HC08AZ32TS/D
Rev. 1.1

HC08AZ0
HC08AZ16
HC08AZ24
HC08AZ32
HC08AB0
HC08AB16
HC08AB24
HC08AB32

HCMOS Microcontroller Unit

TECHNICAL SUMMARY

MOTOROLA

HC08AZ32

HCMOS MICROCONTROLLER UNIT

TABLE OF CONTENTS

SECTION 4
ROM

SECTION 5
EEPROM

SECTION 6
CENTRAL PROCESSOR UNIT (CPU)

SECTION 7
SYSTEM INTEGRATION MODULE (SIM)

SECTION 8
CLOCK GENERATOR MODULE (CGM)

SECTION 9
MASK OPTIONS

SECTION 10
BREAK MODULE

SECTION 11
MONITOR ROM (MON)

SECTION 12
COMPUTER OPERATING PROPERLY MODULE

SECTION 13
LOW-VOLTAGE INHIBIT MODULE (LVI)

SECTION 14
EXTERNAL INTERRUPT MODULE (IRQ)

SECTION 15
SERIAL COMMUNICATIONS INTERFACE MODULE (SCI)

SECTION 16
SERIAL PERIPHERAL INTERFACE MODULE (SPI)

SECTION 17
TIMER INTERFACE MODULE A (TIMA)

SECTION 18
TIMER INTERFACE MODULE B (TIMB)

SECTION 19
PROGRAMMABLE INTERRUPT TIMER (PIT)

SECTION 20
ANALOG TO DIGITAL CONVERTOR (ADC)

SECTION 21
KEYBOARD MODULE (KB)

SECTION 22
I/O PORTS

SECTION 23
msCAN08 CONTROLLER

APPENDIX A
Devices Similar to the HC08AZ32

LIST OF FIGURES

LIST OF TABLES

SECTION 1
GENERAL DESCRIPTION

1.1 Introduction

The HC08AZ32 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.

1.2 Features

Features of the HC08AZ32 include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 families
- 8MHz internal bus frequency at 85˚C
- msCAN Controller (Motorola Scalable CAN) (implementing CAN 2.0b protocol as defined in BOSCH specification Sep. 1991)
- Available in 64 QFP package
- 32,255 bytes User ROM
- User ROM data security
- 512 bytes of on-chip EEPROM with security feature
- 1K byte of on-chip RAM
- Serial Peripheral Interface (SPI) module
- Serial Communications Interface (SCI) module
- 16-bit timer interface module (TIMA) with four input capture/output compare channels
- 16-bit timer interface module (TIMB) with two input capture/output compare channels
- Periodic Interrupt Timer (PIT)
- Clock Generator Module (CGM)
- 8-bit, 8-channel Analog to Digital Convertor module (ADC)

- 23 dedicated I/O lines
- 25 shared I/O lines
- 5-bit key wakeup port
- System protection features
  - Optional Computer Operating Properly (COP) reset
  - Low-voltage detection with optional reset
  - Illegal opcode detection with optional reset
  - Illegal address detection with optional reset (Non-Expanded mode)
- Low-power design (fully static with STOP and WAIT modes)
- Master reset pin and power-on reset

Features of the CPU08 include the following:

- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (8 more than the HC05)
- 16-Bit Index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-Coded Decimal (BCD) instructions
- Optimization for controller applications
- ‘C’ language support

Figure 1-1 shows the structure of the HC08AZ32.

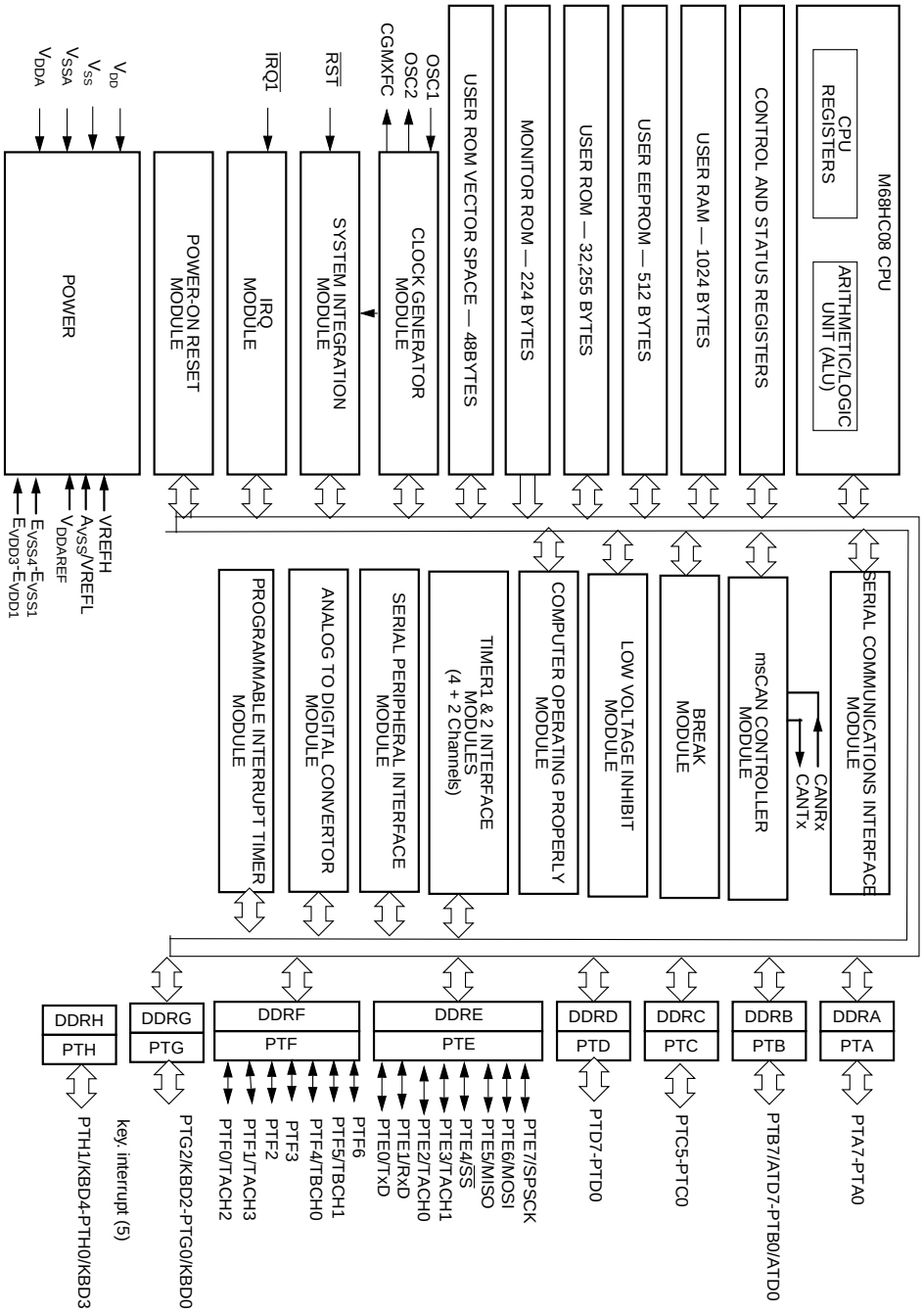


Figure 1-1. MCU block diagram

1.3 Pin Assignments

Figure 1-2 shows the 64 QFP pin assignments.

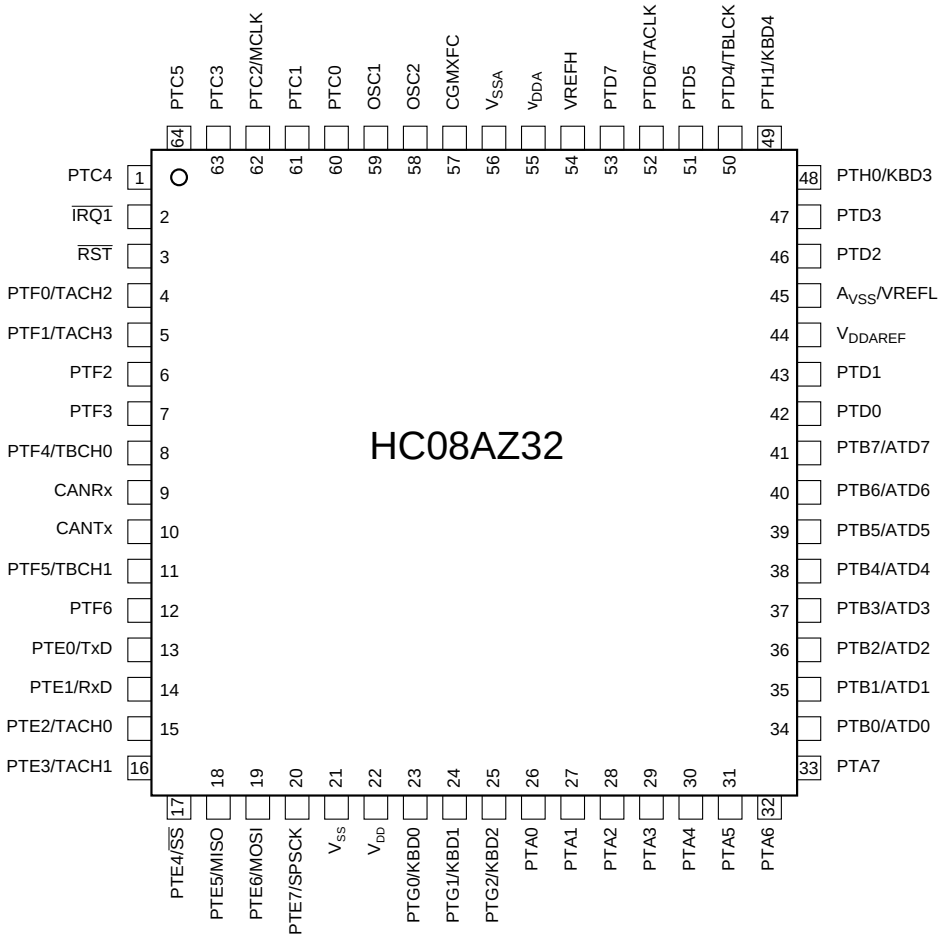


Figure 1-2 64 QFP pin assignments (top view)

1.3.1 Power supply pins (V_{DD} and V_{SS})

V_{DD} and V_{SS} 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 1-3 shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.

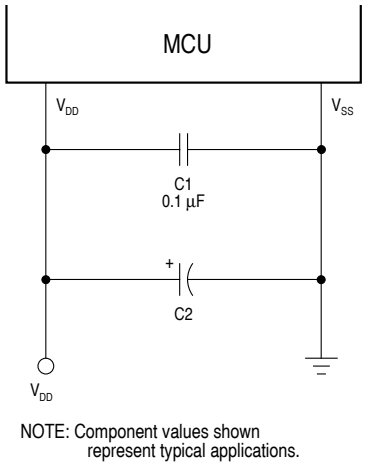


Figure 1-3 Power supply bypassing

V_{SS} is also the ground for the port output buffers and the ground return for the serial clock in the serial peripheral interface module (SPI). See **SECTION 16 SERIAL PERIPHERAL INTERFACE MODULE (SPI)**.

NOTE

V_{SS} must be grounded for proper MCU operation.

1.3.2 Oscillator pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**..

1.3.3 External reset pin ($\overline{RST}$)

A ’0’ on the $\overline{RST}$ pin forces the MCU to a known start-up state. $\overline{RST}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. See **SECTION 7 SYSTEM INTEGRATION MODULE (SIM)**.

1.3.4 External interrupt pin ($\overline{IRQ1}$)

$\overline{IRQ1}$ is an asynchronous external interrupt pin. See **SECTION 14 EXTERNAL INTERRUPT MODULE (IRQ)**.

1.3.5 Analog power supply pin (V_{DDA})

V_{DDA} is the power supply pin for the clock generator module (CGM).

1.3.6 Analog ground pin (V_{SSA})

The V_{SSA} analog ground pin is used only for the ground connections for the clock generator module (CGM) section of the circuit and should be decoupled as per the V_{SS} digital ground pin. See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**..

1.3.7 Analog ground pin (A_{VSS}/VREFL)

The A_{VSS} analog ground pin is used only for the ground connections for the analog to digital convertor (ADC) and should be decoupled as per the VSS digital ground pin.

1.3.8 ADC voltage reference pin (VREFH)

VREFH is the power supply for setting the reference voltage VREFH. Connect the VREFH pin to a voltage potential<= V_{DDAREF}, not less than 1.5V.

1.3.9 Analog supply pin (V_{DDAREF})

The V_{DDAREF} analog supply pin is used only for the supply connections for the analog to digital convertor (ADC).

1.3.10 External filter capacitor pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.

1.3.11 Port A input/output (I/O) pins (PTA7–PTA0)

PTA7–PTA0 are general-purpose bidirectional I/O port pins. See **SECTION 22 I/O PORTS**.

1.3.12 Port B I/O pins (PTB7/ATD7–PTB0/ATD0)

Port B is an 8-bit special function port that shares all eight pins with the analog to digital convertor (ADC). See **SECTION 20 ANALOG TO DIGITAL CONVERTOR (ADC)** and **SECTION 22 I/O PORTS**.

1.3.13 Port C I/O pins (PTC5–PTC0)

PTC5–PTC3 and PTC1–PTC0 are general-purpose bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with the system clock. See **SECTION 22 I/O PORTS**.

1.3.14 Port D I/O pins (PTD7–PTD0)

Port D is an 8-bit special function port that shares two of its pins with the timer interface modules (TIMA and TIMB). See **SECTION 17 TIMER INTERFACE MODULE A (TIMA)** and **SECTION 18 TIMER INTERFACE MODULE B (TIMB)**.

1.3.15 Port E I/O pins (PTE7/SPSCK–PTE0/TxD)

Port E is an 8-bit special function port that shares two of its pins with the timer interface module (TIMA), four of its pins with the Serial Peripheral Interface Module (SPI), and two of its pins with the Serial Communication Interface Module (SCI). See **SECTION 15 SERIAL COMMUNICATIONS INTERFACE MODULE (SCI)**, **SECTION 16 SERIAL PERIPHERAL INTERFACE MODULE (SPI)**, See **SECTION 17 TIMER INTERFACE MODULE A (TIMA)** and **SECTION 22 I/O PORTS**.

1.3.16 Port F I/O pins (PTF6–PTF0/TACH2)

Port F is a 7-bit special function port that shares four of its pins with the timer interface modules. See **SECTION 17 TIMER INTERFACE MODULE A (TIMA)**, **SECTION 18 TIMER INTERFACE MODULE B (TIMB)** and **SECTION 22 I/O PORTS**.

1.3.17 Port G I/O pins (PTG2/KBD2–PTG0/KBD0)

PTG2/KBD2–PTG0/KBD0 are general-purpose bidirectional I/O pins with Key Wakeup feature. See **SECTION 21 KEYBOARD MODULE (KB)** and **SECTION 22 I/O PORTS**.

1.3.18 Port H I/O pins (PTH1/KBD4–PTH0/KBD3)

PTH1/KBD4–PTH0/KBD3 are general-purpose bidirectional I/O pins with Key Wakeup feature. See **SECTION 21 KEYBOARD MODULE (KB)** and **SECTION 22 I/O PORTS**.

1.3.19 CAN transmit pin (CANTx)

CANTx is the digital output from the msCAN module. See **SECTION 23 msCAN08 CONTROLLER**.

1.3.20 CAN receive pin (CANRx)

CANRx is the digital input to the msCAN module. See **SECTION 23 msCAN08 CONTROLLER**.

Table 1-1 External pins summary

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
PTA7 - PTA0	General purpose I/O	Dual State	No	Input (Hi-Z)
PTB7/ATD7 - PTB0/ATD0	General purpose I/0 / ADC channel	Dual State	No	Input (Hi-Z)
PTC5 - PTC0	General purpose I/O	Dual State	No	Input (Hi-Z)
PTD7	General purpose I/O	Dual State	No	Input (Hi-Z)
PTD6/TACLK	General purpose I/O / Timer External Input clock	Dual State	No	Input (Hi-Z)
PTD5	General purpose I/O/ Timer External Input clock	Dual State	No	Input (Hi-Z)
PTD4/TBLCK-PTD0	General purpose Input	Dual State	No	Input (Hi-Z)
PTE7/SPSCK	General purpose I/0 / SPI clock	Dual State (open drain)	Yes	Input (Hi-Z)
PTE6/MOSI	General purpose I/0 / SPI data path	Dual State (open drain)	Yes	Input (Hi-Z)
PTE5/MISO	General purpose I/0 / SPI data path	Dual State (open drain)	Yes	Input (Hi-Z)
PTE4/$\overline{SS}$	General purpose I/0 / SPI Slave Select	Dual State	Yes	Input (Hi-Z)
PTE3/TACH1	General purpose I/0 / Timer A channel 1	Dual State	Yes	Input (Hi-Z)
PTE2/TACH0	General purpose I/0 / TimerA channel 0	Dual State	Yes	Input (Hi-Z)
PTE1/RxD	General purpose I/0 / SCI Receive Data	Dual State	Yes	Input (Hi-Z)
PTE0/TxD	General purpose I/0 / SCI Transmit Data	Dual State	Yes	Input (Hi-Z)
PTF6	General purpose I/O	Dual State	Yes	Input (Hi-Z)

Table 1-1 External pins summary

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
PTF5/TBCH1	General purpose I/O /Timer B channel 1	Dual State	Yes	Input (Hi-Z)
PTF4/TBCH0	General purpose I/0 / TimerB channel 0	Dual State	Yes	Input (Hi-Z)
PTF3	General purpose I/0	Dual State	Yes	Input (Hi-Z)
PTF2	General purpose I/0	Dual State	Yes	Input (Hi-Z)
PTF1/TACH3	General purpose I/0 /TimerA channel 3	Dual State	Yes	Input (Hi-Z)
PTF0/TACH2	General purpose I/0 /TimerA channel 2	Dual State	Yes	Input (Hi-Z)
PTG2/KBD2 - PTG0/KBD0	General purpose I/0 with key wakeup feature	Dual State	Yes	Input (Hi-Z)
PTH1/KBD4- PTH0/KBD3	General purpose I/0 with key wakeup feature	Dual State	Yes	Input (Hi-Z)
V_{DD}	Logical chip power supply	NA	NA	NA
V_{SS}	Logical chip ground	NA	NA	NA
V_{DDA}	Analog power supply(CGM)	NA	NA	NA
V_{SSA}	Analog ground (CGM)	NA	NA	NA
V_{REFH}	ADC reference voltage	NA	NA	NA
A_{VSS}/VREFL	ADC gnd & reference voltage	NA	NA	NA
V_{DDAREF}	ADC power supply	NA	NA	NA
OSC1	External clock in	NA	NA	Input (Hi-Z)
OSC2	External clock out	NA	NA	Output
CGMXFC	PLL loop filter cap	NA	NA	NA
$\overline{IRQ1}$	External interrupt request	NA	NA	Input (Hi-Z)
$\overline{RST}$	Reset	NA	NA	Input (Hi-Z)
CANRx	msCAN serial Input	NA	YES	Input (Hi-Z)
CANTx	msCAN serial output	Output	NA	Output

Details of the clock connections to each of the modules on the HC08AZ32 are shown in Table 1-3. A short description of each clock source is also given in Table 1-2.

Table 1-2 Signal name conventions

Signal name	Description
CGMXCLK	Buffered version of OSC1 from clock generator module (CGM)
CGMOUT	PLL-based or OSC1-based clock output from CGM module)
Bus clock	CGMOUT divided by two
SPSCK	SPI serial clock (see **16.12.3 SPSCK (serial clock)**)
TACLK	External clock Input for TIMA (see **17.7.1 TIMA clock pin (PTD6/TACLK)**)
TBCLK	External clock Input for TIMB (see **18.7.1 TIMB clock Pin (PTD4/TBLCK)**)

Table 1-3 Clock source summary

Module	Clock source
ADC	CGMXCLK or bus clock
msCAN	CGMXCLK or CGMOUT
COP	CGMXCLK
CPU	Bus clock
EEPROM	CGMXCLK or bus clock
ROM	Bus clock
RAM	Bus clock
SPI	SPSCK
SCI	CGMXCLK
TIMA	Bus clock or PTD6/TACLK
TIMB	Bus clock or PTD4/TBLCK
PIT	Bus clock
KBI	Bus clock

SECTION 2
MEMORY MAP

2.1 Introduction

The CPU08 can address 64K bytes of memory space. The memory map includes:

- 1024 bytes of RAM
- 32,255 bytes of User ROM
- 512 bytes of EEPROM
- 48 bytes of user-defined vectors
- 224 bytes of monitor ROM

2.2 Memory section

The HC08AZ32 operates in Non-Expanded mode, where the full 32,255 bytes of User ROM are available.

