KL8 is available in a 52-lead plastic quad flat pack (QFP) package and a 42-lead shrink dual in-line package (SDIP). This section gives the dimensions for these packages.

# Mechanical Specifications

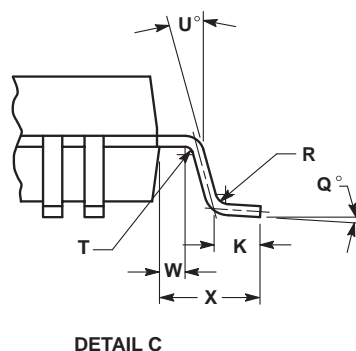
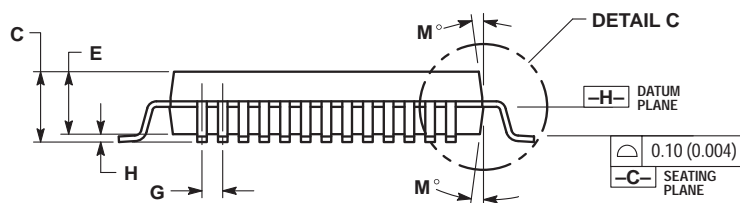
## 18.3 Quad Flat Pack (Case 848B-04)



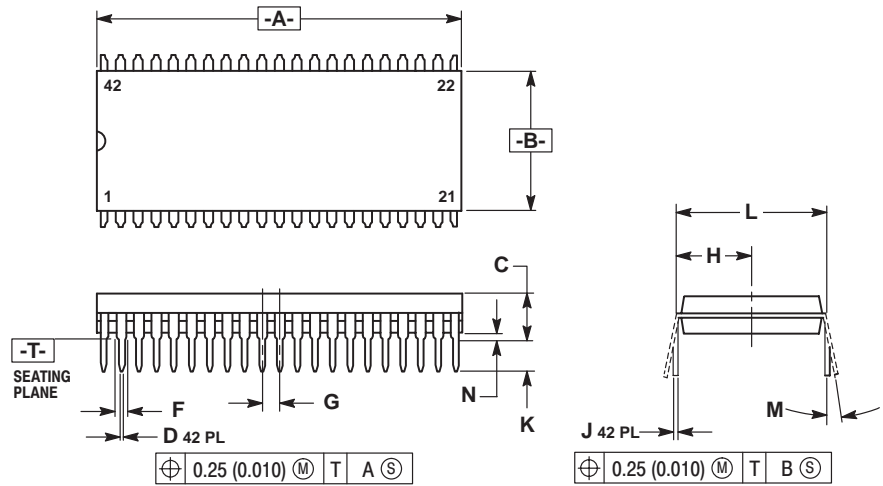
### NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS -A-, -B- AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	9.90	10.10	0.390	0.398
B	9.90	10.10	0.390	0.398
C	2.10	2.45	0.083	0.096
D	0.22	0.38	0.009	0.015
E	2.00	2.10	0.079	0.083
F	0.22	0.33	0.009	0.013
G	0.65 BSC		0.026 BSC	
H	—	0.25	—	0.010
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
L	7.80 REF		0.307 REF	
M	5°	10°	5°	10°
N	0.13	0.17	0.005	0.007
Q	0°	7°	0°	7°
R	0.13	0.30	0.005	0.012
S	12.95	13.45	0.510	0.530
T	0.13	—	0.005	—
U	0°	—	0°	—
V	12.95	13.45	0.510	0.530
W	0.35	0.45	0.014	0.018
X	1.6 REF		0.063 REF	



18.4 Shrink Dual In-Line Package (Case 858-01)



- NOTES:
1. DIMENSIONS AND TOLERANCING PER ANSI Y14.5M, 1982.
  2. CONTROLLING DIMENSION: INCH.
  3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
  4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010).

DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	1.435	1.465	36.45	37.21
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.56
F	0.032	0.046	0.81	1.17
G	0.070 BSC		1.778 BSC	
H	0.300 BSC		7.62 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.135	2.92	3.43
L	0.600 BSC		15.24 BSC	
M	0°	15°	0°	15°
N	0.020	0.040	0.51	1.02



Section 19. Ordering Information

19.1 Contents

19.2 Introduction.....253

19.3 MC Order Numbers .....253

19.2 Introduction

This section contains ordering information.

19.3 MC Order Numbers


Table 19-1. MC Order Numbers

MC Order Number	Operating Temperature Range
MC68HC08KL8FB <sup>(1)</sup>	0 °C to + 85 °C
MC68HC08KL8B <sup>(2)</sup>	0 °C to + 85 °C

1. FB = Quad Flat Pack
2. B = Shrink Dual In-Line Package





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

**How to reach us:**

**USA/EUROPE/Locations Not Listed:** Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217, 1-800-441-2447 or 1-303-675-2140. Customer Focus Center, 1-800-521-6274

**JAPAN:** Nippon Motorola Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan. 03-5487-8488

**ASIA/PACIFIC:** Motorola Semiconductors H.K. Ltd., 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

**Mfax™, Motorola Fax Back System:** RMFAX0@email.sps.mot.com; <http://sps.motorola.com/mfax/>;

TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848

**HOME PAGE:** <http://motorola.com/sps/>

Mfax is a trademark of Motorola, Inc.



**MOTOROLA**

© Motorola, Inc., 1998

**HC08KL8GRS/D**