2.3 I/O section

Addresses $0000–$004F, shown in Figure 2-2, contain most of the control, status, and data registers. Additional I/O registers have the following addresses:

- $0500 to $057F – CAN control and message buffers. See **SECTION 23 msCAN08 CONTROLLER**.
- $FE00 – (SIM break status register, SBSR)
- $FE01 – (SIM reset status register, SRSR)
- $FE03 – (SIM break flag control register, SBFCR)
- $FE07 – (EPROM control register, EPMCR)
- $FE0C and $FE0D – (break address registers, BRKH and BRKL)
- $FE0E – (break status and control register, BRKSCR)
- $FE0F – (LVI status register, LVISR)
- $FE1C – (EEPROM non-volatile register, EENVR)
- $FE1D – (EEPROM control register, EECR)
- $FE1F – (EEPROM array configuration register, EEACR)
- $FFFF – (COP control register, COPCTL)

$0000 ↓ $004F	I/O REGISTERS (80 BYTES)
$0050 ↓ $044F	RAM (1024 BYTES)
$0450 ↓ $04FF	UNIMPLEMENTED (176 BYTES)
$0500 ↓ $057F	CAN CONTROL AND MESSAGE BUFFERS(128 BYTES)
$0580 ↓ $07FF	UNIMPLEMENTED (640 BYTES)
$0800 ↓ $09FF	EEPROM (512 BYTES)
$0A00 ↓ $0FFF	UNIMPLEMENTED (1536 BYTES)
$1000 ↓ $7FFF	UNIMPLEMENTED (28,672 BYTES)
$8000 ↓ $BFFF	ROM (16,384BYTES)
$C000 ↓ $FDFF	ROM (15,872 BYTES)
$FE00	SIM BREAK STATUS REGISTER (SBSR)
$FE01	SIM RESET STATUS REGISTER (SRSR)
$FE02	RESERVED
$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04	RESERVED
$FE05	RESERVED
$FE06	UNIMPLEMENTED
$FE07	RESERVED
$FE08	RESERVED
$FE09	RESERVED

Figure 2-1 Memory map

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (48 BYTES)

Figure 2-1 Memory map

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	R:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		W:								
$0001	Port B Data Register (PTB)	R:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		W:								
$0002	Port C Data Register	R:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		W:								
$0003	Port D Data Register (PTD)	R:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		W:								
$0004	Data Direction Register A (DDRA)	R:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		W:								
$0005	Data Direction RegisterB (DDRB)	R:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		W:								
$0006	Data Direction Register C (DDRC)	R:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		W:								
$0007	Data Direction Register D (DDRD)	R:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		W:								
$0008	Port E Data Register (PTE)	R:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		W:								
$0009	Port F Data Register (PTF)	R:	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		W:								
$000A	Port G Data Register (PTG)	R:	0	0	0	0	0	PTG2	PTG1	PTG0
		W:								
$000B	Port H Data Register (PTH)	R:	0	0	0	0	0	0	PTH1	PTH0
		W:								
$000C	Data Direction Register E (DDRE)	R:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		W:								
$000D	Data Direction Register F (DDRF)	R:	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		W:								
$000E	Data Direction Register G (DDRG)	R:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
		W:								
$000F	Data Direction Register (DDRH)	R:	0	0	0	0	0	0	DDRH1	DDRH0
		W:								
$0010	SPI Control Register (SPCR)	R:	SPRIE	DMAS	SP-MSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		W:								
$0011	SPI Status and Control Register (SPSCR)	R:	SPRF	0	OVRF	MODF	SPTE	0	SPR1	SPR0
		W:								
$0012	SPI Data Register (SPDR)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0013	SCI Control Register 1 (SCC1)	R:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		W:								
$0014	SCI Control Register 2 (SCC2)	R:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		W:								
$0015	SCI Control Register 3 (SCC3)	R:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		W:								
$0016	SCI Status Register 1 (SCS1)	R:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		W:								

= Unimplemented R = Reserved

Figure 2-2 Control, status, and data registers

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0017	SCI Status Register 2 (SCS2)	R:	0	0	0	0	0	0	BKF	RPF
		W:								
$0018	SCI Data Register (SCDR)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0019	SCI Baud Rate Register (SCBR)	R:	0	0	SCP1	SCP0	0	SCR2	SCR1	SCR0
		W:								
$001A	IRQ Status and Control Register (ISCR)	R:					IRQ1F	0	IMASK1	MODE1
		W:						ACK1		
$001B	Keyboard Status/Control (KBSCR)	R:	0	0	0	0	KEYF	0	IMASKK	MODEK
		W:						ACKK		
$001C	PLL Control Register (PCTL)	R:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		W:								
$001D	PLL Bandwidth Control Register (PBWC)	R:	AUTO	LOCK	ACQ	XLD	0	0	0	0
		W:								
$001E	PLL Programming Register (PPG)	R:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		W:								
$001F	Mask Option Register A (MORA)	R:		SEC	LVIRSTD	LVIPWRD	SSREC	COPRS	STOP	COPD
		W:								
$0020	TimerA Status and Control Register (TASC)	R:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		W:	0			TRST				
$0021	Keyboard Interrupt Enable Register (KBIER)	R				KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		W								
$0022	TimerA Counter Register High (TACNTH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0023	TimerA Counter Register Low (TACNTL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0024	TimerA Modulo Register High (TAMODH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0025	TimerA Modulo Register Low (TAMODL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0026	Timer A Channel 0 Status and Control Register (TASC0)	R:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		W:	0							
$0027	TimerA Channel 0 Register High (TACH0H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0028	Timer A Channel 0 Register Low (TACH0L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0029	Timer A Channel 1 Status and Control Register (TASC1)	R:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		W:	0							
$002A	Timer A Channel 1 Register High (TACH1H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002B	Timer A Channel 1 Register Low (TACH1L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002C	Timer A Channel 2 Status and Control Register (TASC2)	R:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		W:	0							

= Unimplemented R = Reserved

Figure 2-2 Control, status, and data registers (Continued)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	Timer A Channel 2 Register High (TACH2H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002E	Timer A Channel 2 Register Low (TACH2L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002F	Timer Channel 3 Status/ Control Register (TASC3)	R:	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		W:	0							
$0030	Timer Channel 3 Register High (TACH3H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0031	Timer Channel 3 Register Low (TACH3L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0032	Unimplemented	R:								
		W:								
$0033	Unimplemented	R:								
		W:								
$0034	Unimplemented	R:								
		W:								
$0035	Unimplemented	R:								
		W:								
$0036	Unimplemented	R:								
		W:								
$0037	Unimplemented	R:								
		W:								
$0038	ADC Status/Control Register (ADSCR)	R:	COCO	AIEN	ADCO	CH4	CH3	CH2	CH1	CH0
		W:								
$0039	ADC Data Register (ADR)	R:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		W:								
$003A	ADC Clock Register (ADCLKR)	R:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		W:								
$003B	EBI Control Register (EBIC)	R:	0	IRV	MODE	CS0WS	WSCLK1	WSCLK0	CSC1	CSC0
		W:								
$003C	EBI Chip Select Register (EBICS)	R:	CS1WS1	CS1WS0	CS1POL	CS1EN	CS0WS1	CS0WS0	CS0POL	CS0EN
		W:								
$003D	Unimplemented	R:								
		W:								
$003E	Unimplemented	R:								
		W:								
$003F	Mask Option Register B (MORB)	R:	R	R	EESEC	R	R	R	R	R
		W:								
$0040	TimerB Status and Control Register (TBSC)	R:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		W:	0			TRST				
$0041	TimerB Counter Register High (TBCNTH)	R:	Bit 15	14	13	12	11	10	9	8
		W:								
$0042	TimerB Counter Register Low (TBCNTL)	R:	Bit 7	6	5	4	3	2	1	0
		W:								
$0043	TimerB Modulo Register High (TBMODH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								

= Unimplemented R = Reserved

Figure 2-2 Control, status, and data registers (Continued)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0044	TimerB Modulo Register Low (TBMODL)	R: W:	Bit 7	6	5	4	3	2	1	Bit 0
$0045	Timer B Channel 0Status and Control Register (TBSC0)	R: W:	CH4F 0	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH0MAX
$0046	Timer B Channel 0Register High (TBCH0H)	R: W:	Bit 15	14	13	12	11	10	9	8
$0047	Timer B Channel 0Register Low (TBCH0L)	R: W:	Bit 7	6	5	4	3	2	1	0
$0048	Timer B Channel 1Status/ Control Register (TBSC1)	R: W:	CH5F 0	CH5IE	MS5B	MS5A	ELS5B	ELS5A	TOV5	CH1MAX
$0049	Timer B Channel 1Register High (TBCH1H)	R: W:	Bit 15	14	13	12	11	10	9	8
$004A	Timer B Channel1Register Low (TBCH1L)	R: W:	Bit 7	6	5	4	3	2	1	0
$004B	Programmable Interrupt Timer Status & Control Register (PSC)	R: W:	POF 0	PIE	PSTOP	0 PRST	0	PPS2	PPS1	PPS0
$004C	PIT Counter Register HIGH) (PCNTH)	R: W:	Bit 15	14	13	12	11	10	9	8
$004D	PIT Counter Register Low (PCNTL)	R: W	7	6	5	4	3	2	1	0
$004E	PIT Modulo Register High (PMODH)	R W	Bit 15	14	13	12	11	10	9	8
$004F	PIT Modulo Register Low (PMODL)	R: W	7	6	5	4	3	2	1	0
$FE00	SIM Break Status Register (SBSR)	R: W:	R	R	R	R	R	R	SBSW	R
$FE01	SIM Reset Status Register (SRSR)	R: W:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
$FE03	SIM Break Flag Control Register (SBFCR)	R: W:	BCFE	R	R	R	R	R	R	R
$FE07	Reserved	R: W:								
$FE0C	Break Address Register High (BRKH)	R: W:	Bit 15	14	13	12	11	10	9	Bit 8
$FE0D	Break Address Register Low (BRKL)	R: W:	Bit 7	6	5	4	3	2	1	Bit 0
$FE0E	Break Status and Control Register (BRKSCR)	R: W:	BRKE	BRKA	0	0	0	0	0	0
$FE0F	LVI Status Register (LVISR)	R: W:	LVIOUT	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 2-2 Control, status, and data registers (Continued)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE1C	EENVR	R:	EERA	CON2	CON1	CON0	EEPB3	EEPB2	EEPB1	EEPB0
		W:								
$FE1D	EECR	R:	EEBCLK	0	EEOFF	EERAS1	EERAS0	ELAT	0	EEPGM
		W:								
$FE1E	Reserved	R:								
		W:								
$FE1F	EEACR	R:	EERA	CON2	CON1	CON0	EEBP3	EEBP2	EEBP1	EEBP0
		W:								

$FFFF	COP Control Register (COPCTL)	R:	LOW BYTE OF RESET VECTOR
		W:	WRITING TO $FFFF CLEARS COP COUNTER

= Unimplemented R = Reserved

Figure 2-2 Control, status, and data registers (Continued)

Table 2-1 Vector addresses

Low

Priority

Address	Vector
$FFD0	ADC vector (high)
$FFD1	ADC vector (low)
$FFD2	Keyboard vector (high)
$FFD3	Keyboard vector (low)
$FFD4	SCI transmit vector (high)
$FFD5	SCI Transmit vector (Low)
$FFD6	SCI Receive vector (High)
$FFD7	SCI Receive vector (Low)
$FFD8	SCI Error vector (High)
$FFD9	SCI Error vector (Low)
$FFDA	msCAN Transmit vector(High)
$FFDB	msCAN Transmit vector (Low)
$FFDC	msCAN Receive vector(High)
$FFDD	msCAN Receive vector (Low)
$FFDE	msCAN Error vector(High)
$FFDF	msCAN Error vector (Low)
$FFE0	msCAN Wakeup vector(High)
$FFE1	msCAN Wakeup vector (Low)
$FFE2	SPI Transmit vector(High)
$FFE3	SPI Transmit vector (Low)
$FFE4	SPI Receive vector(High)
$FFE5	SPI Receive vector (Low)
$FFE6	TIMB Overflow vector(High)
$FFE7	TIMB Overflow vector (Low)
$FFE8	TIMB CH1 vector(High)
$FFE9	TIMB CH1 vector (Low)
$FFEA	TIMB CH0 vector(High)
$FFEB	TIMB CH0 vector (Low)
$FFEC	TIMA Overflow vector(High)
$FFED	TIMA Overflow vector (Low)
$FFEE	TIMA CH3 vector(High)
$FFEF	TIMA CH3 vector (Low)
$FFF0	TIMACH2 vector(High)
$FFF1	TIMA CH2 vector (Low)
$FFF2	TIMA CH1 vector(High)
$FFF3	TIMA CH1 vector (Low)

Table 2-1 Vector addresses (Continued)

	Address	Vector
Priority ↑	$FFF4	TIMA CH0 vector(High)
	$FFF5	TIMA CH0 vector (Low)
	$FFF6	PIT vector(High)
	$FFF7	PIT vector (Low)
	$FFF8	PLL vector(High)
	$FFF9	PLL vector (Low)
	$FFFA	IRQ1 vector(High)
	$FFFB	IRQ1 vector (Low)
	$FFFC	SWI vector(High)
↓	$FFFD	SWI vector (Low)
High	$FFFE	Reset vector (High)
	$FFFF	Reset vector (Low)

SECTION 3
RAM

3.1 Introduction

This section describes the 1024 bytes of RAM.

3.2 Functional description

Addresses $0050 through $044F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64K byte memory space.

NOTE

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

Within page zero there are 176 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 efficiently access all page zero RAM locations. Page zero RAM, therefore, provides an ideal location for frequently accessed global variables.

Before processing an interrupt, the CPU uses 5 bytes of the stack to save the contents of the CPU registers.

NOTE

For M6805 compatibility, the H register is not stacked.

During a subroutine call, the CPU uses 2 bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

NOTE

Care should be taken when using nested subroutines. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.

3

SECTION 4
ROM

4

4.1 Introduction

This section describes the 32,255 bytes of user read-only-memory (ROM), 224 bytes of Monitor ROM, and 48 bytes of user vectors.

4.2 Functional description

The user ROM consists of up to 32,255 bytes depending on whether the device is operated in Non-Expanded or Expanded mode.

4.3 User ROM

In Non-Expanded mode (64QFP) the user ROM consists of 32,272 bytes from addresses $8000–$FDFF and $FFC0–$FFCF. The monitor ROM and vectors are located from $FE20–$FEFF.

Forty of the user vectors, $FFD0–$FFFF, are dedicated to user-defined reset and interrupt vectors.

4.4 Security

Security has been incorporated into the HC08AZ32 to prevent external viewing of the ROM contents. This feature is selected by a mask option and ensures that customer-developed software remains propriety. See **SECTION 9 MASK OPTIONS**.

4

SECTION 5
EEPROM

5.1 Introduction

This section describes the electrically eraseable programmable ROM (EEPROM).

5.2 Features

- Modular architecture expandable in 128 bytes
- Byte, block or bulk erasable
- Non-volatile redundant array option
- Non-volatile block protection option
- Non-volatile MCU configuration bits
- On-chip charge pump for programming/erasing.
- Security option

5.3 Functional description

512 bytes of EEPROM can be programmed or erased without an external voltage supply. The EEPROM has a lifetime of 10,000 write-erase cycles in the non-redundant mode. Reliability (data retention) is further extended if the redundancy option is selected. EEPROM cells are protected with a non-volatile block protection option. These options are stored in the EEPROM non-volatile register (EENVR) and are loaded into the EEPROM array configuration register after reset (EEACR) or after a read of EENVR. Hardware interlocks are provided to protect stored data corruption from accidental programming/erasing.

5.3.1 EEPROM programming

The unprogrammed state is a logic ‘1’. Programming changes the state to a logic ‘0’. Only valid EEPROM bytes in the non-protected blocks and EENVR can be programmed. When the array is configured in the redundant mode, programming the first 256 bytes will also program the last 256 bytes with the same data. It is recommended that the EEPROM be programmed in the non-redundant mode and that the data is programmed to both locations before entering the redundant mode.

The following procedure describes how to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0 and set EELAT in the EECR (see notes A and B).
2. Write the desired data to any user EEPROM address.
3. Set the EEPGM bit. (See note C.)
4. Wait for a time, t_{EEPGM}, to program the byte.
5. Clear EEPGM bit.
6. Wait for the programming voltage time to fall (t_{EEFPV}).
7. Clear EELAT bits. (See note D.)
8. Repeat steps 1 to 7 for more EEPROM programming.

NOTES:

A. EERAS1 and EERAS0 must be cleared for programming.

B. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM address will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. Any attempts to read other EEPROM data will result in the latched data being read. If EELAT is set, other writes to the EECR will be allowed after a valid EEPROM write.

C. The EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. This is to ensure proper programming sequence. When EEPGM is set, the on-board charge pump generates the program voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the program voltage is removed from the array and the internal charge pump is turned off.

D. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.

5.3.2 EEPROM erasing

The unprogrammed state is a logic ’1’. Only the valid EEPROM bytes in the non-protected blocks and EENVR can be erased. When the array is configured in the redundant mode, erasing the first 256 bytes will also erase the last 256 bytes.

The following procedure shows how to erase EEPROM:

1. Clear/set EERAS1 and EERAS0 to select byte/block/bulk erase, and set EELAT in EECR (see note A).
2. Write any data to the desired address for byte erase, to any address in the desired block for block erase, or to any array address for bulk erase.
3. Set the EEPGM bit. (See note B.)
4. Wait for a time, $t_{byte}/t_{block}/t_{bulk}$ before erasing the byte/block/bulk.

5. Clear EEPGM bit.
6. Wait for the erasing voltage time to fall (t_{EEFPV}).
7. Clear EELAT bits. (See note C.)
8. Repeat steps 1 to 7 for more EEPROM byte/block erasing.

The EEBPx bit must be cleared to erase EEPROM data in the corresponding block. If any EEBPx is set, the corresponding block cannot be erased and bulk erase mode does not apply.

NOTES:

A. Setting the EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM addresses with its data will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. In block erase mode, any EEPROM address in the block may be used in step 2. All locations within this block will be erased. In bulk erase mode, any EEPROM address may be used to erase the whole EEPROM. EENVR is not affected with block or bulk erase. Any attempts to read other EEPROM data will result in the latched data being read. If EELAT is set, other writes to the EECR will be allowed after a valid EEPROM write.

B. The EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. This is to ensure proper erasing sequence. Once EEPGM is set, the type of erase mode cannot be modified. If EEPGM is set, the on-board charge pump generates the erase voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the erase voltage is removed from the array and the internal charge pump is turned off.

C. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.

In general, all bits should be erased before being programmed. However, if program/erase cycling is of concern, the following procedure can be used to minimize bit cycling in each EEPROM byte. If any bit in a byte requires to be changed from a ’0’ to a ‘1’, the byte needs be erased before programming. Table 5-1 summarizes the conditions for erasing before programming.

EEPROM Data To Be Programmed	EEPROM Data Before Programming	Erase Before Programming?
0	0	No
0	1	No
1	0	Yes
1	1	No

Table 5-1 EEPROM program/erase cycling reduction

5.3.3 EEPROM block protection

The 512 bytes of EEPROM is divided into four 128 byte blocks. Each of these blocks can be separately protected by the EEBPx bit. Any attempt to program or erase memory locations within the protected block will not allow the program/erase voltage to be applied to the array. Table 5-2 shows the address ranges within the blocks.

Block number (EEBPx)	Address range
EEBP0	$0800-$087F
EEBP1	$0880-$08FF
EEBP2	$0900-$097F
EEBP3	$0980-$09FF

Table 5-2 EEPROM array address blocks

If the EEBPx bit is set, the corresponding address block is protected. These bits are effective after a reset or a read to EENVR register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EENVR register and then reading the EENVR register.

In redundant mode, EEBP3 and EEBP2 have no meaning.

5.3.4 EEPROM redundant mode

To extend the EEPROM data retention, the array can be placed in redundant mode. In this mode, the first 256 bytes of user EEPROM array is mapped to the last 256 bytes. Reading, programming and erasing of the first 256 EEPROM bytes will physically affect two bytes of EEPROM. Addressing the last 256 bytes will not

be recognized. Block protection still applies but EEBP3 and EEBP2 are meaningless.

NOTE

It is recommended that the EEPROM be programmed in the non-redundant mode and the data programmed to its corresponding location before entering the redundant mode.

5.3.5 EEPROM configuration

The EEPROM non-volatile register (EENVR) contains configurations concerning block protection and redundancy. EENVR is physically located on the bottom of the EEPROM array. The contents are non-volatile and are not modified by reset. On reset, this special register loads the EEPROM configuration into a corresponding volatile EEPROM array configuration register (EEACR). Thereafter, all reads to the EENVR will result in EEACR being reloaded.

The EEPROM configuration can be changed by programming/erasing the EENVR like a normal EEPROM byte. The new array configuration will take effect with a system reset or a read of the EENVR.

5.3.6 MCU configuration

The EEPROM non-volatile register (EENVR) also contains general purpose bits which can be used to enable/disable functions within the MCU which, for safety reasons, need to be controlled from non-volatile memory. On reset, this special register loads the MCU configuration into the volatile EEPROM array configuration register (EEACR). Thereafter, all reads to the EENVR will result in EEACR being reloaded.

The MCU configuration can be changed by programming/erasing the EENVR like a normal EEPROM byte. The new array configuration will take effect with a system reset or a read of the EENVR.

5.3.7 HC08AZ32 EEPROM security

The **HC08AZ32** has a special security option which prevents program/erase access to memory locations $08F0 to $08FF. This security function is enabled by:

- Choosing the EEPROM security mask option See **SECTION 9 MASK OPTIONS**.
- Programming the CON0 bit in the EENVR to ‘0’.

Once armed, the security is permanently enabled. A consequence of this is that all functions in the EENVR will remain in the state they were in immediately before the security was enabled.

Once the security is armed ‘Bulk’ and ‘Block’ erase modes are disabled. ‘Byte’ erasing can be used for all locations except $08F0 to $08FF. These protected locations can be read as normal.

5

5.3.8 EEPROM control register (EECR)

This read/write register controls programming/erasing of the array.

		7	6	5	4	3	2	1	0
EECR $FE1D	READ:	EEBCLK	0	EEOFF	EERAS1	EERAS0	EELAT	0	EEPGM
	WRITE:								
	RESET:	0	0	0	0	0	0	0	0

Figure 5-1 EEPROM control register (EECR)

EEBCLK - EEPROM BUS CLOCK ENABLE

This read/write bit determines which clock will be used to drive the internal charge pump for programming/erasing. Reset clears this bit.

1 = Bus clock drives charge pump
0 = Internal RC oscillator drives charge pump

NOTE

It is recommended that the internal RC oscillator be used for applications in the 3-5V range.

EEOFF - EEPROM power down

This read/write bit disables the EEPROM module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM array
0 = Enable EEPROM array

NOTE

The EEPROM requires a recovery time t_{EEOFF} to stabilize after clearing the EEOFF bit.

EERAS1 – EERAS0 — Erase bits

These read/write bits set the erase modes. Reset clears these bits.

EEBPx	EERAS1	EERA0	Mode
0	0	0	Byte Program
0	0	1	Byte Erase
0	1	0	Block Erase
0	1	1	Bulk Erase
1	X	X	No Erase/Program

X = don’t care

Table 5-3 EEPROM program/erase mode select

EELAT — EEPROM latch control

This read/write bit latches the address and data buses for programming the EEPROM array. EELAT can not be cleared if EEPGM is still set. Reset clears this bit.

1 = Buses configured for EEPROM programming
0 = Buses configured for normal read operation

EEPGM — EEPROM program/erase enable

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM array if the EELAT bit is set and a write to a valid EEPROM location has occurred. Reset clears the EEPGM bit.

1 = EEPROM programming/erasing power switched on
0 = EEPROM programming/erasing power switched off

NOTE

Writing ‘0’s to both the EELAT and EEPGM bits with a single instruction will only clear EEPGM. This is to allow time for the removal of high voltage.

5.3.9 EEPROM non-volatile register (EENVR) and EEPROM array configuration register (EEACR)

EEACR $FE1F	7	6	5	4	3	2	1	0
READ:	EERA	CON2	CON1	CON0	EEBP3	EEBP2	EEBP1	EEBP0
WRITE::								
RESET:	EENVR	EENVR	EENVR	EENVR	EENVR	EENVR	EENVR	EENVR

Figure 5-2 EEPROM array control register (EEACR)

EENVR $FE1C	7	6	5	4	3	2	1	0
READ:	EERA	CON2	CON1	CON0	EEBP3	EEBP2	EEBP1	EEBP0
WRITE:								
RESET:	PV	PV	PV	PV	PV	PV	PV	PV

Figure 5-3 EEPROM non-volatile register (EENVR)

PV = Programmed Value or ’1’ in the erased state.

EERA — EEPROM redundant array

This programmable/erase/read bit in EENVR and read-only bit in EEACR configures the array in redundant mode. Reset loads EERA from EENVR to EEACR.

1 = EEPROM array is in redundant mode configuration
0 = EEPROM array is in normal mode configuration

CONx — MCU configuration bits

These read/write bits can be used to enable/disable functions within the MCU. Reset loads CONx from EENVR to EEACR.

CON2 — Unused

CON1 — Unused

CON0 — EEPROM security
1 = EEPROM security disabled
0 = EEPROM security enabled

EEBP3–EEBP0 — EEPROM block protection bits.

These read/write bits prevent blocks of EEPROM array from being programmed or erased. Reset loads EEBP[3:0] from EENVR to EEACR.
1 = EEPROM array block is protected
0 = EEPROM array block is unprotected

5.3.10 Low power modes

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

5.3.10.1 WAIT mode

The WAIT instruction does not affect the EEPROM. It is possible to program the EEPROM and put the MCU in WAIT mode. However, if the EEPROM is inactive, power can be reduced by setting the EEOFF bit before executing the WAIT instruction.

5.3.10.2 STOP mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while the high voltage is turned on (EEPGM=1).

If STOP mode is entered while program/erase is in progress, high voltage will automatically be turned off. However, the EEPGM bit will remain set. When STOP mode is terminated, if EEPGM is still set, the high voltage will automatically be turned back on. Program/erase time will need to be extended if program/erase is interrupted by entering STOP mode.

The module requires a recovery time t_{EESTOP} to stabilize after leaving STOP mode. Attempts to access the array during the recovery time will result in unpredictable behavior.

5

SECTION 6
CENTRAL PROCESSOR UNIT (CPU)

6.1 Introduction

This section describes the central processor unit (CPU8). 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.

6.2 Features

Features of the CPU include the following:

- Full upward, object-code compatibility with M68HC05 family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with X-register manipulation instructions
- 8MHz CPU internal bus frequency
- 64K byte 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 64K bytes
- Low-power STOP and WAIT Modes

6.3 CPU registers

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

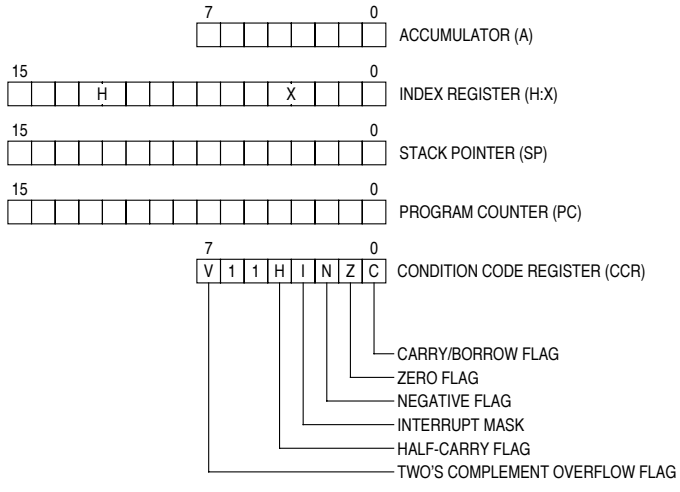


Figure 6-1 CPU registers

6.3.1 Accumulator (A)

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

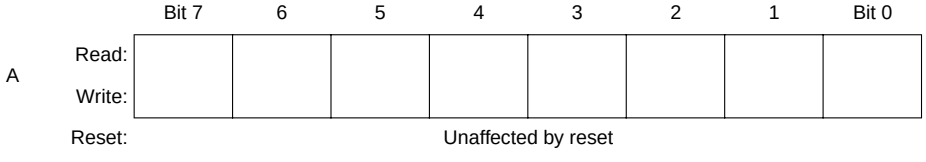


Figure 6-2 Accumulator (A)

6.3.2 Index register (H:X)

The 16-bit index register allows indexed addressing of a 64K byte 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.

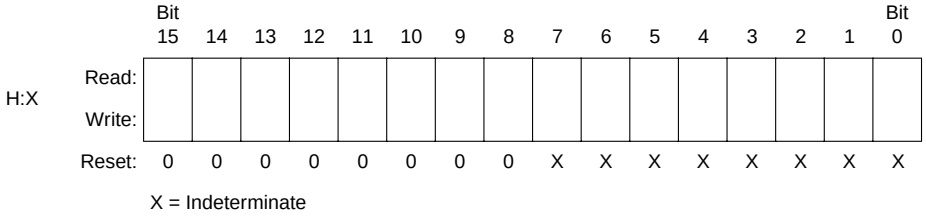


Figure 6-3 Index register (H:X)

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

6

6.3.3 Stack pointer (SP)

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 sets the least significant byte to $FF and does not affect the most significant byte. 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.

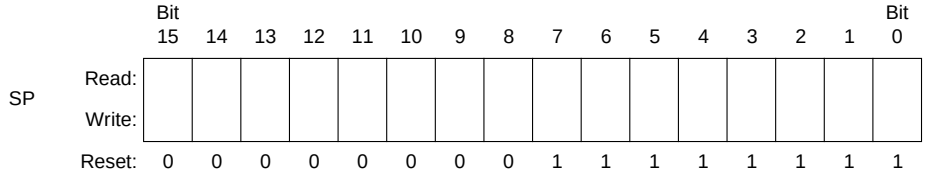


Figure 6-4 Stack pointer (SP)

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.

6.3.4 Program counter (PC)

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.

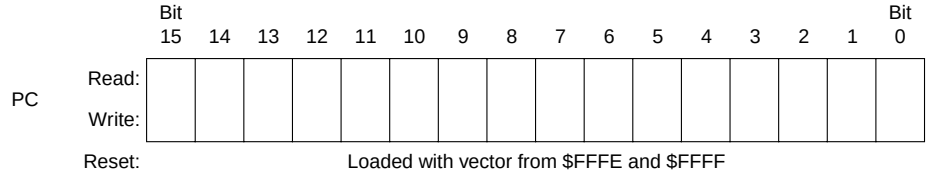


Figure 6-5 Program counter (PC)

6.3.5 Condition code register (CCR)

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 ‘1’. The following paragraphs describe the functions of the condition code register.

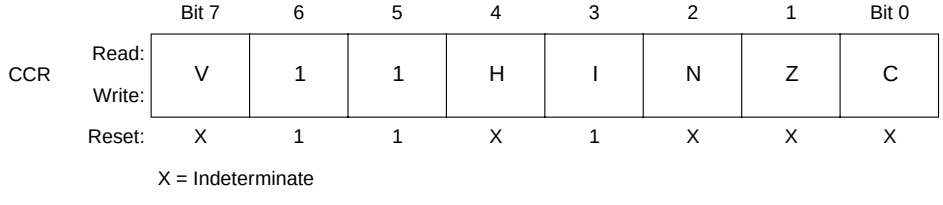


Figure 6-6 Condition code register (CCR)

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 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 mask software instruction (CLI).

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

6.4 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 detail about CPU architecture.

6.5 CPU during break interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction. See **SECTION 10 BREAK MODULE**. The program counter vectors to $FFFC–$FFFD ($FEFC–$FEFD in monitor mode).

A return from interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation if the break interrupt has been deasserted.

SECTION 7
SYSTEM INTEGRATION MODULE (SIM)

7.1 Introduction

This section describes the system integration module, which supports up to 24 external and/or internal interrupts. Together with the CPU, the SIM controls all MCU activities. A block diagram of the SIM is shown in Figure 7-1. Table 7-1 is a summary of the SIM I/O registers. The SIM is a system state controller that coordinates CPU and exception timing. 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 COP timeout
- Interrupt control:
  - Acknowledge timing
  - Arbitration control timing
  - Vector address generation
- CPU enable/disable timing
- Modular architecture expandable to 128 interrupt sources

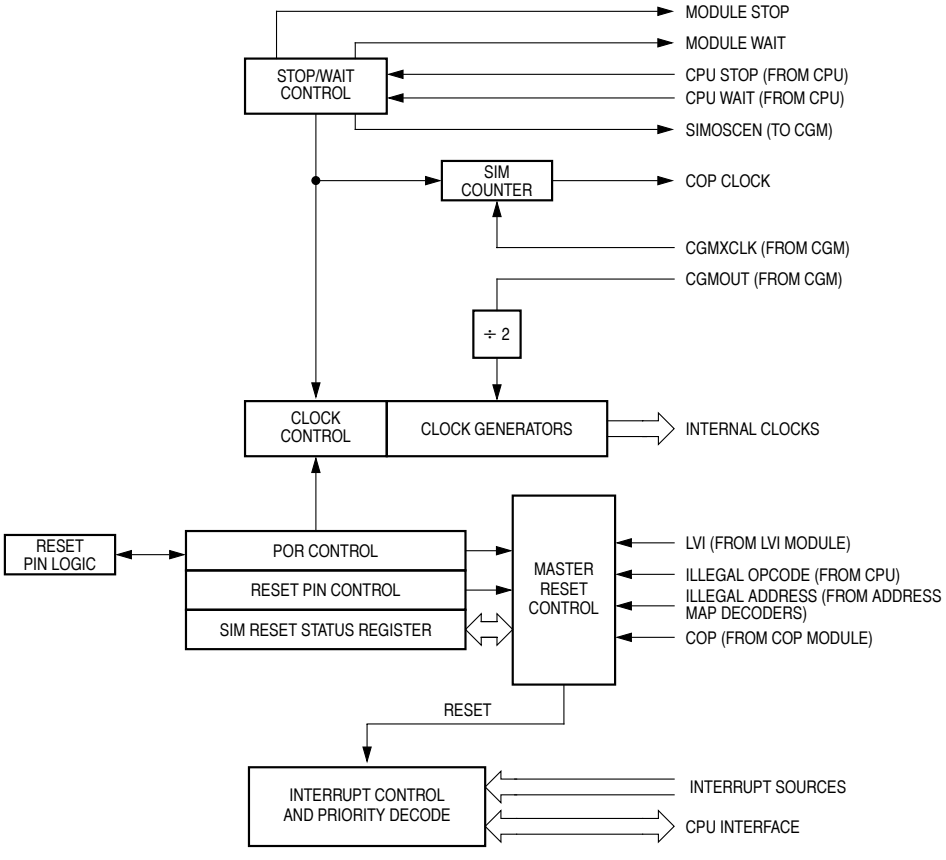


Figure 7-1 SIM block diagram

Table 7-1 SIM I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SIM Break Status Register (SBSR)	R	R	R	R	R	R	SBSW	R	$FE00
SIM Reset Status Register (SRSR)	POR	PIN	COP	ILOP	ILAD	0	LVI	0	$FE01
SIM Break Flag Control Register (SBFCR)	BCFE	0	0	0	0	0	0	0	$FE03

R = Reserved for factory test

Table 7-2 shows the internal signal names used in this section.

Table 7-2 Signal naming conventions

Signal Name	Description
CGMXCLK	Buffered version of OSC1 from clock generator module (CGM)
CGMVCLK	PLL output
CGMOUT	PLL-based or OSC1-based clock output from CGM module (Bus clock = CGMOUT divided by two)
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

7.2 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 7-2. This clock can come from either an external oscillator or from the on-chip PLL. See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.

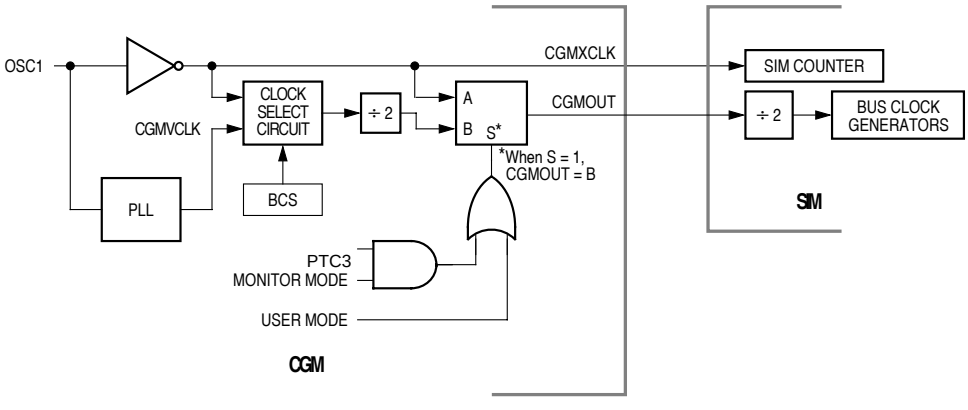


Figure 7-2 CGM clock signals

7.2.1 Bus timing

In user mode**,** the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.

7.2.2 Clock start-up from POR or LVI reset

When the power-on reset module or the low-voltage inhibit 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 been 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.

7.2.3 Clocks in STOP 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 **7.6.2 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.

7.3 Reset and system initialization

The MCU has the following reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{\text{RST}}$)
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address

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 **7.4 SIM counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See **7.7 SIM registers**.)

7.3.1 External pin reset

Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See Table 7-3 for details. Figure 7-3 shows the relative timing.

Table 7-3 PIN bit set timing

Reset type	Number of cycles required to set PIN
POR/LVI	4163 (4096 + 64 + 3)
All others	67 (64 + 3)

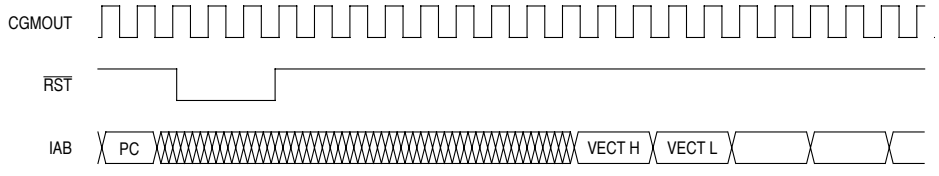


Figure 7-3 External reset timing

7.3.2 Active resets from internal sources

All internal reset sources actively pull the $\overline{RST}$ pin low for 32 CGMXCLK cycles to allow for resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles. See Figure 7-4. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. See Figure 7-5. Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles, during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ as shown in Figure 7-4.

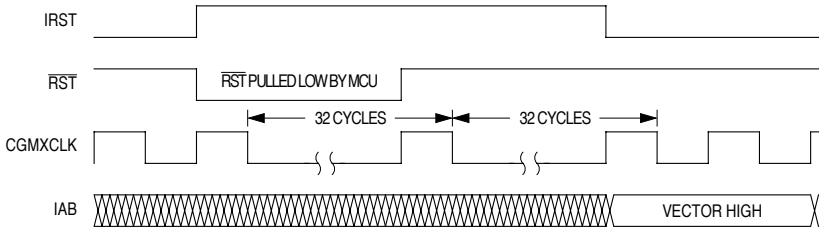


Figure 7-4 Internal reset timing

The COP reset is asynchronous to the bus clock.

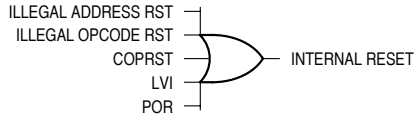


Figure 7-5 Sources of internal reset

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

7.3.2.1 Power-on reset

7

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 ($\overline{RST}$) is held low while the SIM counter counts out 4096 CGMXCLK cycles. 64 CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, the following events occur:

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

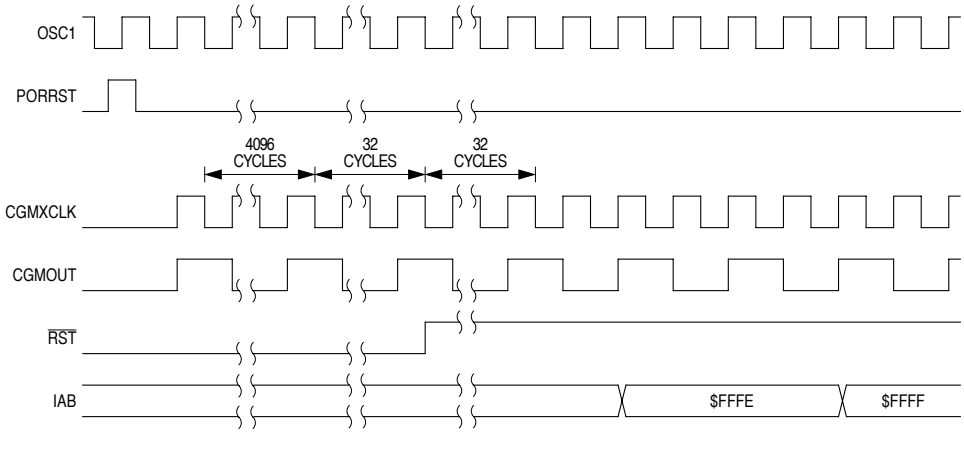


Figure 7-6 POR recovery

7.3.2.2 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 SIM reset status register (SRSR). The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

To prevent a COP module timeout, a value (any value) should be written to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 4 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} – 2^4$ 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 $\overline{RST}$ pin or the $\overline{IRQ1}/V_{PP}$ pin is held at $V_{DD} + V_{HI}$ 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 $\overline{RST}$ or the $\overline{IRQ1}/V_{PP}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, $V_{DD} + V_{HI}$ on the $\overline{RST}$ pin disables the COP module.

7.3.2.3 Illegal opcode reset

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) 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 $\overline{RST}$ pin for all internal reset sources.

7.3.2.4 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 SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

NOTE

This function is only active in Non-Expanded mode.

7.3.2.5 Low-voltage inhibit (LVI) reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the LVI_{TRIPF} voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 CGMXCLK cycles. 64 CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.4 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 (COP) module. The SIM counter overflow supplies the clock for the COP module. The SIM counter is 13 bits long and is clocked by the falling edge of CGMXCLK.

7.4.1 SIM counter during power-on reset

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

7.4.2 SIM counter during STOP mode recovery

The SIM counter is also 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 start-up times from STOP mode. External crystal applications should use the full STOP recovery time, that is, with SSREC cleared.

7.4.3 SIM counter and reset states

External reset has no effect on the SIM counter. (See **7.6.2 STOP mode** for details). The SIM counter is free-running after all reset states. (See **7.3.2 Active resets from internal sources** for counter control and internal reset recovery sequences).

7.5 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

7.5.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents onto 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 7-7 shows interrupt entry timing, and Figure 7-9 shows interrupt recovery timing.

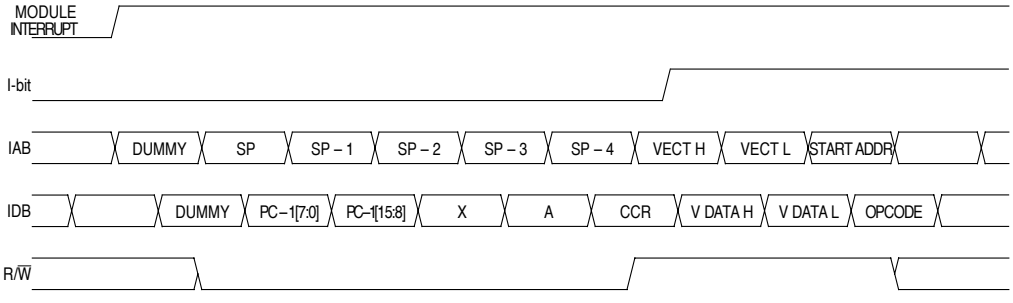


Figure 7-7 Interrupt entry

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 may take precedence, regardless of priority, until the latched interrupt is serviced (or the I-bit is cleared). See Figure 7-8.

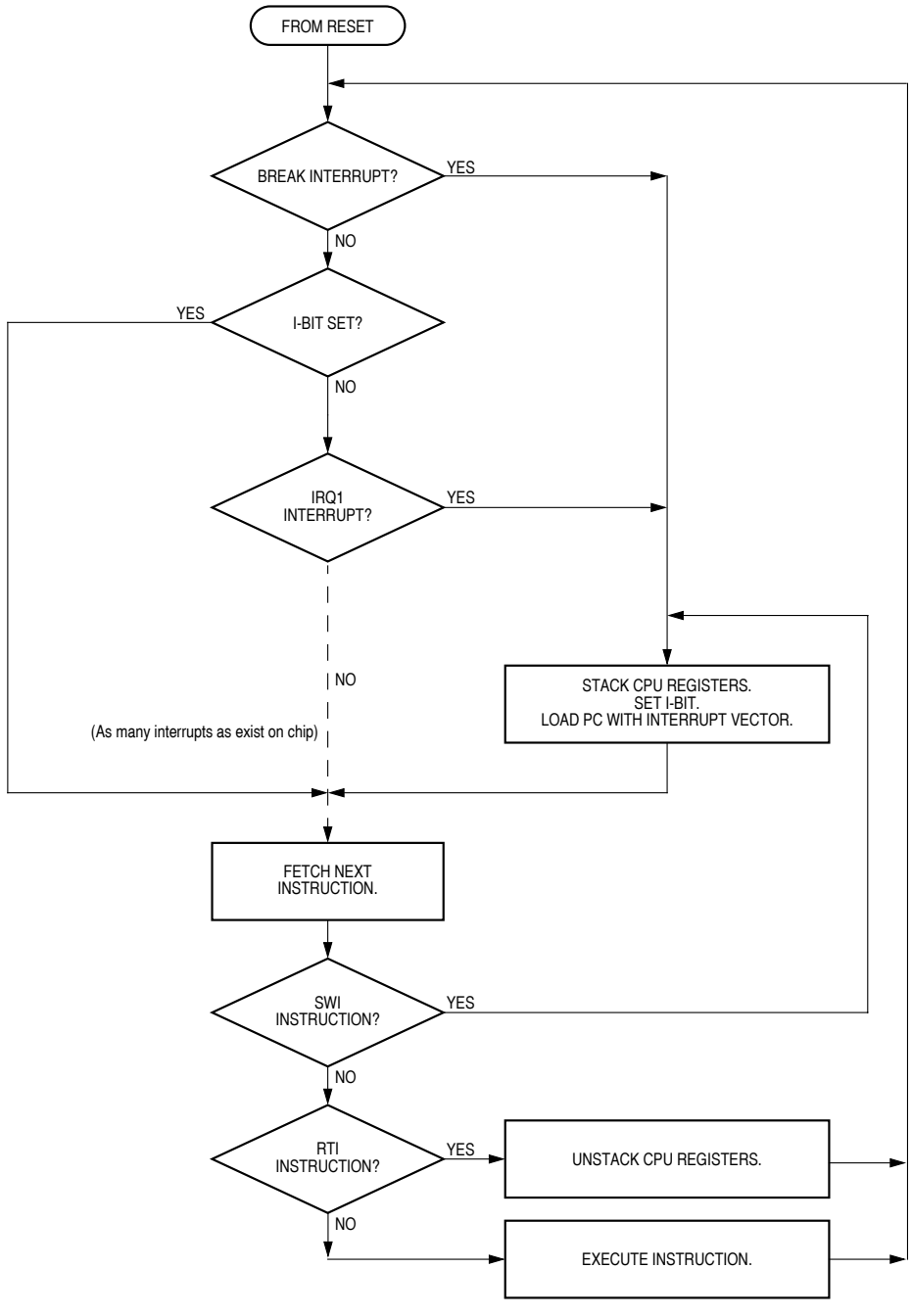


Figure 7-8 Interrupt processing

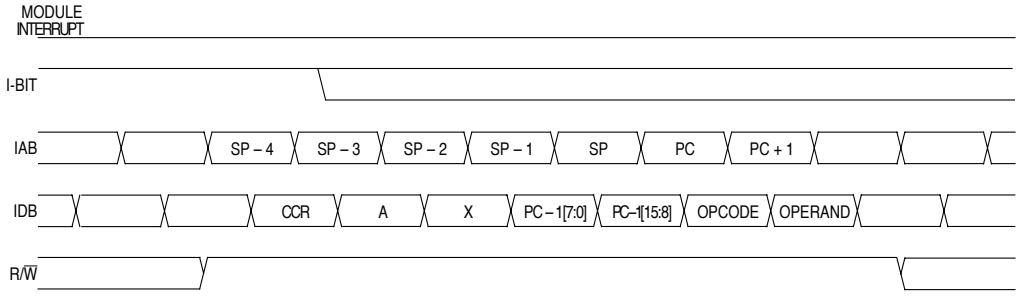


Figure 7-9 Interrupt recovery

7.5.1.1 Hardware interrupts

Processing of a hardware interrupt begins after completion of the current instruction. When the 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 7-10 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.

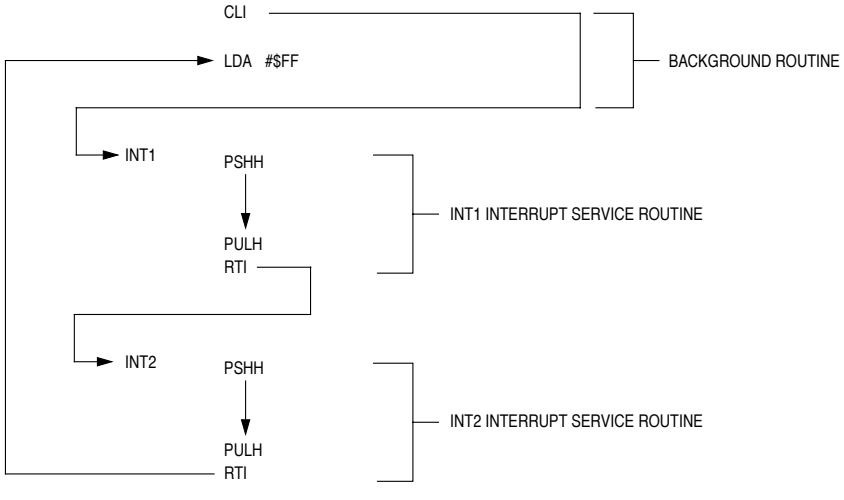


Figure 7-10 Interrupt recognition example

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.

7.5.1.2 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.

7.5.2 Reset

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

7.5.3 Break interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See **SECTION 10 BREAK MODULE**. 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.

7.5.4 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 SIM break flag control register (SBFCR).

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 two-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.

7.6 Low-power modes

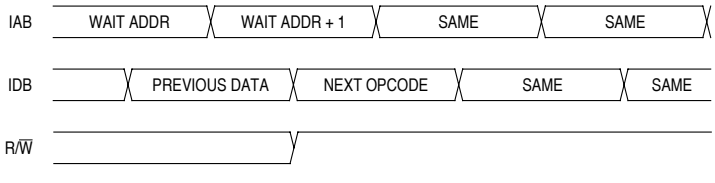
Executing the STOP/WAIT instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts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

7.6.1 WAIT mode

In WAIT mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 7-11 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.

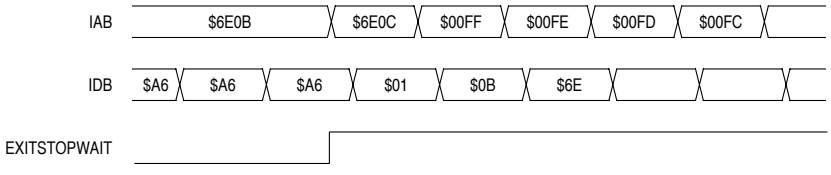
WAIT mode can also be exited by a reset or break. A break interrupt during WAIT mode sets the SIM break STOP/WAIT bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is ’0’, then the computer operating properly (COP) module is enabled and remains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



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

Figure 7-11 WAIT mode entry timing

Figure 7-12 and Figure 7-13 show the timing for WAIT recovery.



NOTE: EXITSTOPWAIT = $\overline{\text{RST}}$ pin OR CPU interrupt OR break interrupt

Figure 7-12 WAIT recovery from interrupt or break

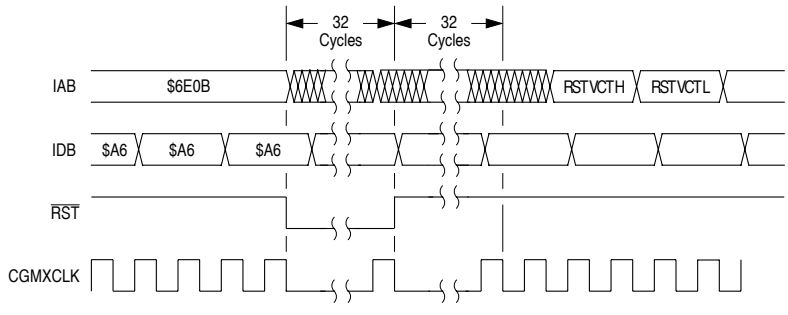


Figure 7-13 WAIT recovery from internal reset

7.6.2 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.

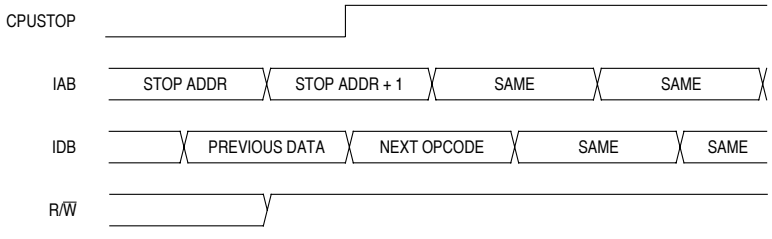
The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in STOP mode, stopping the CPU and peripherals. STOP recovery time is selectable using the SSREC bit in the mask option register (MOR). If SSREC is set, STOP recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from STOP mode.

NOTE

External crystal applications should use the full STOP recovery time by clearing the SSREC bit.

A break interrupt during STOP mode sets the SIM break STOP/WAIT bit (SBSW) in the SIM break status register (SBSR).

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 7-14 shows STOP mode entry timing.



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

Figure 7-14 STOP mode entry timing

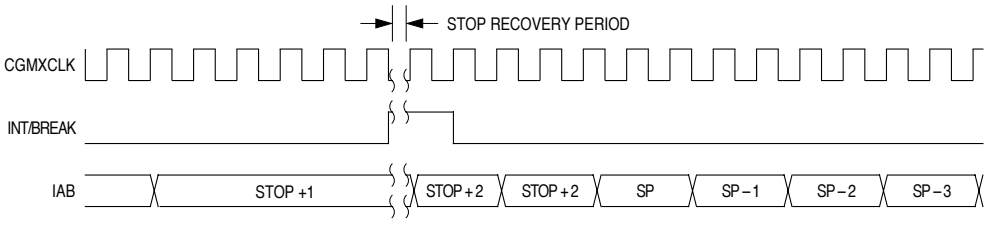


Figure 7-15 STOP mode recovery from interrupt or break

7.7 SIM registers

The SIM has three memory mapped registers. Table 7-4 shows the mapping of these registers.

Table 7-4 SIM Registers

Address	Register	Access mode
$FE00	SBSR	User
$FE01	SRSR	User
$FE03	SBFCR	User

7.7.1 SIM break status register (SBSR)

The SIM break status register contains a flag to indicate that a break caused an exit from STOP or WAIT mode.

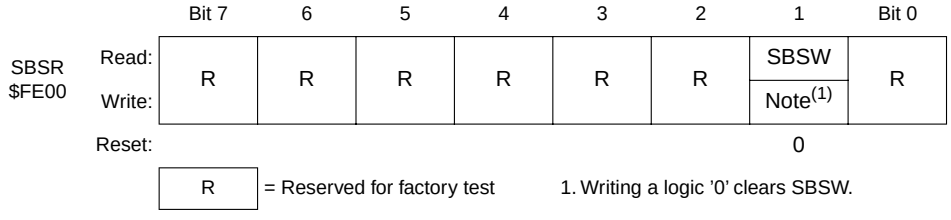


Figure 7-16 SIM break status register (SBSR)

SBSW — SIM Break STOP/WAIT

This status bit is useful in applications requiring a return to STOP or WAIT mode after exiting from a break interrupt. SBSW can be cleared by writing a logic '0' to it. Reset clears SBSW.

1 = STOP or WAIT mode was exited by break interrupt
0 = STOP or WAIT mode was 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.

```
; 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,SBSR, RETURN ; See if STOP or WAIT mode was exited by
                                    ; break.

           TST    LOBYTE,SP         ; If RETURNLO is not '0',

           BNE    DOLO              ; then just decrement low byte.

           DEC    HIBYTE,SP         ; Else deal with high byte, too.

  DOLO     DEC    LOBYTE,SP         ; Point to STOP/WAIT opcode.

  RETURN   PULH                     ; Restore H register.
           RTI
```

7.7.2 SIM reset status register (SRSR)

This register contains six flags that show the source of the last reset. The SIM reset status register can be cleared by reading it. A power-on reset sets the POR bit and clears all other bits in the register.

		Bit 7	6	5	4	3	2	1	Bit 0
SRSR $FE01	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
	Write:								
	POR:	1	0	0	0	0	0	0	0

= Unimplemented

Figure 7-17 SIM reset status register (SRSR)

POR — Power-on reset bit
- 1 = Last reset caused by POR circuit
- 0 = Read of SRSR

PIN — External reset bit
- 1 = Last reset caused by external reset pin ($\overline{RST}$)
- 0 = POR or read of SRSR

COP — Computer operating properly reset bit
- 1 = Last reset caused by COP counter
- 0 = POR or read of SRSR

ILOP — Illegal opcode reset bit
- 1 = Last reset caused by an illegal opcode
- 0 = POR or read of SRSR

ILAD — Illegal address reset bit (opcode fetches only)
- 1 = Last reset caused by an opcode fetch from an illegal address
- 0 = POR or read of SRSR

LVI — Low-voltage inhibit reset bit
- 1 = Last reset was caused by the LVI circuit
- 0 = POR or read of SRSR

7.7.3 SIM break flag control register (SBFCR)

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

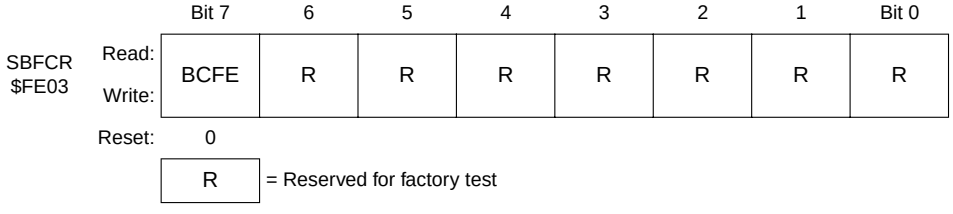


Figure 7-18 SIM break flag control register (SBFCR)

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 8
CLOCK GENERATOR MODULE (CGM)

8.1 Introduction

This section describes the clock generator module (CGM). The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates the base clock signal, CGMOUT, from which the system integration module (SIM) derives the system clocks. CGMOUT is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. The PLL is a frequency generator designed for use with 1MHz to 16MHz crystals or ceramic resonators. The PLL can generate an 8MHz bus frequency without using a 32MHz crystal.

8.2 Features

Features of the CGM include the following:

- Phase-locked loop with output frequency in integer multiples of the crystal reference
- Programmable hardware voltage-controlled oscillator (VCO) for low-jitter operation
- Automatic bandwidth control mode for low-jitter operation
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition

8.3 Functional description

The CGM consists of three major submodules:

- Crystal oscillator circuit which generates the constant crystal frequency clock, CGMXCLK.
- Phase-locked loop (PLL) which generates the programmable VCO frequency clock CGMVCLK.
- Base clock selector circuit; this software-controlled circuit selects either CGMXCLK divided by two or the VCO clock CGMVCLK divided by two, as the base clock CGMOUT. The SIM derives the system clocks from CGMOUT.

Figure 8-1 shows the structure of the CGM.

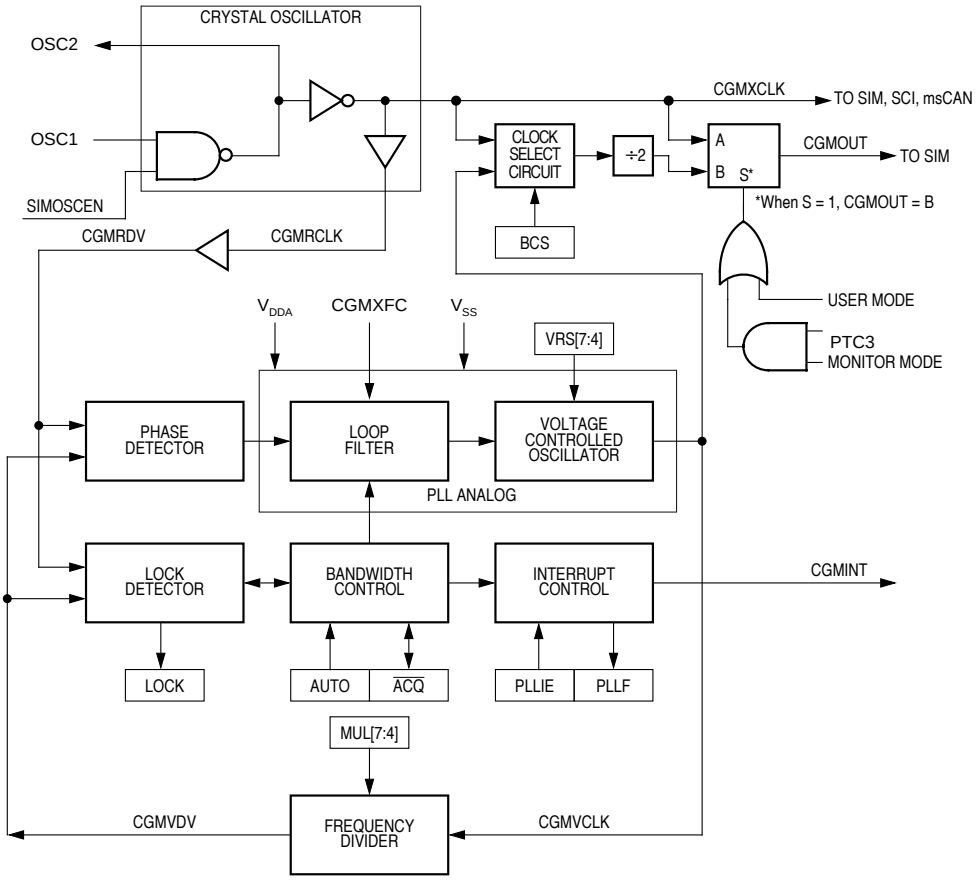


Figure 8-1 CGM Block Diagram

Table 8-1 CGM I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0
PLL Control Register (PCTL)	PLLIE	PLLF	PLLON	BCS	1	1	1	1
PLL Bandwidth Control Register (PBWC)	AUTO	LOCK	$\overline{ACQ}$	XLD	0	0	0	0
PLL Programming Register (PPG)	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4

8.3.1 Crystal oscillator circuit

The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the input to the amplifier and the OSC2 pin is the output. The SIMOSCEN signal from the system integration module (SIM) enables the crystal oscillator circuit.

The CGMXCLK signal is the output of the crystal oscillator circuit and runs at a rate equal to the crystal frequency. CGMXCLK is then buffered to produce CGMRCLK, the PLL reference clock.

CGMXCLK can be used by other modules which require precise timing for operation. The duty cycle of CGMXCLK is not guaranteed to be 50% and depends on external factors, including the crystal and related external components.

An externally generated clock can also feed the OSC1 pin of the crystal oscillator circuit. For this configuration, the external clock should be connected to the OSC1 pin and the OSC2 pin allowed to float.

8.3.2 Phase-locked loop (PLL) circuit

The PLL is a frequency generator that can operate in either acquisition mode or tracking mode, depending on the accuracy of the output frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

8.3.2.1 PLL circuits

The PLL consists of the following circuits:

- Voltage-controlled oscillator (VCO)
- Modulo VCO frequency divider
- Phase detector
- Loop filter
- Lock detector

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGMXFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency, f_{VRS}. Modulating the voltage on the CGMXFC pin changes the frequency within this range. By design, f_{VRS} is equal to the nominal center-of-range frequency, f_{NOM}, (4.9152 MHz) times a linear factor L, or (L)f_{NOM}.

CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency f_{RCLK}, and is fed to the PLL through a buffer. The buffer output is the final reference clock, CGMRDV, running at a frequency $f_{RDV} = f_{RCLK}$.

The VCO's output clock, CGMVCLK, running at a frequency f_{VCLK}, is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor N. The divider's output is the VCO feedback clock, CGMVDV, running at a frequency $f_{VDV} = f_{VCLK}/N$. (See **8.3.2.4 Programming the PLL** for more information).

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in **8.3.2.2 Acquisition and tracking modes**. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

The lock detector compares the frequencies of the VCO feedback clock, CGMVDV, and the final reference clock, CGMRDV. Therefore, the speed of the lock detector is directly proportional to the final reference frequency f_{RDV}. The circuit determines the mode of the PLL and the lock condition based on this comparison.

8.3.2.2 Acquisition and tracking modes

The PLL filter is manually or automatically configurable into one of two operating modes:

- Acquisition mode — in acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the resulting VCO frequency is much different from the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. See **8.5.2 PLL Bandwidth control register (PBWC)**.
- Tracking mode — in tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. See **8.3.3 Base clock selector circuit**. The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

8.3.2.3 Manual and automatic PLL bandwidth modes

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode is used also to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. See **8.5.2 PLL Bandwidth control register (PBWC)**. If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. See **8.3.3 Base clock selector circuit**. If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See **8.6 Interrupts** for information and precautions on using interrupts). The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{ACQ}$ bit (see **8.5.2 PLL Bandwidth control register (PBWC)**) is a read-only indicator of the mode of the filter. (See **8.3.2.2 Acquisition and tracking modes**)
- The $\overline{ACQ}$ bit is set when the VCO frequency is within a certain tolerance Δ_{TRK} and is cleared when the VCO frequency is outwith a certain tolerance Δ_{UNT}. (See **8.9 Acquisition/lock time specifications**)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance Δ_{LOCK} and is cleared when the VCO frequency is outwith a certain tolerance Δ_{UNL}. (See **8.9 Acquisition/lock time specifications**)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See **8.5.1 PLL control register (PCTL)**)

The PLL also may operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below f_{BUSMAX} and require fast start-up. The following conditions apply when in manual mode:

- $\overline{ACQ}$ is a writeable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{ACQ}$ bit must be clear.
- Before entering tracking mode ($\overline{ACQ}$ = 1), software must wait a given time, t_{ACQ} (see **8.9 Acquisition/lock time specifications**), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

8

8.3.2.4 Programming the PLL

The following procedure shows how to program the PLL.

NOTE

The round function in the following equations means that the real number should be rounded to the nearest integer number.

---

1. Choose the desired bus frequency, f_{BUSDES}.
2. Calculate the desired VCO frequency (four times the desired bus frequency).

$$f_{VCLKDES} = 4 \times f_{BUSDES}$$

3. Choose a practical PLL reference frequency, f_{RCLK}.
4. Select a VCO frequency multiplier, N.

$$N = \text{round}\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right)$$

5. Calculate and verify the adequacy of the VCO and bus frequencies f_{VCLK} and f_{BUS}.

$$f_{VCLK} = N \times f_{RCLK}$$

$$f_{BUS} = (f_{VCLK})/4$$

6. Select a VCO linear range multiplier, L.

$$L = \text{round}\left(\frac{f_{VCLK}}{f_{NOM}}\right)$$

where f_{NOM} = 4.9152 MHz

7. Calculate and verify the adequacy of the VCO programmed center-of-range frequency f_{VRS}.

$$f_{VRS} = (L)f_{NOM}$$

8. Verify the choice of N and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. For proper operation, f_{VCLK} must be within the application's tolerance of $f_{VCLKDES}$, and f_{VRS} must be as close as possible to f_{VCLK}.

NOTE

Exceeding the recommended maximum bus frequency or VCO frequency can cause the MCU to "crash".

---

9. Program the PLL registers accordingly:
   a. In the upper 4 bits of the PLL programming register (PPG), program the binary equivalent of N.
   b. In the lower 4 bits of the PLL programming register (PPG), program the binary equivalent of L.

8.3.2.5 Special programming exceptions

The programming method described in **8.3.2.4 Programming the PLL** does not account for two possible exceptions — a value of zero for N or L is meaningless when used in the equations given. To account for these exceptions:

- A zero value for N is interpreted exactly the same as a value of one.
- A zero value for L disables the PLL and prevents its selection as the source for the base clock. (See **8.3.3 Base clock selector circuit**)

8.3.3 Base clock selector circuit

This circuit is used to select either the crystal clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The output of the transition control circuit is then divided by two to correct the duty cycle. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a zero. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

8.3.4 CGM external connections

In its typical configuration, the CGM requires seven external components. Five of these are for the crystal oscillator and two are for the PLL.

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in Figure 8-2. This figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

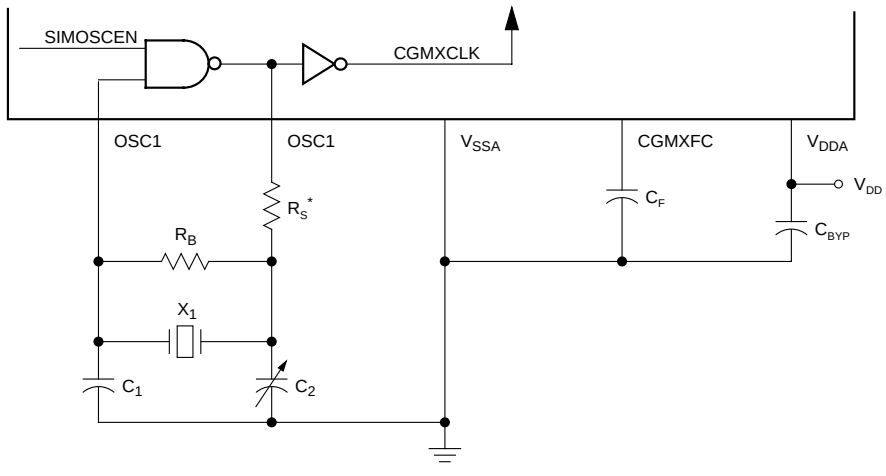
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

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.

Figure 8-2 also shows the external components for the PLL:

- Bypass capacitor, C_{BYP}
- Filter capacitor, C_F

Care should be taken with routing in order to minimize signal cross talk and noise. (See **8.9 Acquisition/lock time specifications** for routing information and more information on the filter capacitor's value and its effects on PLL performance).



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

Figure 8-2 CGM external connections

8.4 I/O Signals

The following paragraphs describe the CGM I/O signals.

8.4.1 Crystal amplifier input pin (OSC1)

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

8.4.2 Crystal amplifier output pin (OSC2)

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

8.4.3 External filter capacitor pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. A small external capacitor is connected to this pin.

NOTE

To prevent noise problems, C_F should be placed as close to the CGMXFC pin as possible, with minimum routing distances and no routing of other signals across the C_F connection.

8.4.4 PLL analog power pin (V_{DDA})

V_{DDA} is a power pin used by the analog portions of the PLL. The pin should be connected to the same voltage potential as the V_{DD} pin.

NOTE

Route V_{DDA} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.

8.4.5 Oscillator enable signal (SIMOSCEN)

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

8.4.6 Crystal output frequency signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal (f_{XCLK}) and is generated directly from the crystal oscillator circuit. Figure 8-2 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 start-up.

8.4.7 CGM base clock output (CGMOUT)

CGMOUT is the clock output of the CGM. This signal goes to the SIM, which generates the MCU clocks. CGMOUT is a 50% duty cycle clock running at twice the bus frequency. CGMOUT is software programmable to be either the oscillator output (CGMXCLK) divided by two or the VCO clock (CGMVCLK) divided by two.

8.4.8 CGM CPU interrupt (CGMINT)

CGMINT is the interrupt signal generated by the PLL lock detector.

8.5 CGM registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL). (See **8.5.1 PLL control register (PCTL)**)
- PLL bandwidth control register (PBWC). (See **8.5.2 PLL Bandwidth control register (PBWC)**)
- PLL programming register (PPG). (See **8.5.3 PLL Programming register (PPG)**)

Figure 8-3 is a summary of the CGM registers.

		Bit 7	6	5	4	3	2	1	Bit 0
PCTL $001C	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
	Write:								
PBWC $001D	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
	Write:								
PPG $001E	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
	Write:								

= Unimplemented

NOTES:

1. When AUTO = 0, PLLIE is forced to logic zero and is read-only.
2. When AUTO = 0, PLLF and LOCK read as logic zero.
3. When AUTO = 1, $\overline{\text{ACQ}}$ is read-only.
4. When PLLON = 0 or VRS[7:4] = $0, BCS is forced to logic zero and is read-only.
5. When PLLON = 1, the PLL programming register is read-only.
6. When BCS = 1, PLLON is forced set and is read-only.

Figure 8-3 CGM I/O Register Summary

8.5.1 PLL control register (PCTL)

The PLL control register contains the interrupt enable and flag bits, the on/off switch, and the base clock selector bit.

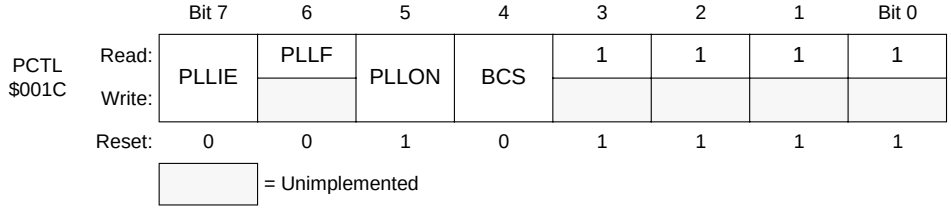


Figure 8-4 PLL control register (PCTL)

8

PLLIE — PLL interrupt enable bit

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as ‘0’. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL interrupt flag bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit is set also. PLLF always reads as ‘0’ when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. The PLLF bit should be cleared by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition
0 = No change in lock condition

NOTE

The PLLF bit should not be inadvertently cleared. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.

PLLON — PLL on bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). See **8.3.3 Base clock selector circuit**. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base clock select bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. See **8.3.3 Base clock selector circuit**. Reset and the STOP instruction clear the BCS bit.

1 = CGMOUT driven by CGMVCLK/2
0 = CGMOUT driven by CGMXCLK/2

NOTE

PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. See **8.3.3 Base clock selector circuit**.

PCTL[3:0] — Unimplemented bits

These bits provide no function and always read as ‘1’.

8.5.2 PLL Bandwidth control register (PBWC)

The PLL bandwidth control register does the following:

- Selects automatic or manual (software-controlled) bandwidth control mode
- Indicates when the PLL is locked
- In automatic bandwidth control mode, indicates when the PLL is in acquisition or tracking mode
- In manual operation, forces the PLL into acquisition or tracking mode

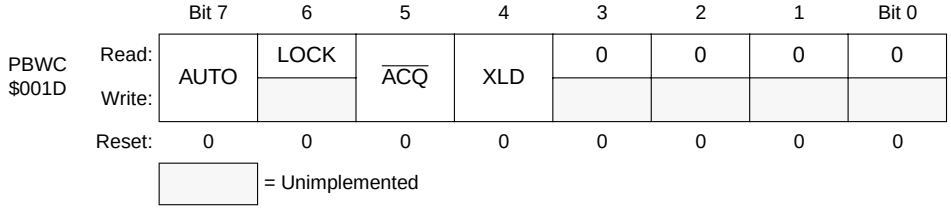


Figure 8-5 PLL bandwidth control register (PBWC)

AUTO — Automatic bandwidth control bit

This read/write bit selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), the $\overline{ACQ}$ bit should be cleared before turning the PLL on. Reset clears the AUTO bit.

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock indicator bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as '0' and has no meaning. Reset clears the LOCK bit.

1 = VCO frequency correct or locked
0 = VCO frequency incorrect or unlocked

$\overline{ACQ}$ — Acquisition mode bit

When the AUTO bit is set, $\overline{ACQ}$ is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, $\overline{ACQ}$ is a read/write bit that controls whether the PLL is in acquisition or tracking mode.

In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit, enabling acquisition mode.

1 = Tracking mode
0 = Acquisition mode

XLD — Crystal loss detect bit

When the VCO output, CGMVCLK, is driving CGMOUT, this read/write bit indicates whether the crystal reference frequency is active or not. To check the status of the crystal reference, the following procedure should be followed:

1. Write a '1' to XLD.
2. Wait 4 $\times$ N cycles. (N is the VCO frequency multiplier.)
3. Read XLD.

1 = Crystal reference is not active
0 = Crystal reference is active

The crystal loss detect function works only when the BCS bit is set, selecting CGMVCLK to drive CGMOUT. When BCS is clear, XLD always reads as '0'.

PBWC[3:0] — Reserved for test

These bits enable test functions not available in user mode. To ensure software portability from development systems to user applications, software should write zeros to PBWC[3:0] whenever writing to PBWC.

8.5.3 PLL Programming register (PPG)

The PLL programming register contains the programming information for the modulo feedback divider and the programming information for the hardware configuration of the VCO.

		Bit 7	6	5	4	3	2	1	Bit 0
PPG $001E	Read: Write:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
	Reset:	0	1	1	0	0	1	1	0

Figure 8-6 PLL Programming register (PPG)

MUL[7:4] — Multiplier select bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See **8.3.2.1 PLL circuits** and **8.3.2.4 Programming the PLL**). A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6.

Table 8-2 VCO frequency multiplier (N) selection

MUL7:MUL6:MUL5:MUL4	VCO Frequency Multiplier (N)
0000	1
0001	1
0010	2
0011	3
↓	↓
1101	13
1110	14
1111	15

8

NOTE

The multiplier select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1).

VRS[7:4] — VCO range select bits

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency f_{VRS}. (See **8.3.2.1 PLL circuits**, **8.3.2.4 Programming the PLL**, and **8.5.1 PLL control register (PCTL)**). VRS[7:4] cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See **8.3.2.5 Special programming exceptions**). A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See **8.3.3 Base clock selector circuit** and **8.3.2.5 Special programming exceptions** for more information). Reset initializes the bits to $6 to give a default range multiply value of 6.

NOTE

The VCO range select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1) and prevents selection of the VCO clock as the source of the base clock (BCS = 1) if the VCO range select bits are all clear.

The VCO range select bits must be programmed correctly. Incorrect programming may result in failure of the PLL to achieve lock.

8.6 Interrupts

When the AUTO bit is set in the PLL bandwidth control register (PBWC), the PLL can generate a CPU interrupt request every time the LOCK bit changes state. The PLLIE bit in the PLL control register (PCTL) enables CPU interrupts from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether interrupts are enabled or not. When the AUTO bit is clear, CPU interrupts from the PLL are disabled and PLLF reads as ‘0’.

Software should read the LOCK bit after a PLL interrupt request to see if the request was due to an entry into lock or an exit from lock. When the PLL enters lock, the VCO clock CGMVCLK, divided by two can be selected as the CGMOUT source by setting BCS in the PCTL. When the PLL exits lock, the VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency-sensitive, interrupts should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

NOTE

Software can select CGMVCLK/2 as the CGMOUT source even if the PLL is not locked (LOCK = 0). Therefore, software should make sure the PLL is locked before setting the BCS bit.

8.7 Special modes

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

8.7.1 WAIT mode

The WAIT instruction does not affect the CGM. Before entering WAIT mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from WAIT mode also can deselect the PLL output without turning off the PLL.

8.7.2 STOP mode

When the STOP instruction executes, the SIM drives the SIMOSCEN signal low, disabling the CGM and holding low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

If the STOP instruction is executed with the VCO clock, CGMVCLK, divided by two driving CGMOUT, the PLL automatically clears the BCS bit in the PLL control register (PCTL), thereby selecting the crystal clock, CGMXCLK, divided by two as the source of CGMOUT. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

8.8 CGM during break interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **SECTION 7 SYSTEM INTEGRATION MODULE (SIM)**.

To allow software to clear status bits during a break interrupt, a ‘1’ should be written 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 the PLLF bit during the break state, write a ‘0’ to the BCFE bit. With BCFE at ‘0’ (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

8.9 Acquisition/lock time specifications

The acquisition and lock times of the PLL are, in many applications, the most critical PLL design parameters. Proper design and use of the PLL ensures the highest stability and lowest acquisition/lock times.

8.9.1 Acquisition/lock time definitions

Typical control systems refer to the acquisition time or lock time as the reaction time of the system, within specified tolerances, to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percentage of the step input or when the output settles to the desired value plus or minus a percentage of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input. For example, consider a system with a 5% acquisition time tolerance. If a command instructs the system to change from 0Hz to 1MHz, the acquisition time is the time taken for the frequency to reach 1MHz ± 50kHz. 50kHz = 5% of the 1MHz step input. If the system is operating at 1MHz and suffers a –100kHz noise hit, the acquisition time is the time taken to return from 900kHz to 1MHz ± 5kHz. 5kHz = 5% of the 100kHz step input.

Other systems refer to acquisition and lock times as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the acquisition or lock time varies according to the original error in the output. Minor errors may not even be registered. Typical PLL applications prefer to use this definition because the system requires the output frequency to be within a certain tolerance of the desired frequency regardless of the size of the initial error.

The discrepancy in these definitions makes it difficult to specify an acquisition or lock time for a typical PLL. Therefore, the definitions for acquisition and lock times for this module are as follows:

- Acquisition time, t_{ACQ}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance Δ_{TRK}. Acquisition time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode (see **8.3.2.3 Manual and automatic PLL bandwidth modes**), acquisition time expires when the $\overline{ACQ}$ bit becomes set in the PLL bandwidth control register (PBWC).
- Lock time, t_{LOCK}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance Δ_{LOCK}. Lock time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). See **8.3.2.3 Manual and automatic PLL bandwidth modes**.

Obviously, the acquisition and lock times can vary according to how large the frequency error is and may be shorter or longer in many cases.

8.9.2 Parametric influences on reaction time

Acquisition and lock times are designed to be as short as possible while still providing the highest possible stability. These reaction times are not constant, however. Many factors directly and indirectly affect the acquisition time.

The most critical parameter which affects the reaction times of the PLL is the reference frequency, f_{RDV}. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is also under user control via the choice of crystal frequency f_{XCLK}.

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. See **8.9.3 Choosing a filter capacitor**.

Also important is the operating voltage potential applied to V_{DDA}. The power supply potential alters the characteristics of the PLL. A fixed value is best. Variable supplies, such as batteries, are acceptable if they vary within a known range at very slow speeds. Noise on the power supply is not acceptable, because it causes small frequency errors which continually change the acquisition time of the PLL.

Temperature and processing also can affect acquisition time because the electrical characteristics of the PLL change. The part operates as specified as long as these influences stay within the specified limits. External factors, however, can cause drastic changes in the operation of the PLL. These factors include noise injected into the PLL through the filter capacitor, filter capacitor leakage, stray impedances on the circuit board, and even humidity or circuit board contamination.

8.9.3 Choosing a filter capacitor

As described in **8.9.2 Parametric influences on reaction time**, the external filter capacitor C_F is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to the following equation:

$$C_F = C_{FACT}\left(\frac{V_{DDA}}{f_{RDV}}\right)$$

For the value of V_{DDA}, the voltage potential at which the MCU is operating should be used. If the power supply is variable, choose a value near the middle of the range of possible supply values.

This equation does not always yield a commonly available capacitor size, so round to the nearest available size. If the value is between two different sizes, choose the higher value for better stability. Choosing the lower size may seem attractive for acquisition time improvement, but the PLL may become unstable. Also, always choose a capacitor with a tight tolerance (±20% or better) and low dissipation.

8.9.4 Reaction time calculation

The actual acquisition and lock times can be calculated using the equations below. These equations yield nominal values under the following conditions:

- Correct selection of filter capacitor, C_F, (see **8.9.3 Choosing a filter capacitor**)
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{ACQ} is the K factor when the PLL is configured in acquisition mode, and K_{TRK} is the K factor when the PLL is configured in tracking mode. See **8.3.2.2 Acquisition and tracking modes**.

$$t_{ACQ} = \left(\frac{V_{DDA}}{f_{RDV}}\right)\left(\frac{8}{K_{ACQ}}\right)$$

$$t_{AL} = \left(\frac{V_{DDA}}{f_{RDV}}\right)\left(\frac{4}{K_{TRK}}\right)$$

$$t_{LOCK} = t_{ACQ} + t_{AL}$$

Note the inverse proportionality between the lock time and the reference frequency.

In automatic bandwidth control mode the acquisition and lock times are quantized into units based on the reference frequency. See **8.3.2.3 Manual and automatic PLL bandwidth modes**. A certain number of clock cycles, n_{ACQ}, is required to ascertain whether the PLL is within the tracking mode entry tolerance Δ_{TRK}, before exiting acquisition mode. Also, a certain number of clock cycles, n_{TRK}, is required to ascertain whether the PLL is within the lock mode entry tolerance Δ_{LOCK}. Therefore, the acquisition time t_{ACQ}, is an integer multiple of n_{ACQ}/f_{RDV}, and the acquisition to lock time t_{AL}, is an integer multiple of n_{TRK}/f_{RDV}. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than t_{LOCK} as calculated above.

In manual mode, it is usually necessary to wait considerably longer than t_{LOCK} before selecting the PLL clock (see **8.3.3 Base clock selector circuit**), because the factors described in **8.9.2 Parametric influences on reaction time** may slow the lock time considerably.

8

.

Table 8-3 CGM component specifications

Characteristic	Symbol	Min	Typ.	Max	Notes
Crystal load capacitance	C_L	–	–	–	Consult crystal mfg. data
Crystal fixed capacitance	C_f	–	2 * C_L	–	Consult crystal mfg. data
Crystal tuning capacitance	C_2		2 * C_L	–	Consult crystal mfg. data
Feedback bias resistor	R_B	–	22MΩ	–	
Series resistor	R_S	0	330kΩ	1MΩ	Not required
Filter capacitor	C_F	–	C_{FACT} * (V_{DDA} / f_{XCLK})	–	
Filter capacitor multiply factor	C_{FACT}	–	0.0154	–	F/sV
Bypass capacitor	C_{BYP}	–	0.1	µF	C_{BYP} must provide low AC impedance from f = f_{XCLK}/100 to 100 *f_{XCLK} , so series resistance must be considered

SECTION 9
MASK OPTIONS

9.1 Introduction

This section describes the mask options and the two mask option registers. The mask options are hardwired connections specified at the same time as the ROM code, which allow the user to customize the MCU. The options control the enable or disable of the following functions:

- Resets caused by the LVI module
- Power to the LVI module
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- ROM security(1)
- STOP instruction
- Computer operating properly (COP) module enable
- EEPROM security

9.2 Functional description

		Bit 7	6	5	4	3	2	1	Bit 0
MORA $001F	Read:		SEC	LVIRSTD	LVIPWRD	SSREC	COPRS	STOP	COPD
	Write:								
	Reset:	Unaffected by reset							

= Unimplemented

Figure 9-1 Mask option register A (MORA)

SEC — ROM security bit

SEC enables the ROM security feature. Setting the SEC bit prevents dumping of the ROM contents.

1 = ROM security enabled
0 = ROM security disabled

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

LVIRSTD — LVI reset enable bit

LVIRSTD disables the reset signal from the LVI module. See **SECTION 13 LOW-VOLTAGE INHIBIT MODULE (LVI)**.

1 = LVI module resets disabled
0 = LVI module resets enabled

LVIPWRD — LVI power enable bit

LVIPWRD disables the LVI module. See **SECTION 13 LOW-VOLTAGE INHIBIT MODULE (LVI)**.

1 = LVI module power disabled
0 = LVI module power enabled

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

If using an external crystal oscillator, the SSREC bit should not be set.

COPRS — COp rate select

COPRS selects the COP timeout period.

1 = COP timeout period is $2^{18} - 2^4$ CGMXCLK cycles.
0 = COP timeout period is $2^{13} - 2^4$ CGMXCLK cycles

STOP — STOP 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 12 COMPUTER OPERATING PROPERLY MODULE**.

1 = COP module disabled
0 = COP module enabled

		Bit 7	6	5	4	3	2	1	Bit 0
MORB $003F	Read:			EESEC					
	Write:								
	Reset:	Unaffected by reset							

= Unimplemented

Figure 9-2 Mask option register B (MORB)

EESEC — EEPROM security enable bit.

EESEC enables the EEPROM security function. Setting EESEC prevents program/erase access to locations $8F0 - $8FF of the EEPROM array and to the EEACR/EENVR configuration registers. See **5.3.7 HC08AZ32 EEPROM security**.

1 = EEPROM security function enabled
0 = EEPROM security function disabled

9

SECTION 10
BREAK MODULE

10.1 Introduction

This section describes the break module. The break module can generate a break interrupt which stops normal program flow at a defined address in order to begin execution of a background program.

10.2 Features

Features of the break module include the following:

- Accessible I/O registers during the break interrupt
- CPU-generated and DMA-generated break Interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

10.3 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).

The following 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.
- A DMA-generated address matches the contents of the break address registers during a DMA transfer.
- Software writes a ‘1’ to the BRKA bit in the break status and control register (BRKSCR).

NOTE

DMA section and associated functions are only valid if the MCU has a DMA module.

When a CPU- or DMA-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 10-1 shows the structure of the break module.

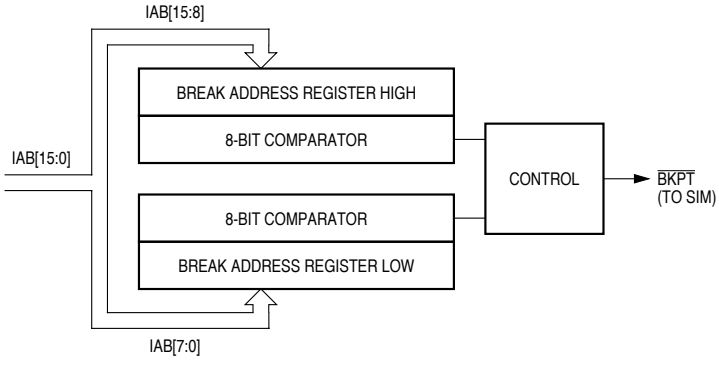


Figure 10-1 Break module block diagram

Table 10-1 Break I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Break Address Register High (BRKH)	Bit 15	14	13	12	11	10	9	Bit 8	$FE0C
Break Address Register Low (BRKL)	Bit 7	6	5	4	3	2	1	Bit 0	$FE0D
Break Status/Control Register (BRKSCR)	BRKE	BRKA							$FE0E

= Unimplemented

10.3.0.1 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **7.7.3 SIM break flag control register (SBFCR)** and the **Break Interrupts** subsection for each module.

10.3.1 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.

10.3.2 DMA during break interrupts

During a break interrupt, the DMA is inactive.

If a DMA-generated address matches the contents of the break address registers, a break interrupt begins at the end of the current CPU instruction.

If a break interrupt is asserted during the current address cycle and the DMA is active, the DMA releases the internal address and data buses at the next address boundary to preserve the current MCU state. During the break interrupt, the DMA continues to arbitrate DMA channel priorities. After the break interrupt, the DMA becomes active again and resumes transferring data according to its highest priority service request.

10.3.3 TIM and PIT during break interrupts

A break interrupt stops TimerA, TimerB, and the PIT counters.

10.3.4 COP during break interrupts

The COP is disabled during a break interrupt when $V_{DD} + V_{HI}$ is present on the $\overline{RST}$ pin.

10.4 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)

10.4.1 Break status and control register (BRKSCR)

The break status and control register contains break module enable and status bits.

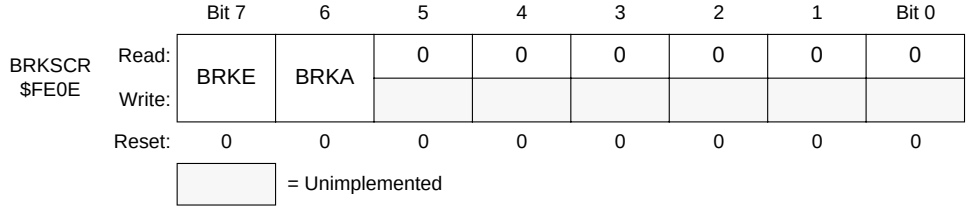


Figure 10-2 Break status and control register (BRKSCR)

BRKE — Break enable bit

This read/write bit enables breaks on break address register matches. BRKE is cleared by writing a ‘0’ to bit 7. Reset clears the BRKE bit.

1 = Breaks enabled on 16-bit address match
0 = Breaks disabled on 16-bit address match

BRKA — Break active bit

This read/write status and control bit is set when a break address match occurs. Writing a ‘1’ to BRKA generates a break interrupt. BRKA is cleared by writing a ‘0’ to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

10.4.2 Break address registers (BRKH and BRKL)

The break address registers contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

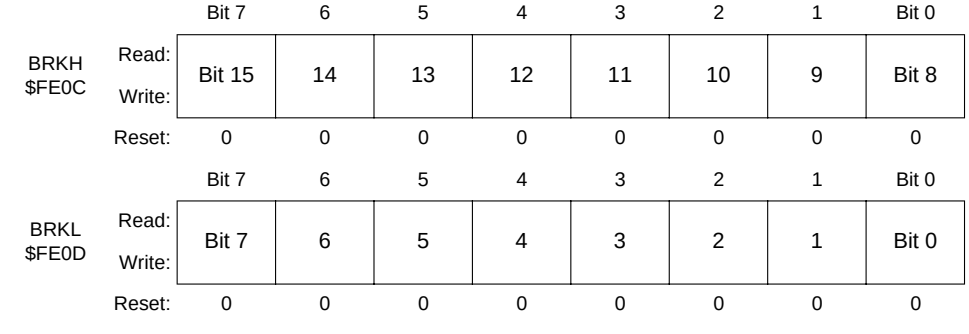


Figure 10-3 Break address registers (BRKH and BRKL)

10.5 Low-power modes

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

10.5.1 WAIT mode

If enabled, the break module and the DMA module are active in WAIT mode. The SIM break STOP/WAIT bit (SBSW) in the SIM break status register (see **7.7 SIM registers**) becomes set if a DMA-generated address matches the break address registers in WAIT mode. The DMA can also use the break status and control register as its destination address in order to write to the BRKA and BRKE bits during WAIT mode. The SBSW bit is set if the DMA writes to the break status and control register. SBSW is for applications that require a return to WAIT mode after exiting WAIT mode for a DMA-generated break interrupt. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. The SBSW bit is cleared by writing a ‘0’ to it.

10.5.2 STOP mode

A break interrupt causes exit from STOP mode and sets the SBSW bit in the SIM break status register. **7.7 SIM registers**

10

SECTION 11
MONITOR ROM (MON)

11.1 Introduction

This section describes the monitor ROM (MON08). The monitor ROM allows complete testing of the MCU through a single-wire interface with a host computer.

11.2 Features

Features of the monitor ROM include the following:

- 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
- 4800 baud-28.8K baud communication with host computer
- Execution of code in RAM or ROM
- (E)EPROM programming

11.3 Functional description

The monitor ROM receives and executes commands from a host computer. Figure 11-1 shows a sample 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 pull-up resistor.

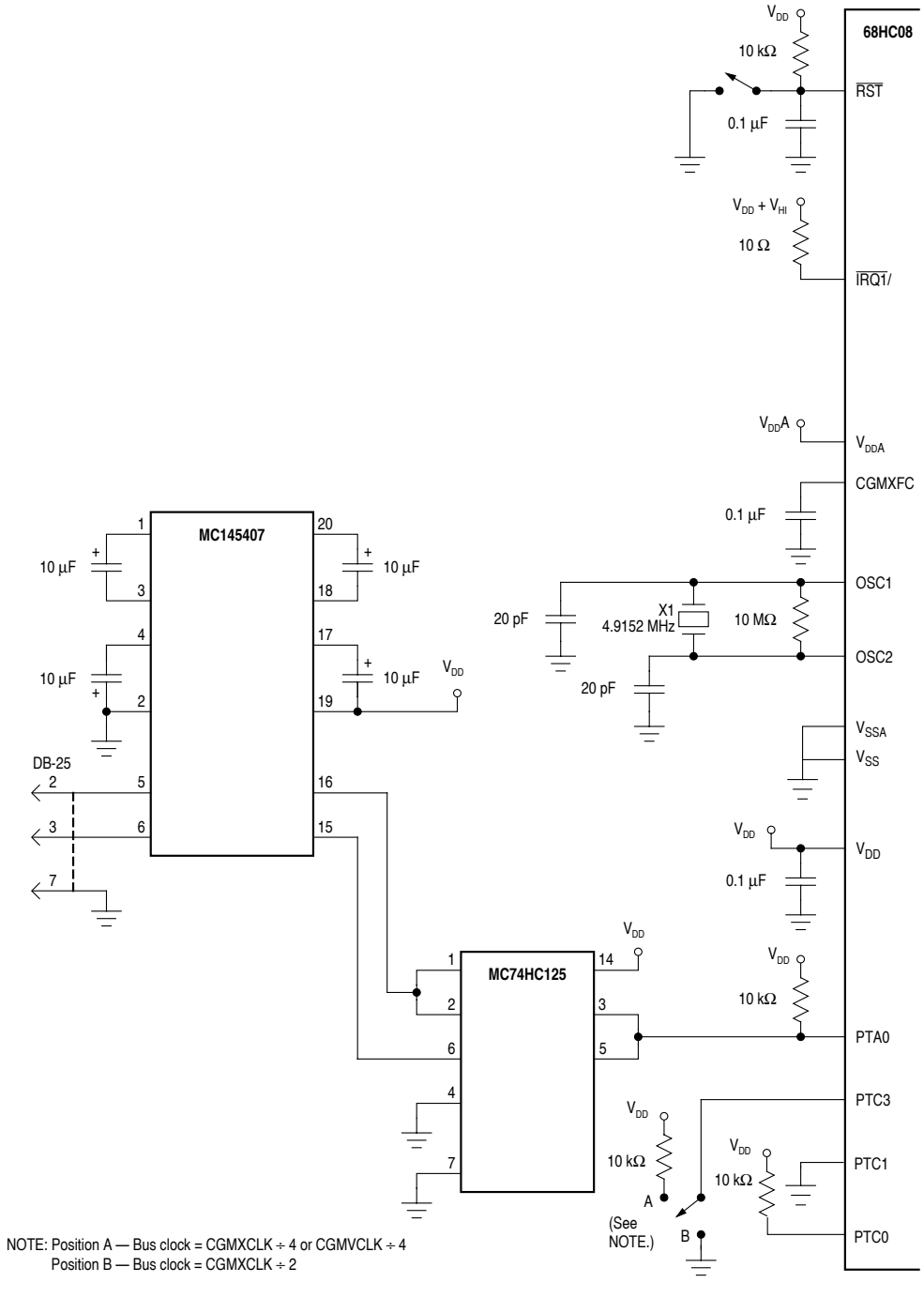


Figure 11-1 Monitor mode circuit

11.3.1 Entering monitor mode

Table 11-1 shows the pin conditions for entering monitor mode.

Table 11-1 Mode selection

$\overline{IRQ1}$ Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	Mode	CGMOUT	Bus Frequency
$V_{DD} + V_{HI}$	1	0	1	1	Monitor	$\frac{CGMXCLK}{2}$ or $\frac{CGMVCLK}{2}$	$\frac{CGMOUT}{2}$
$V_{DD} + V_{HI}$	1	0	1	0	Monitor	CGMXCLK	$\frac{CGMOUT}{2}$

Enter monitor mode by either

- Executing a software interrupt instruction (SWI) or
- Applying a ‘0’ and then a ‘1’ to the $\overline{RST}$ pin.

The MCU sends a break signal (10 consecutive ‘0’s) to the host computer, indicating that it is ready to receive a command. The break signal also provides a timing reference to allow the host to determine the necessary baud rate.

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. 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 $V_{DD} + V_{HI}$ is applied to either the $\overline{IRQ1}$ pin or the $\overline{RST}$ pin. See **SECTION 7 SYSTEM INTEGRATION MODULE (SIM)** for more information on modes of operation.

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.

Table 11-2 is a summary of the differences between user mode and monitor mode.

Table 11-2 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(1)	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

1. If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{IRQ1}$/V_{PP} 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.

11.3.2 Data format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See Figure 11-2 and Figure 11-3).

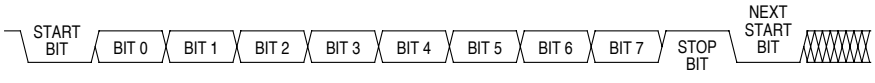


Figure 11-2 Monitor data format

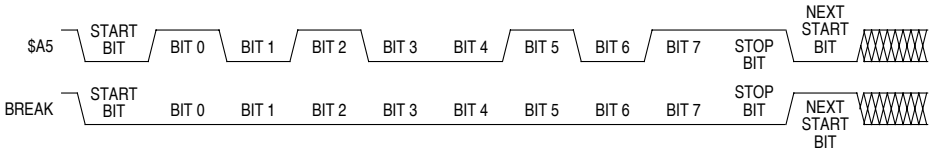


Figure 11-3 Sample monitor waveforms

The data transmit and receive rate can be anywhere from 4800 baud to 28.8K baud. Transmit and receive baud rates must be identical.

11.3.3 Echoing

The monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking, as shown in Figure 11-4.

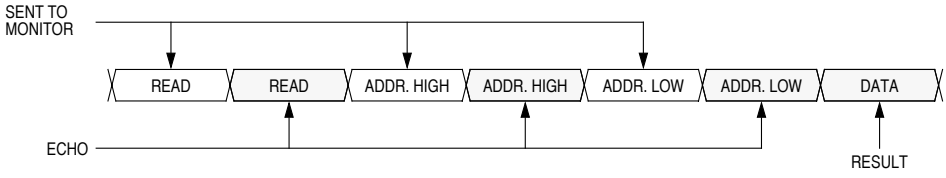


Figure 11-4 Read transaction

Any result of a command appears after the echo of the last byte of the command.

11.3.4 Break signal

A break signal is a start bit followed by nine low bits. This is shown in Figure 11-5. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

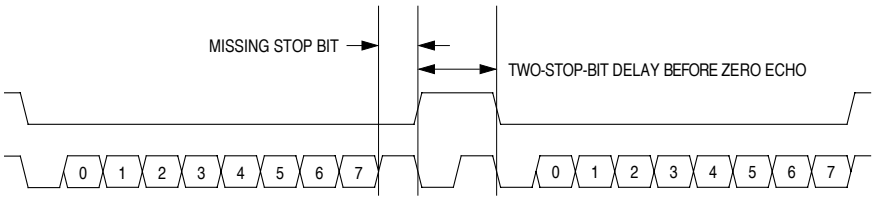


Figure 11-5 Break transaction

11.3.4.1 Commands

The monitor ROM uses the following commands:

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

Table 11-3 READ (read memory) command

Description	Read byte from memory
Operand	Specifies 2-byte address in high byte:low byte order
Data returned	Returns contents of specified address
Opcode	$4A

Table 11-3 READ (read memory) command

Command sequence

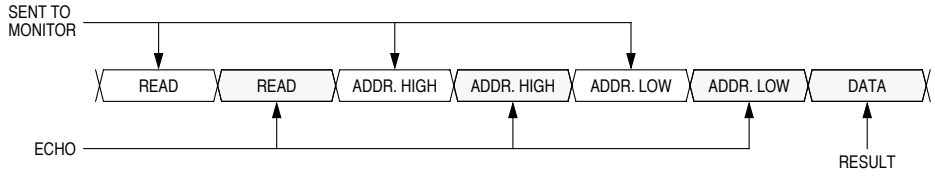


Table 11-4 WRITE (write memory) command

Description	Write byte to memory
Operand	Specifies 2-byte address in high byte:low byte order; low byte followed by data byte
Data returned	None
Opcode	$49

Command sequence

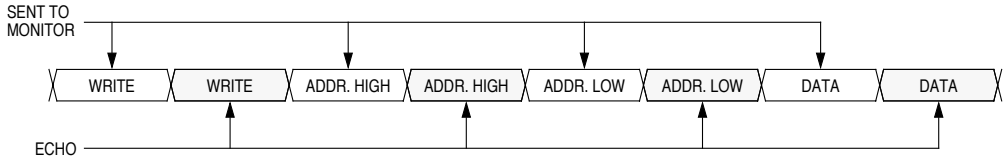


Table 11-5 IREAD (indexed read) command

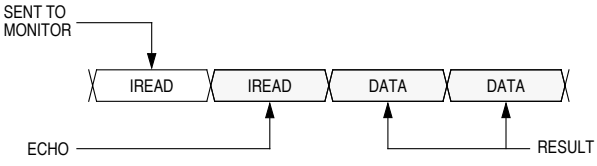
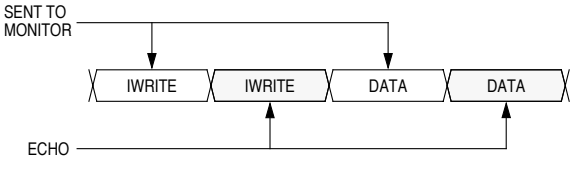
Description	Read next 2 bytes in memory from last address accessed
Operand	Specifies 2-byte address in high byte:low byte order
Data returned	Returns contents of next two addresses
Opcode	$1A
Command sequence	

Table 11-6 IWRITE (indexed write) command

Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data returned	None
Opcode	$19
Command sequence	

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

Table 11-7 READSP (read stack pointer) command

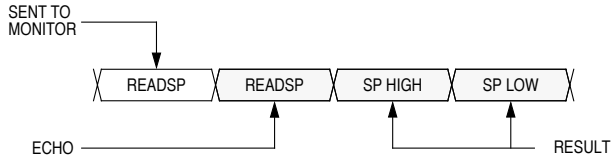
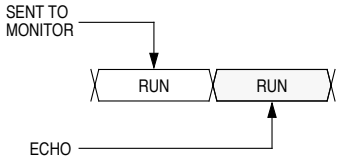
Description	Reads stack pointer
Operand	None
Data returned	Returns stack pointer in high byte:low byte order
Opcode	$0C
Command sequence	

Table 11-8 RUN (run user program) command

Description	Executes RTI instruction
Operand	None
Data returned	None
Opcode	$28
Command sequence	

11.3.5 Baud rate

With a 4.9152MHz crystal and the PTC3 pin at ’1’ during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at ‘0’ during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). Refer to **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.

Table 11-9 Monitor baud rate selection

	VCO Frequency Multiplier (N)					
	1	**2**	**3**	**4**	**5**	**6**
Monitor Baud Rate	4800	9600	14,400	19,200	24,000	28,800

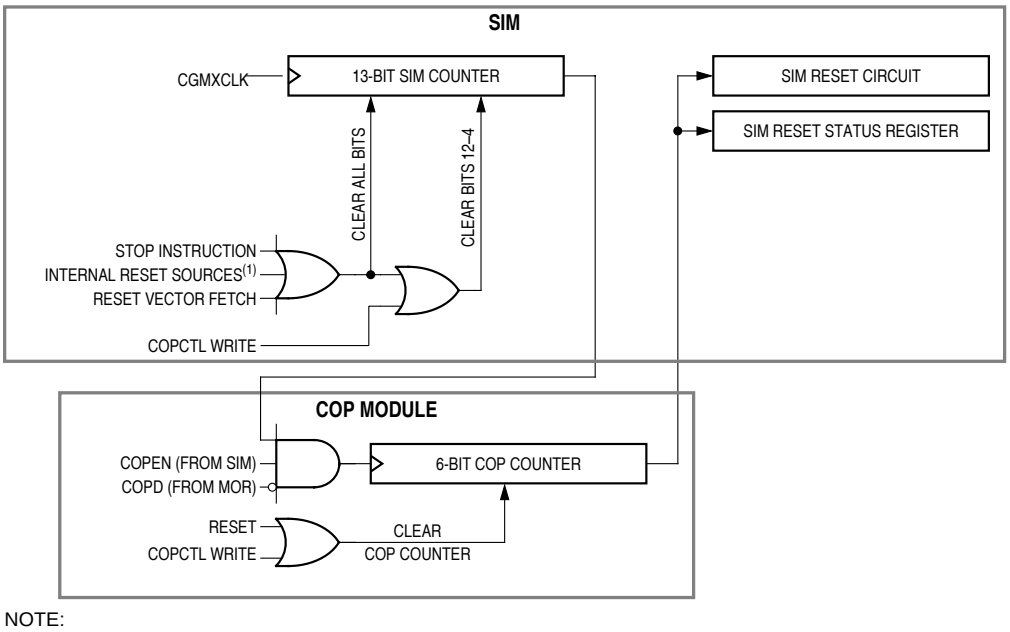
SECTION 12
COMPUTER OPERATING PROPERLY MODULE

12.1 Introduction

This section describes the computer operating properly (COP) module, a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. COP resets can be prevented by periodically clearing the COP counter.

12.2 Functional description

Figure 12-1 shows the structure of the COP module.



NOTE:
1. See **7.3.2 Active resets from internal sources**.

Figure 12-1 COP block diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit sprescaler. If not cleared by software, the COP counter overflows and generates an

Table 12-1 COP I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
COP Control Register (COPCTL)									$FFFF

asynchronous reset after $2^{13} - 2^4$ or $2^{18} - 2^4$ CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in MORA. When COPRS = 1, a 4.9152MHz crystal gives a COP timeout period of 53.3ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 5 through 12 of the prescaler.

NOTE

In Expanded mode location $FFFF will be external to the MCU. Therefore during the COP clearing operation, the peripheral located at $FFFF will also be written to.

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the SIM reset status register (SRSR). See **7.7.2 SIM reset status register (SRSR)**. The COP should be cleared immediately before entering or after exiting STOP mode to assure a full COP timeout period. A CPU interrupt routine or a DMA service routine can be used to clear the COP.

NOTE

COP clearing instructions should be placed 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.

12.3 I/O Signals

The following paragraphs describe the signals shown in Figure 12-1.

12.3.1 CGMXCLK

CGMXCLK is the crystal oscillator output signal. The CGMXCLK frequency is equal to the crystal frequency.

12.3.2 STOP instruction

The STOP instruction clears the SIM counter.

12.3.3 COPCTL write

Writing any value to the COP control register (COPCTL) (See **12.4 COP Control register (COPCTL)**) clears the COP counter and clears bits 12 – 4 of the SIM counter. Reading the COP control register returns the reset vector.

12.3.4 Power-on reset

The power-on reset (POR) circuit in the SIM clears the SIM counter 4096 CGMXCLK cycles after power-up.

12.3.5 Internal reset

An internal reset clears the SIM counter and the COP counter.

12.3.6 Reset vector fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the SIM counter.

12.3.7 COPD (COP disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register (CONFIG). See **SECTION 9 MASK OPTIONS**.

12.4 COP Control register (COPCTL)

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.

		Bit 7	6	5	4	3	2	1	Bit 0
COPCTL $FFFF	Read:	Low byte of reset vector							
	Write:	Clear COP counter							
	Reset:	Unaffected by reset							

Figure 12-2 COP control register (COPCTL)

12.5 Interrupts

The COP does not generate CPU interrupt requests or DMA service requests.

12.6 Monitor mode

The COP is disabled in monitor mode when V_{DD} + V_{HI} is present on the $\overline{IRQ1}$ pin or on the $\overline{RST}$ pin.

12.7 Low-power modes

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

12.7.1 WAIT mode

The COP continues to operate during WAIT mode. To prevent a COP reset during WAIT mode, the COP counter should be cleared periodically in a CPU interrupt routine or a DMA service routine.

12.7.2 STOP mode

STOP mode turns off the CGMXCLK input to the COP and clears the SIM counter. The COP should be serviced immediately before entering or after exiting STOP mode to ensure a full COP timeout period after entering or exiting STOP mode.

The STOP bit in the mask option register (MOR) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, the STOP instruction should be disabled by programming the STOP bit to ‘0’.

12.8 COP module during break interrupts

The COP is disabled during a break interrupt when V_{DD} + V_{HI} is present on the $\overline{RST}$ pin.

SECTION 13
LOW-VOLTAGE INHIBIT MODULE (LVI)

13.1 Introduction

This section describes the low-voltage inhibit module, which monitors the voltage on the V_{DD} pin and can force a reset when the V_{DD} voltage falls to the LVI trip voltage.

13.2 Features

Features of the LVI module include the following:

- Programmable LVI reset
- Programmable power consumption
- Digital filtering of VDD pin level

13.3 Functional description

Figure 13-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWRD, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRSTD, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}, and remains at or below that level for 9 or more consecutive CPU cycles. LVIPWRD and LVIRSTD are mask options. See **SECTION 9 MASK OPTIONS**. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

An LVI reset also drives the $\overline{RST}$ pin low to provide low-voltage protection to external peripheral devices.

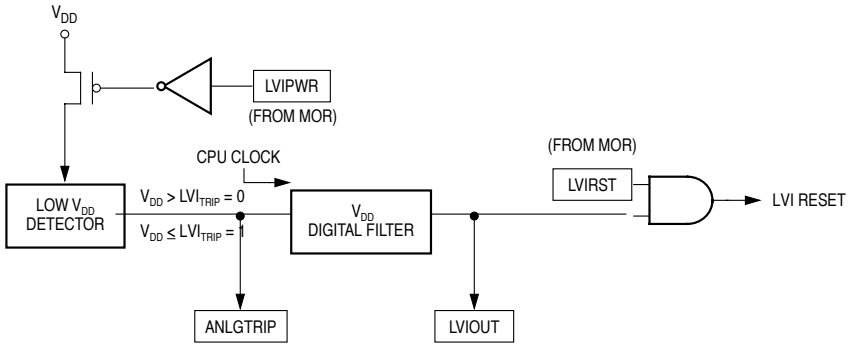


Figure 13-1 LVI module block diagram

Table 13-1 LVI I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
LVI Status Register (LVISR)	LVIOUT								$FE0F

= Unimplemented

13.3.1 Polled LVI operation

13

In applications that can operate at V_{DD} levels below the LVI_{TRIPF} level, software can monitor V_{DD} by polling the LVIOUT bit. In the mask option register, the LVIPWRD and LVIRSTD bits must be at ‘0’ to enable the LVI module, and to enable LVI resets. Also, the LVIPWRD bit must be at ‘0’ to enable the LVI module, and the LVIRSTD bit must be at ‘1’ to disable LVI resets.

13.3.2 Forced reset operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the LVI_{TRIPF} level and remains at or below that level for 9 or more consecutive CPU cycles. In the mask option register, the LVIPWRD and LVIRSTD bits must be at ‘1’ to enable the LVI module and to enable LVI resets.

13.3.3 False reset protection

The V_{DD} pin level is digitally filtered to reduce false resets due to power supply noise. In order for the LVI module to reset the MCU,V_{DD} must remain at or below the LVI_{TRIPF} level for 9 or more consecutive CPU cycles. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset.

13.4 LVI Status Register (LVISR)

The LVI status register flags V_{DD} voltages below the LVI_{TRIPF} level.

		Bit 7	6	5	4	3	2	1	Bit 0
LVISR $FE0F	Read:	LVIOUT	0	0	0	0	0	0	0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-2 LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when V_{DD} falls below the LVI_{TRIPF} voltage for 32-40 CGMXCLK cycles. (See Table 13-2). Reset clears the LVIOUT bit.

Table 13-2 LVIOUT bit indication

V_{DD}		LVIOUT
at level:	for number of CGMXCLK cycles:	
$V_{DD} > LVI_{TRIPR}$	ANY	0
$V_{DD} < LVI_{TRIPF}$	< 32 CGMXCLK cycles	0
$V_{DD} < LVI_{TRIPF}$	between 32 & 40 CGMXCLK cycles	0 or 1
$V_{DD} < LVI_{TRIPF}$	> 40 CGMXCLK cycles	1
$LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$	ANY	Previous Value

13.5 LVI interrupts

The LVI module does not generate interrupt requests.

13.6 Low-power modes

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

13.6.1 WAIT mode

When the LVIPWR mask option is programmed to ‘0’, the LVI module is active after a WAIT instruction.

When the LVIRST mask option is programmed to ‘0’, the LVI module can generate a reset and bring the MCU out of WAIT mode.

13

SECTION 14
EXTERNAL INTERRUPT MODULE (IRQ)

14.1 Introduction

The IRQ module provides the nonmaskable interrupt input.

14.2 Features

Features of the IRQ module include the following:

- Dedicated external interrupt pins ($\overline{\text{IRQ1}}$)
- IRQ1 interrupt control bit
- Hysteresis buffer
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge

14.3 Functional description

A ’0’ applied to any of the external interrupt pins 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 occurs:

- Vector fetch — a vector fetch automatically generates an interrupt acknowledge signal which clears the latch that caused the vector fetch.
- Software clear — software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a ’1’ to the ACK1 bit clears the IRQ1 latch.
- Reset — a reset automatically clears the interrupt latch.

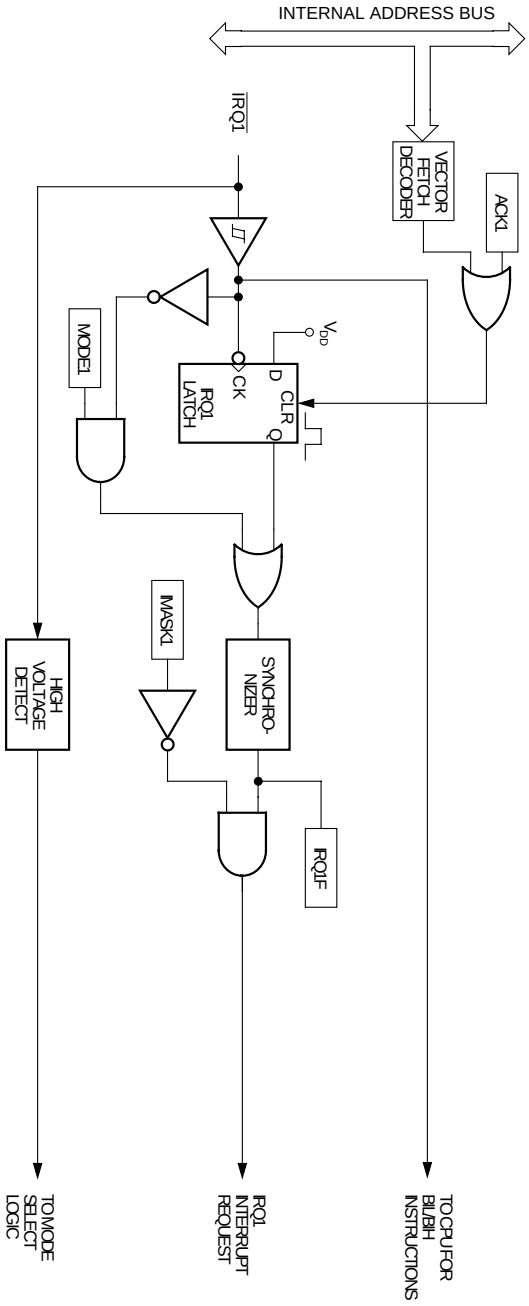


Figure 14-1. IRQ module block diagram

Table 14-1 IRQ I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
IRQ Status/Control Register (ISCR)					IRQF1	ACK1	IMASK1	MODE1	$001A

All of the external interrupt pins are falling-edge-triggered and are software-configurable to be both falling-edge and low-level-triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin.

When an interrupt pin is edge-triggered only, the interrupt latch remains set until a vector fetch, software clear, or reset occurs.

When an interrupt pin is both falling-edge and low-level-triggered, the interrupt latch remains set until both of the following occur:

- Vector fetch or software clear
- Return of the interrupt pin to ’1’

The vector fetch or software clear may occur before or after the interrupt pin returns to ’1’. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODEx1control 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 corresponding IMASK bit is clear.

14

NOTE

The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. See Figure 14-2.

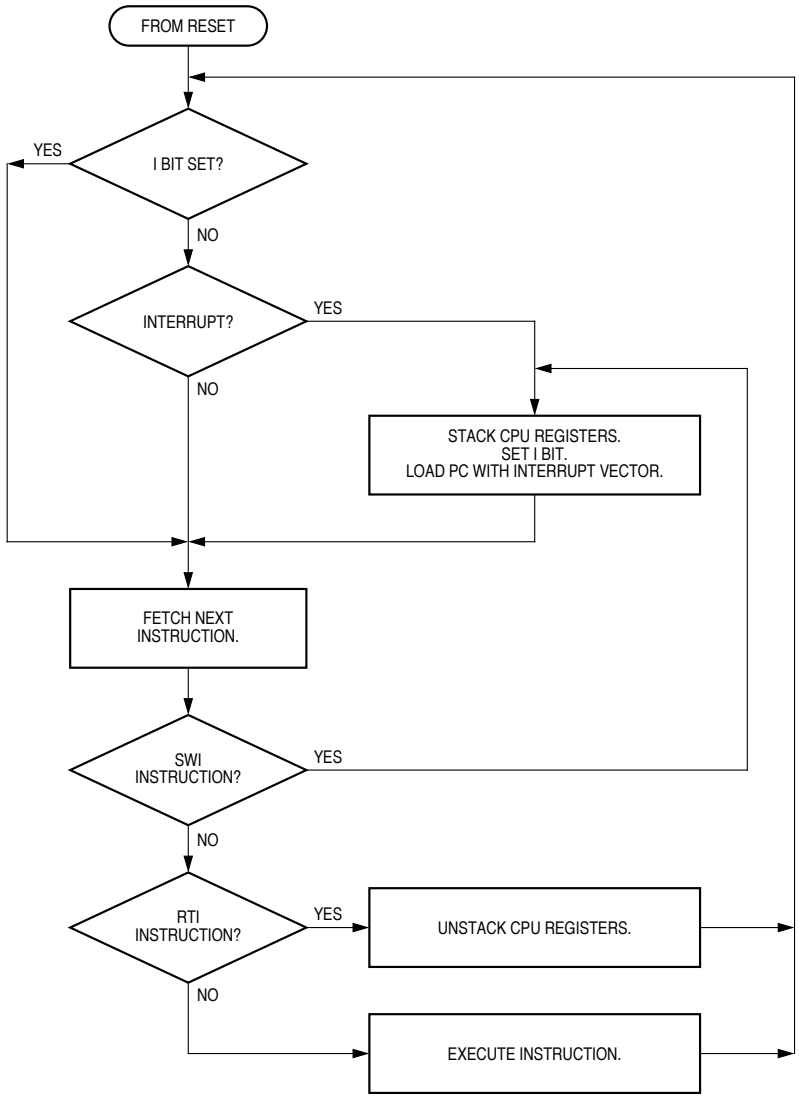


Figure 14-2 IRQ interrupt flowchart

14.3.1 $\overline{\text{IRQ1}}$ pin

A ’0’ on the $\overline{\text{IRQ1}}$ pin can latch an interrupt request into the IRQ1 latch. A vector fetch, software clear, or reset clears the 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 the IRQ1 latch:

- 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 ’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 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 ’1’ — as long as the $\overline{\text{IRQ1}}$ pin is at ’0’, the IRQ1 latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ1}}$ pin to ’1’ may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ1}}$ pin is at ’0’. A reset will clear the latch and the MODEx 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.

The BIH or BIL instruction is used to read the logic level on the $\overline{\text{IRQ1}}$ pin.

NOTE

When using the level-sensitive interrupt trigger, false interrupts can be avoided by masking interrupt requests in the interrupt routine.

14.4 IRQ module during break interrupts

The system integration module (SIM) controls whether the IRQ1 interrupt latch can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latches during the break state. See **7.7.3 SIM break flag control register (SBFCR)**.

To allow software to clear the IRQ1 latch during a break interrupt, a ’1’ is written 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, a ’0’ is written to the BCFE bit. With BCFE at ’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.

14.5 IRQ status and control register (ISCR)

The IRQ status and control register (ISCR) controls and monitors operation of the IRQ module. The ISCR performs the following functions:

- Indicates the state of the IRQ1 interrupt flag
- Clears the IRQ1 interrupt latch
- Masks IRQ1 interrupt requests
- Controls triggering sensitivity of the $\overline{\text{IRQ1}}$ interrupt pin

Figure 14-3 IRQ status and control register (ISCR)

IRQ1F — IRQ1 flag

This read-only status bit is high when the IRQ1 interrupt is pending.
1 = Interrupt pending
0 = Interrupt not pending

ACK1 — IRQ1 interrupt request acknowledge bit

Writing a ’1’ to this write-only bit clears the IRQ1 latch. ACK1 always reads as ’0’. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt mask bit

Writing a ’1’ to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.
1 = IRQ1 interrupt requests disabled
0 = IRQ1 interrupt requests enabled

MODE1 — IRQ1 edge/level select bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}/V_{PP}$ pin. Reset clears MODE1.
1 = Interrupt requests on falling edges and low levels
0 = Interrupt requests on falling edges only

SECTION 15
SERIAL COMMUNICATIONS INTERFACE MODULE (SCI)

15.1 Introduction

This section describes the serial communications interface module, which allows high-speed asynchronous communications with peripheral devices and other MCUs.

NOTE

DMA associated functions are only valid if the MCU has a DMA module.

15.2 Features

Features of the SCI module include the following:

- Full duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 32 programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Separate receiver and transmitter DMA service requests
- Programmable transmitter output polarity
- Two receiver wake-up methods:
  - Idle line wake-up
  - Address mark wake-up
- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error

  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

15.3 Functional description

Figure 15-1 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data. During DMA transfers, the DMA fetches data from memory for the SCI to transmit and/or the DMA stores received data in memory.

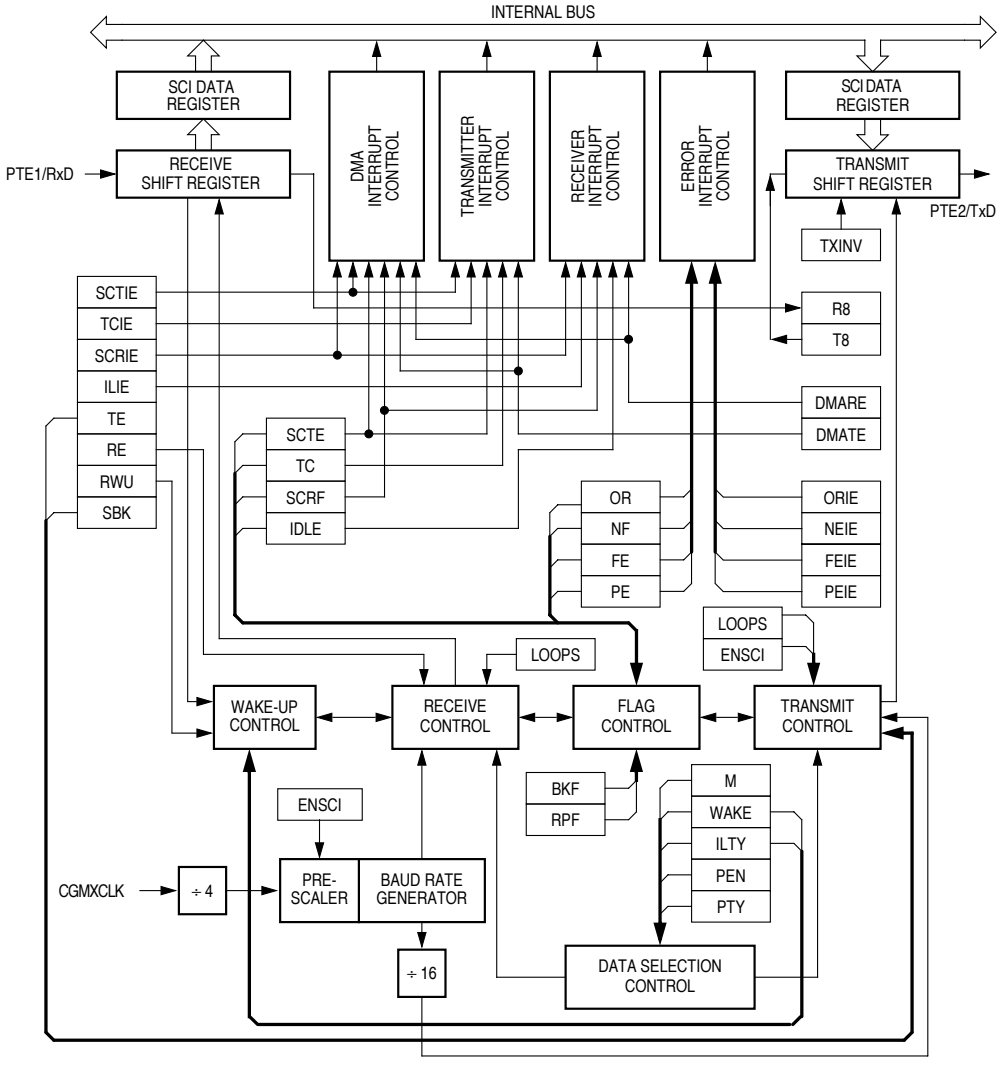


Figure 15-1 SCI module block diagram

Table 15-1 SCI I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Control Register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI Control Register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI Control Register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI Status Register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016

Table 15-1 SCI I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Status Register 2 (SCS2)							BKF	RPF	$0017
SCI Data Register (SCDR)									$0018
SCI Baud Rate Register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

15.3.1 Data format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 15-2.

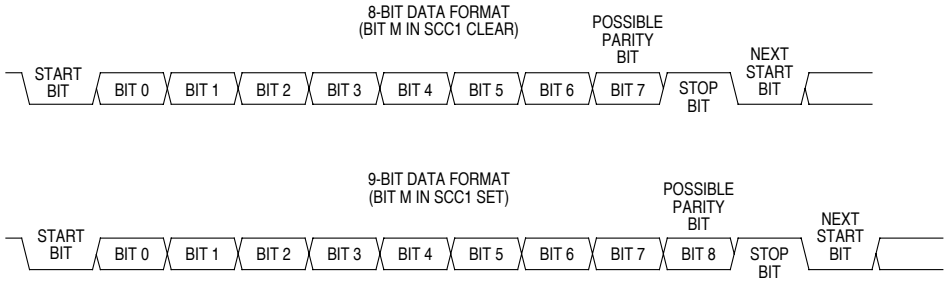


Figure 15-2 SCI data formats

15

15.3.2 Transmitter

Figure 15-3 shows the structure of the SCI transmitter.

Character length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in SCI control register 3 (SCC3) is the ninth bit (bit 8).

Character transmission

During an SCI transmission, the transmit shift register shifts a character out to the PTE0/TxD pin. The SCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register. To initiate an SCI transmission:

1. Enable the SCI by writing a ’1’ to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).

2. Enable the transmitter by writing a ’1’ to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit by first reading SCI status register 1 (SCS1) and then writing to the SCDR. In a DMA transfer, the DMA automatically clears the SCTE bit by writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of ’1’s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A ’0’ start bit automatically goes into the least significant bit position of the transmit shift register. A ’1’ STOP bit goes into the most significant bit position.

The SCI transmitter empty bit, SCTE, in SCS1 becomes set when the SCDR transfers a byte to the transmit shift register. The SCTE bit indicates that the SCDR can accept new data from the internal data bus. If the SCI transmit interrupt enable bit, SCTIE, in SCC2 is also set, the SCTE bit generates a transmitter CPU interrupt request or a transmitter DMA service request.

The SCTE bit generates a transmitter DMA service request if the DMA transfer enable bit, DMATE, in SCI control register 3 (SCC3) is set. Setting the DMATE bit enables the SCTE bit to generate transmitter DMA service requests and disables transmitter CPU interrupt requests.

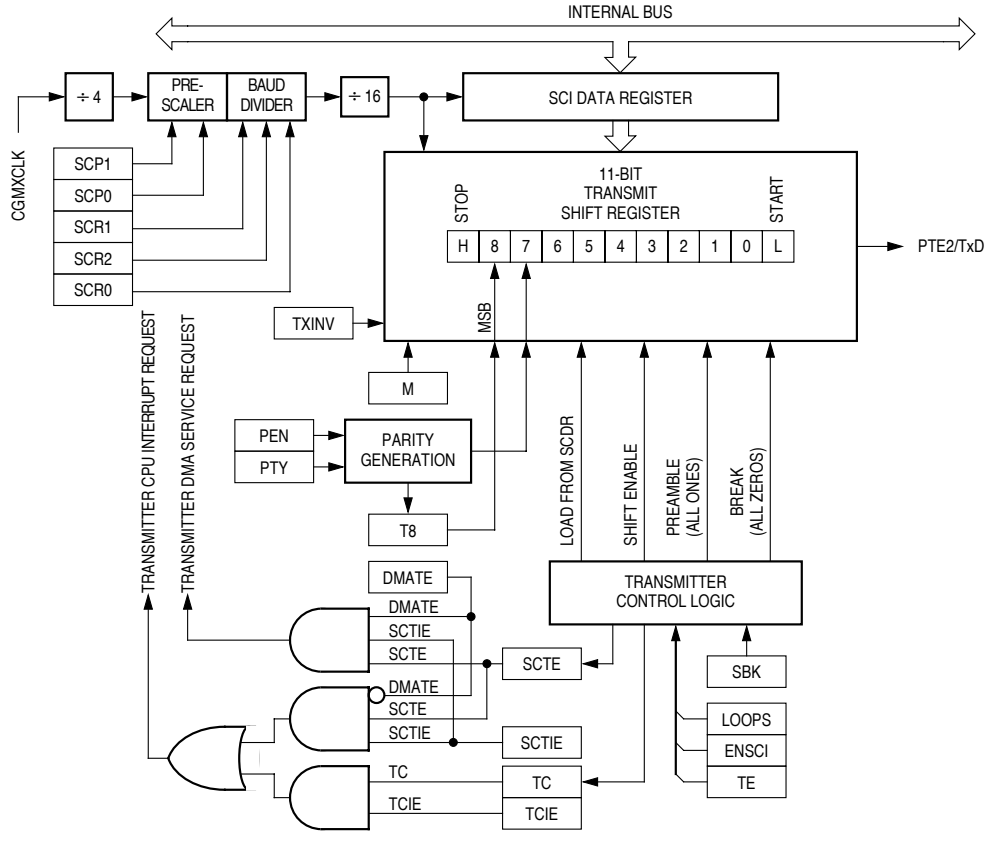


Figure 15-3 SCI transmitter

Table 15-2 SCI transmitter I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Control Register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI Control Register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI Control Register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI Status Register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016
SCI Data Register (SCDR)									$0018
SCI Baud Rate Register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

When the transmit shift register is not transmitting a character, the PTE0/TxD pin goes to the idle condition, ’1’. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

Break characters

Writing a ’1’ to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all ’0’s and has no start, STOP, or parity bit. Break character length depends on the M bit in SCC1. As long as SBK is at ’1’, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one ’1’. The automatic ’1’ at the end of a break character guarantees the recognition of the start bit of the next character.

The SCI recognizes a break character when a start bit is followed by 8 or 9 ’0’ data bits and a ’0’ where the STOP bit should be. Receiving a break character has the following effects on SCI registers:

- Sets the framing error bit (FE) in SCS1
- Sets the SCI receiver full bit (SCRF) in SCS1
- Clears the SCI data register (SCDR)
- Clears the R8 bit in SCC3
- Sets the break flag bit (BKF) in SCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

Idle characters

An idle character contains all ’1’s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the PTE2/TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

NOTE

When queueing an idle character, return the TE bit to ’1’ before the stop bit of the current character shifts out to the PTE0/TxD pin. Setting TE after the stop bit appears on PTE0/TxD causes data previously written to the SCDR to be lost.

A good time to toggle the TE bit is when the SCTE bit becomes set and just before writing the next byte to the SCDR.

Inversion of transmitted output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at '1'. See **15.7.1 SCI control register 1 (SCC1)**.

Transmitter interrupts

The following conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request or a transmitter DMA service request. Setting the SCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter CPU interrupt requests. Setting both the SCTIE bit and the DMA transfer enable bit, DMATE, in SCC3 enables the SCTE bit to generate transmitter DMA service requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in SCC2 enables the TC bit to generate transmitter CPU interrupt requests.

15.3.3 Receiver

Figure 15-4 shows the structure of the SCI receiver.

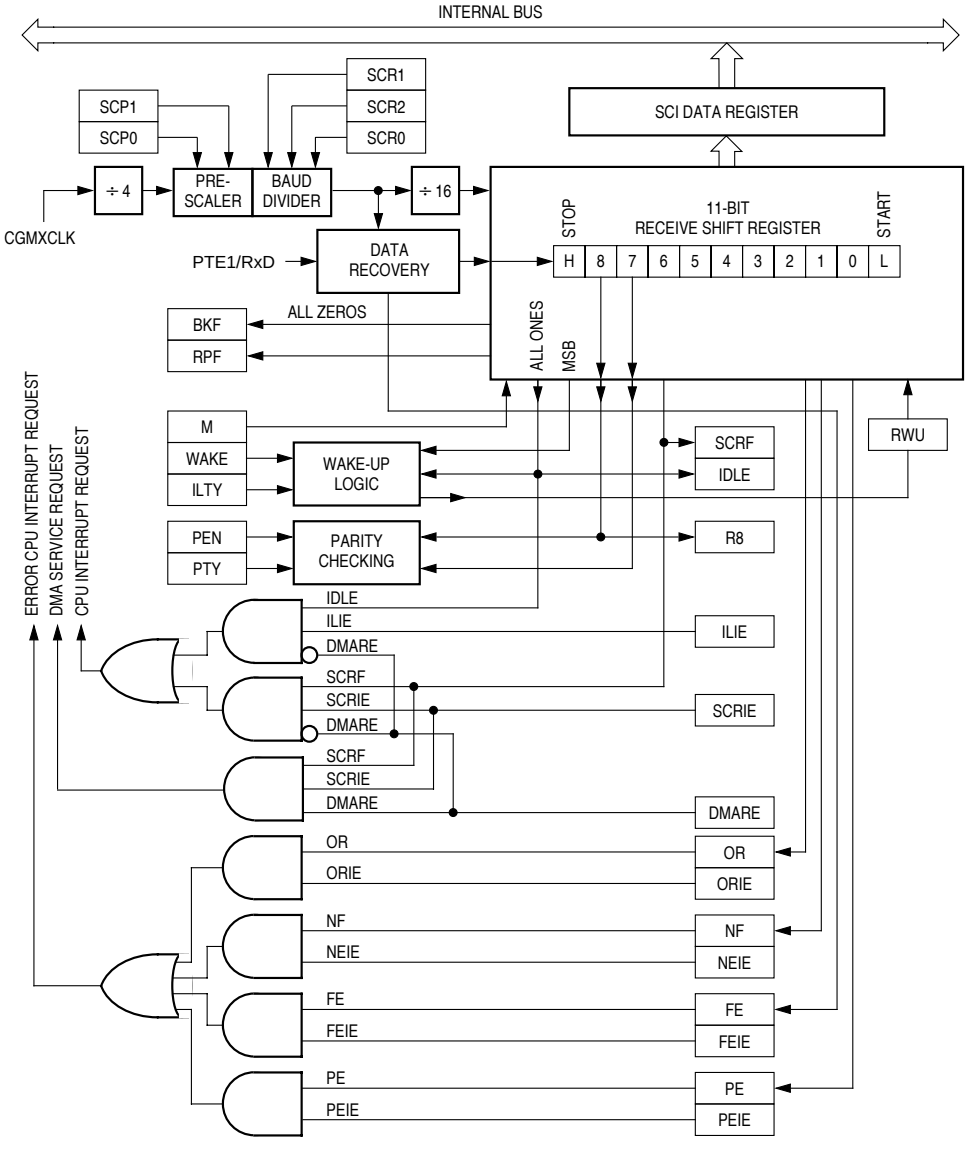


Figure 15-4 SCI receiver block diagram

Table 15-3 SCI receiver I/O register summary

Register name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI control register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI control register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI control register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI status register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016
SCI status register 2 (SCS2)							BKF	RPF	$0017
SCI data register (SCDR)									$0018
SCI baud rate register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

Character length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

Character reception

During an SCI reception, the receive shift register shifts characters in from the PTE1/RxD pin. The SCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The SCI receiver full bit, SCRF, in SCI status register 1 (SCS1) becomes set, indicating that the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates a receiver CPU interrupt request or a receiver DMA service request.

The SCRF bit generates a receiver DMA service request if the DMA receive enable bit, DMARE, in SCI control register 3 (SCC3) is set. Setting the DMARE bit enables the SCRF bit to generate receiver DMA service requests and disables receiver CPU interrupt requests.

Data sampling

The receiver samples the PTE1/RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see **Figure 14-5**):

- After every start bit
- After the receiver detects a data bit change from ’1’ to ’0’ (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid ’1’ and the majority of the next RT8, RT9, and RT10 samples returns a valid ’0’)

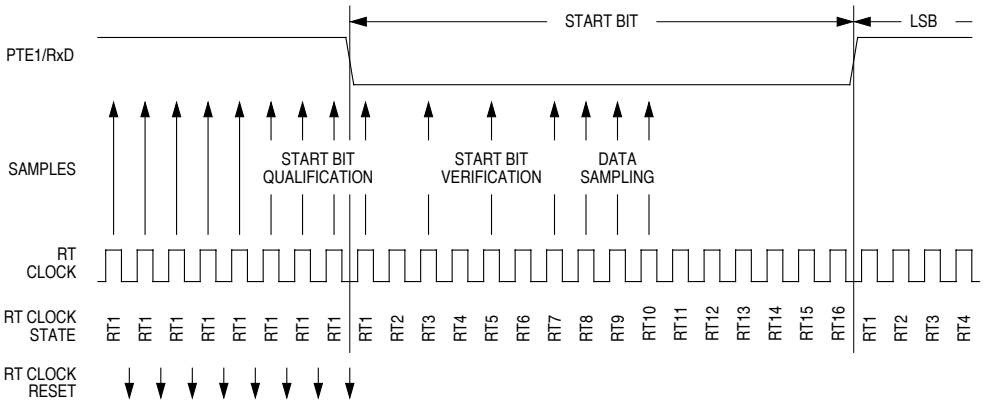


Figure 15-5 Receiver data sampling

To locate the start bit, data recovery logic does an asynchronous search for a ’0’ preceded by three ’1’s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 14-4** summarizes the results of the start bit verification samples.

15

Table 15-4 Start bit verification

RT3, RT5, and RT7 samples	Start bit verification	Noise flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 15-5 summarizes the results of the data bit samples.

Table 15-5 Data bit recovery

RT8, RT9, and RT10 Samples	Data bit determination	Noise flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are ’1’s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 15-6 summarizes the results of the stop bit samples.

Table 15-6 Stop bit recovery

RT8, RT9, and RT10 samples	Framing error flag	noise flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

Framing errors

If the data recovery logic does not detect a ’1’ where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. The FE flag is set at the same time that the SCRF bit is set. A break character that has no stop bit also sets the FE bit.

Receiver wake-up

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wake-up bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the PTE1/RxD pin can bring the receiver out of the standby state:

- Address mark — An address mark is a ’1’ in the most significant bit position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the SCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.
- Idle input line condition — When the WAKE bit is clear, an idle character on the PTE1/RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the SCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting ’1’s as idle character bits after the start bit or after the stop bit.

NOTE

Clearing the WAKE bit after the PTE1/RxD pin has been idle may cause the receiver to wake up immediately.

Receiver interrupts

The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request or a receiver DMA service request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit

to generate receiver CPU interrupts. Setting both the SCRIE bit and the DMA receive enable bit, DMARE, in SCC3 enables receiver DMA service requests and disables receiver CPU interrupt requests.

- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive ’1’s shifted in from the PTE1/RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

NOTE

When receiver DMA service requests are enabled (DMARE = 1), then receiver CPU interrupt requests are disabled, and the state of the ILIE bit has no effect.

Error interrupts

The following receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a ’0’ occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate SCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in SCS1 is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate SCI error CPU interrupt requests.

Error flags during DMA service requests

When the DMA is servicing the SCI receiver, it clears the SCRF bit when it reads the SCI data register. The DMA does not clear the other status bits (BKF or RPF), nor does it clear error bits (OR, NF, FE, and PE). If the error bits are enabled to generate interrupt requests, the interrupt requests may accumulate during DMA servicing. To clear error bits while the DMA is servicing the receiver, enable SCI error CPU interrupts and clear the bits in an interrupt routine. Note the following latency considerations:

1. If interrupt latency is short enough for an error bit to be serviced before the next SCRF, then it can be determined which byte caused the error. If

interrupt latency is long enough for a new SCRF to occur before servicing an error bit, then:

a. It cannot be determined whether the error bit being serviced is due to the byte in the SCI data register or to a previous byte. Multiple errors can accumulate that correspond to different bytes. In a message-based system, you may have to repeat the entire message

b. When the DMA is enabled to service the SCI receiver, merely reading the SCI data register clears the SCRF bit. The second step in clearing an error bit, reading the SCI data register, could inadvertently clear a new, unserviced SCRF that occurred during the error-servicing routine. Then the DMA would ignore the byte that set the new SCRF, and the new byte would be lost.
   To prevent clearing of an unserviced SCRF bit, clear the SCRIE bit at the beginning of the error-servicing interrupt routine and set it at the end. Clearing SCRIE disables DMA service so that both a read of SCS1 and a read of SCDR are required to clear the SCRF bit. Setting SCRIE enables DMA service so that the DMA can recognize a service request that occurred during the error-servicing interrupt routine.

c. In the CPU interrupt routine to service error bits, do not use BRSET or BRCLR instructions. BRSET and BRCLR read the SCS1 register, which is the first step in clearing the register. Then the DMA could read the SCI data register, the second step in clearing it, thereby clearing all error bits. The next read of the data register would miss any error bits that were set.

2. DMA latency should be short enough so that an SCRF is serviced before the next SCRF occurs. If DMA latency is long enough for a new SCRF to occur before servicing an error bit, then:

   a. Overruns occur. Set the ORIE bit to enable SCI error CPU interrupt requests and service the overrun in an interrupt routine. In a message-based system, disable the DMA in the interrupt routine and manually recover. Otherwise, the byte that was lost in the overrun could prevent the DMA from reaching its byte count. If the DMA reaches it byte count in the following message, two messages may be corrupted.

   b. If the CPU does not service an overrun interrupt request, the DMA can eventually clear the SCRF bit by reading the SCI data register. The OR bit remains set. Each time a new byte sets the SCRF bit, new data transfers from the shift register to the SCI data register (provided that another overrun does not occur), even though the OR bit is set. The DMA removed the overrun condition by reading the data register, but the OR bit has not been cleared.

15.4 Low-power modes

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

15.4.1 Wait mode

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

If SCI module functions are not required during wait mode, reduce power consumption by disabling the module before executing the WAIT instruction.

The DMA can service the SCI without exiting wait mode.

15.4.2 STOP mode

The SCI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect SCI register states. SCI module operation resumes after an external interrupt.

Because the internal clock is inactive during stop mode, entering stop mode during an SCI transmission or reception results in invalid data.

15.5 SCI during break module interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during interrupts generated by the break module. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **7.7.3 SIM break flag control register (SBFCR)**.

To allow software to clear status bits during a break interrupt, write a ’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 ’0’ to the BCFE bit. With BCFE at 0 0 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 two-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 ’0’. After the break, doing the second step clears the status bit.

15.6 I/O signals

Port E shares two of its pins with the SCI module. The two SCI I/O pins are:

- PTE0/TxD — Transmit data
- PTE1/RxD — Receive data

15.6.1 PTE2/TxD (transmit data)

The PTE0/TxD pin is the serial data output from the SCI transmitter. The SCI shares the PTE0/TxD pin with port E. When the SCI is enabled, the PTE0/TxD pin is an output regardless of the state of the DDRE0 bit in data direction register E (DDRE).

15.6.2 PTE1/RxD (receive data)

The PTE1/RxD pin is the serial data input to the SCI receiver. The SCI shares the PTE1/RxD pin with port E. When the SCI is enabled, the PTE1/RxD pin is an input regardless of the state of the DDRE1 bit in data direction register E (DDRE).

15.7 I/O registers

The following I/O registers control and monitor SCI operation:

- SCI control register 1 (SCC1)
- SCI control register 2 (SCC2)
- SCI control register 3 (SCC3)
- SCI status register 1 (SCS1)
- SCI status register 2 (SCS2)
- SCI data register (SCDR)
- SCI baud rate register (SCBR)

15.7.1 SCI control register 1 (SCC1)

SCI control register 1 does the following:

- Enables loop mode operation
- Enables the SCI
- Controls output polarity
- Controls character length
- Controls SCI wake-up method
- Controls idle character detection
- Enables parity function
- Controls parity type

		Bit 7	6	5	4	3	2	1	Bit 0
SCC1 $0013	Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Reset:	0	0	0	0	0	0	0	0

Figure 15-6 SCI control register 1 (SCC1)

LOOPS — Loop mode select bit

This read/write bit enables loop mode operation. In loop mode the PTE1/RxD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. Reset clears the LOOPS bit.

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI bit

This read/write bit enables the SCI and the SCI baud rate generator. Clearing ENSCI sets the SCTE and TC bits in SCI status register 1 and disables transmitter interrupts. Reset clears the ENSCI bit.

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit inversion bit

This read/write bit reverses the polarity of transmitted data. Reset clears the TXINV bit.

1 = Transmitter output inverted
0 = Transmitter output not inverted

NOTE

Setting the TXINV bit inverts all transmitted values, including idle, break, start, and stop bits.

M — Mode (character length) bit

This read/write bit determines whether SCI characters are 8 or 9 bits long. (See Table 15-7). The ninth bit can serve as an extra stop bit, as a receiver wake-up signal, or as a parity bit. Reset clears the M bit.

1 = 9-bit SCI characters
0 = 8-bit SCI characters

WAKE — wake-up condition bit

This read/write bit determines which condition wakes up the SCI: a ’1’ (address mark) in the most significant bit position of a received character or an idle condition on the PTE1/RxD pin. Reset clears the WAKE bit.

1 = Address mark wake-up
0 = Idle line wake-up

ILTY — Idle line type bit

This read/write bit determines when the SCI starts counting ’1’s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of ’1’s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

1 = Idle character bit count begins after stop bit
0 = Idle character bit count begins after start bit

PEN — Parity enable bit

This read/write bit enables the SCI parity function. (See Table 15-7). When enabled, the parity function inserts a parity bit in the most significant bit position. See Figure 15-2. Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. See Table 15-7. Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

NOTE

Changing the PTY bit in the middle of a transmission or reception can generate a parity error.

Table 15-7 Character format selection

Control Bits		Character Format				
M	**PEN:PTY**	**Start bits**	**Data bits**	**Parity**	**STOP bits**	**Character length**
0	0X	1	8	None	1	10 bits
1	0X	1	9	None	1	11 bits
0	10	1	7	Even	1	10 bits
0	11	1	7	Odd	1	10 bits
1	10	1	8	Even	1	11 bits
1	11	1	8	Odd	1	11 bits

15.7.2 SCI Control Register 2 (SCC2)

SCI control register 2 does the following:

- Enables the following CPU interrupt requests and DMA service requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests or transmitter DMA service requests
  - Enables the TC bit to generate transmitter CPU interrupt requests
  - Enables the SCRF bit to generate receiver CPU interrupt requests or receiver DMA service requests
  - Enables the IDLE bit to generate receiver CPU interrupt requests
- Enables the transmitter
- Enables the receiver
- Enables SCI wake-up
- Transmits SCI break characters

		Bit 7	6	5	4	3	2	1	Bit 0
SCC2 $0014	Read: / Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Reset:	0	0	0	0	0	0	0	0

Figure 15-7 SCI control register 2 (SCC2)

SCTIE — SCI transmit interrupt enable bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests or DMA service requests. Setting the SCTIE bit and clearing the DMA transfer enable bit, DMATE, in SCC3 enables the SCTE bit to generate CPU interrupt requests. Setting both the SCTIE and DMATE bits enables the SCTE bit to generate DMA service requests. Reset clears the SCTIE bit.

1 = SCTE enabled to generate CPU interrupt or DMA service requests
0 = SCTE not enabled to generate CPU interrupt or DMA service requests

TCIE — Transmission complete interrupt enable bit

This read/write bit enables the TC bit to generate SCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

1 = TC enabled to generate CPU interrupt requests
0 = TC not enabled to generate CPU interrupt requests

SCRIE — SCI receive interrupt enable bit

This read/write bit enables the SCRF bit to generate SCI receiver CPU interrupt requests or SCI receiver DMA service requests. Setting the SCRIE bit and clearing the DMA receive enable bit, DMARE, in SCC3 enables the SCRF bit to generate CPU interrupt requests. Setting both SCRIE and DMARE enables SCRF to generate DMA service requests. Reset clears the SCRIE bit.

1 = SCRF enabled to generate CPU interrupt or DMA service requests
0 = SCRF not enabled to generate CPU interrupt or DMA service requests

ILIE — Idle line interrupt enable bit

This read/write bit enables the IDLE bit to generate SCI receiver CPU interrupt requests. Reset clears the ILIE bit.

1 = IDLE enabled to generate CPU interrupt requests
0 = IDLE not enabled to generate CPU interrupt requests

NOTE

When SCI receiver DMA service requests are enabled (DMARE = 1), then SCI receiver CPU interrupt requests are disabled, and the state of the ILIE bit has no effect.

TE — Transmitter enable bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 ’1’s from the transmit shift register to the PTE2/TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the PTE2/TxD returns to the idle condition (’1’). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

1 = Transmitter enabled
0 = Transmitter disabled

NOTE

Writing to the TE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.

RE — Receiver enable bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits. Reset clears the RE bit.

1 = Receiver enabled
0 = Receiver disabled

NOTE

Writing to the RE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.

RWU — Receiver wake-up bit

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in SCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state
0 = Normal operation

SBK — Send break bit

Setting and then clearing this read/write bit transmits a break character followed by a ’1’. The ’1’ after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no ’1’s between them. Reset clears the SBK bit.

1 = Transmit break characters
0 = No break characters being transmitted

NOTE

Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK too early causes the SCI to send a break character instead of a preamble.

15.7.3 SCI control register 3 (SCC3)

SCI control register 3 does the following:

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted
- Enables SCI receiver full (SCRF) DMA service requests
- Enables SCI transmitter empty (SCTE) DMA service requests
- Enables the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

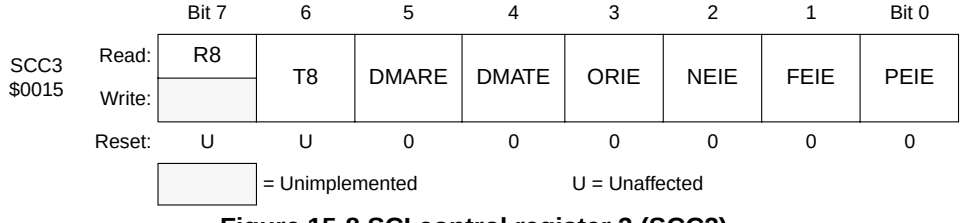


Figure 15-8 SCI control register 3 (SCC3)

R8 — Received bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.
When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted bit 8

When the SCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset has no effect on the T8 bit.

DMARE — DMA receive enable bit

This read/write bit enables the DMA to service SCI receiver DMA service requests generated by the SCRF bit. (See 14.7.4.) Setting the DMARE bit disables SCI receiver CPU interrupt requests. Reset clears the DMARE bit.

1 = DMA enabled to service SCI receiver DMA service requests generated by the SCRF bit
(SCI receiver CPU interrupt requests disabled)

0 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit
(SCI receiver CPU interrupt requests enabled)

DMATE — DMA transfer enable bit

This read/write bit enables SCI transmitter empty (SCTE) DMA service requests. See **15.7.4 SCI status register 1 (SCS1)**. Setting the DMATE bit disables SCTE CPU interrupt requests. Reset clears DMATE.

1 = SCTE DMA service requests enabled (SCTE CPU interrupt requests disabled)

0 = SCTE DMA service requests disabled (SCTE CPU interrupt requests enabled)

ORIE — Receiver overrun interrupt enable bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.

1 = SCI error CPU interrupt requests from OR bit enabled

0 = SCI error CPU interrupt requests from OR bit disabled

NEIE — Receiver noise error interrupt enable bit

This read/write bit enables SCI error CPU interrupt requests generated by the noise error bit, NE. Reset clears NEIE.

1 = SCI error CPU interrupt requests from NE bit enabled.

0 = SCI error CPU interrupt requests from NE bit disabled

FEIE — Receiver framing error interrupt enable bit

This read/write bit enables SCI error CPU interrupt requests generated by the framing error bit, FE. Reset clears FEIE.

1 = SCI error CPU interrupt requests from FE bit enabled

0 = SCI error CPU interrupt requests from FE bit disabled

PEIE — Receiver parity error interrupt enable bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. (See 13.7.4.) Reset clears PEIE.

1 = SCI error CPU interrupt requests from PE bit enabled

0 = SCI error CPU interrupt requests from PE bit disabled

15.7.4 SCI status register 1 (SCS1)

SCI status register 1 contains flags to signal the following conditions:

- Transfer of SCDR data to transmit shift register complete
- Transmission complete
- Transfer of receive shift register data to SCDR complete
- Receiver input idle
- Receiver overrun
- Noisy data
- Framing error
- Parity error

		Bit 7	6	5	4	3	2	1	Bit 0
SCS1 $0016	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
	Write:								
	Reset:	1	1	0	0	0	0	0	0

= Unimplemented

Figure 15-9 SCI status register 1 (SCS1)

SCTE — SCI transmitter empty bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request or an SCI transmitter DMA service request. When the SCTIE bit in SCC2 is set and the DMATE bit in SCC3 is clear, SCTE generates an SCI transmitter CPU interrupt request. With both the SCTIE and DMATE bits set, SCTE generates an SCI transmitter DMA service request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. In DMA transfers, the DMA automatically clears the SCTE bit when it writes to the SCDR. Reset sets the SCTE bit.

1 = SCDR data transferred to transmit shift register
0 = SCDR data not transferred to transmit shift register

NOTE

Setting the TE bit for the first time also sets the SCTE bit. When enabling SCI transmitter DMA service requests, set the TE bit **after** setting the DMATE bit. Otherwise setting the TE and SCTIE bits generates an SCI transmitter CPU interrupt request instead of a DMA service request.

TC — Transmission complete bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. When the DMA services an SCI transmitter DMA service request, the DMA clears the TC bit by writing to the SCDR. TC is automatically cleared when data, preamble or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI receiver full bit

This clearable, read-only bit is set when the data in the receive shift register transfers to the SCI data register. SCRF can generate an SCI receiver CPU interrupt request or an SCI receiver DMA service request. When the SCRIE bit in SCC2 is set and the DMARE bit in SCC3 is clear, SCRF generates a CPU interrupt request. With both the SCRIE and DMARE bits set, SCRF generates a DMA service request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. In DMA transfers, the DMA clears the SCRF bit when it reads the SCDR. Reset clears SCRF.

1 = Received data available in SCDR
0 = Data not available in SCDR

IDLE — Receiver idle bit

This clearable, read-only bit is set when 10 or 11 consecutive ’1’s appear on the receiver input. IDLE generates an SCI error CPU interrupt request if the ILIE bit in SCC2 is also set and the DMARE bit in SCC3 is clear. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

1 = Receiver input idle
0 = Receiver input active (or idle since the IDLE bit was cleared)

OR — Receiver overrun bit

This clearable, read-only bit is set when software fails to read the SCDR before the receive shift register receives the next character. The OR bit generates an SCI error CPU interrupt request if the ORIE bit in SCC3 is also set. The data in the shift register is lost, but the data already in the SCDR is not affected. Clear the OR bit by reading SCS1 with OR set and then reading the SCDR. Reset clears the OR bit.

1 = Receive shift register full and SCRF = 1
0 = No receiver overrun

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. Figure 15-10 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of SCS1 after reading the data register.

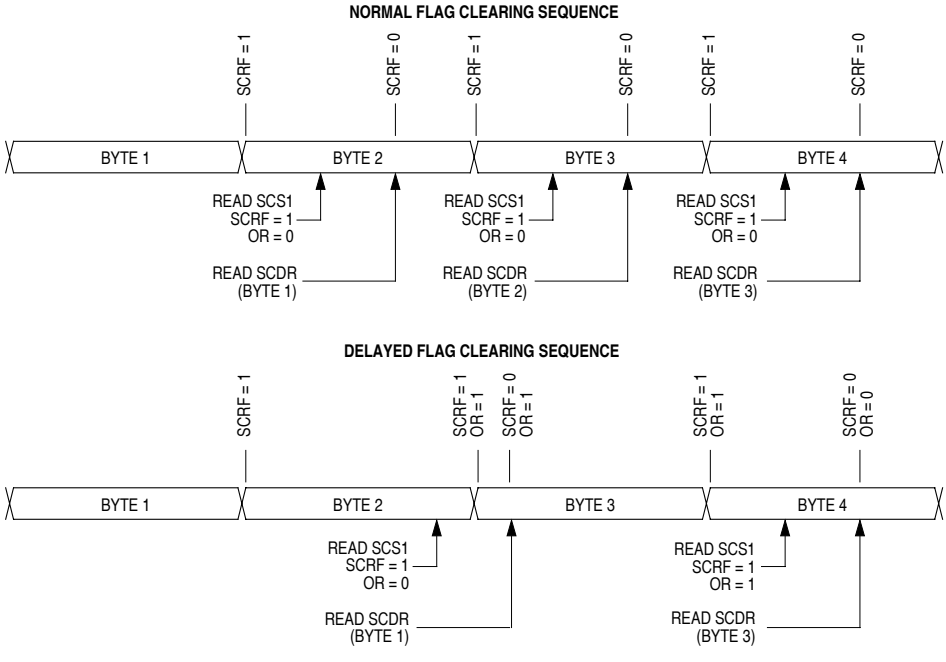


Figure 15-10 Flag clearing sequence

NF — Receiver noise flag bit

This clearable, read-only bit is set when the SCI detects noise on the PTE1/RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

FE — Receiver framing error bit

This clearable, read-only bit is set when a logic is accepted as the STOP bit. FE generates an SCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

PE — Receiver parity error bit

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates a PE CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

15.7.5 SCI status register 2 (SCS2)

SCI status register 2 contains flags to signal the following conditions:

- Break character detected
- Incoming data

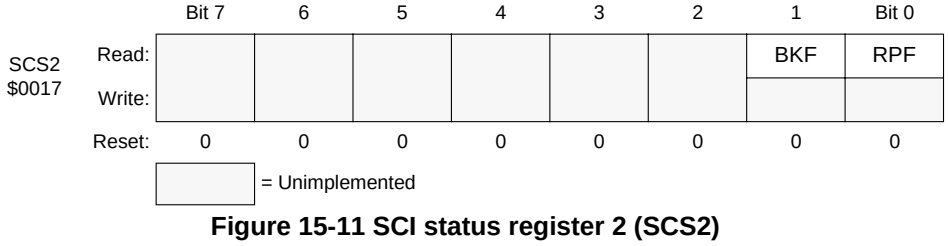


Figure 15-11 SCI status register 2 (SCS2)

BKF — Break flag bit

This clearable, read-only bit is set when the SCI detects a break character on the PTE1/RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate

a CPU interrupt request or a DMA service request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after ’1’s again appear on the PTE1/RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF —Reception in progress flag bit

This read-only bit is set when the receiver detects a ’0’ during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch, or when the receiver detects an idle character. Polling RPF before disabling the SCI module or entering STOP mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

15.7.6 SCI data register (SCDR)

The SCI data register is the buffer between the internal data bus and the receive and transmit shift registers. Reset has no effect on data in the SCI data register.

		Bit 7	6	5	4	3	2	1	Bit 0
SCDR $0018	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by reset							

Figure 15-12 SCI data register (SCDR)

R7/T7–R0/T0 — Receive/Transmit data bits

Reading address $0018 accesses the read-only received data bits, R7–R0. Writing to address $0018 writes the data to be transmitted, T7–T0. Reset has no effect on the SCI data register.

15.7.7 SCI baud rate register (SCBR)

The baud rate register selects the baud rate for both the receiver and the transmitter.

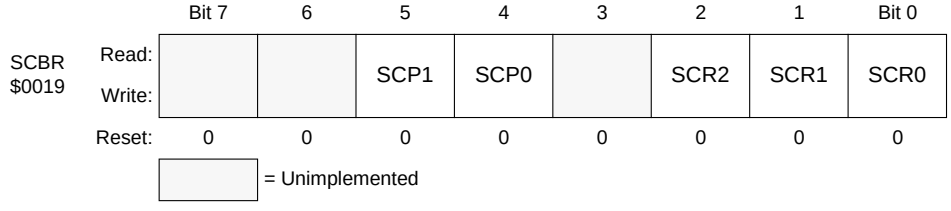


Figure 15-13 SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 15-8. Reset clears SCP1 and SCP0.

Table 15-8 SCI baud rate prescaling

SCP1:0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

15

SCR2–SCR0 — SCI baud rate select bits

These read/write bits select the SCI baud rate divisor as shown in Table 15-9. Reset clears SCR2–SCR0.

Table 15-9 SCI baud rate selection

SCR2:1:0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use the following formula to calculate the SCI baud rate:

$$\text{Baud rate} = \frac{f_{XCLK}}{64 \times PD \times BD}$$

where:

f_{XCLK} = bus frequency
PD = prescaler divisor
BD = baud rate divisor

Table 15-10 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 15-10 SCI baud rate selection examples

SCP1:0	Prescaler divisor (PD)	SCR2:1:0	Baud rate divisor (BD)	Baud rate (f_{XCLK} = 4.9152 MHz)
00	1	000	1	76,800
00	1	001	2	38,400
00	1	010	4	19,200
00	1	011	8	9600
00	1	100	16	4800
00	1	101	32	2400
00	1	110	64	1200
00	1	111	128	600
01	3	000	1	25,600
01	3	001	2	12,800
01	3	010	4	6400
01	3	011	8	3200
01	3	100	16	1600
01	3	101	32	800
01	3	110	64	400
01	3	111	128	200
10	4	000	1	19,200
10	4	001	2	9600
10	4	010	4	4800
10	4	011	8	2400

Table 15-10 SCI baud rate selection examples

SCP1:0	Prescaler divisor (PD)	SCR2:1:0	Baud rate divisor (BD)	Baud rate (f_{XCLK} = 4.9152 MHz)
10	4	100	16	1200
10	4	101	32	600
10	4	110	64	300
10	4	111	128	150
11	13	000	1	5908
11	13	001	2	2954
11	13	010	4	1477
11	13	011	8	739
11	13	100	16	369
11	13	101	32	185
11	13	110	64	92
11	13	111	128	46

SECTION 16
SERIAL PERIPHERAL INTERFACE MODULE (SPI)

16.1 Introduction

This section describes the serial peripheral interface module (SPI, Version C), which allows full-duplex, synchronous, serial communications with peripheral devices.

NOTE

DMA associated functions are only valid if the MCU has a DMA module.

16.2 Features

Features of the SPI module include the following:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts with DMA or CPU service:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with CPU interrupt capability
- Overflow error flag with CPU interrupt capability
- Programmable wired-OR mode
- I^2C (inter-integrated circuit) compatibility

16.3 Pin name conventions and I/O register addresses

The text that follows describes both SPI1 and SPI2. The SPI I/O pin names are $\overline{SS}$ (slave select), SPSCK (SPI serial clock), CGND (clock ground), MOSI (master out slave in), and MISO (master in slave out). The two SPIs share eight I/O pins with two parallel I/O ports. The full names of the SPI I/O pins are as follows:

Table 16-1 Pin name conventions

SPI Generic Pin Names:		MISO	MOSI	$\overline{SS}$	SCK	CGND
Full SPI PinNames:	SPI	PTE5/MISO	PTE6/MOSI	PTE4/$\overline{SS}$	PTE7/SPSCK	CGND

Table 16-2 I/O register addresses

Register name	Register address
SPI Control Register (SPICR)	$0010
SPI Status and Control Register (SPISCR)	$0011
SPI Data Register (SPIDR)	$0012

The generic pins names appear in the text that follows.

16.4 Functional description

Figure 16-1 summarizes the SPI I/O registers and Figure 16-2 show the structure of the SPI module.

Register name	R/W	Bit 7	6	5	4	3	2	1	Bit 0
SPI Control Register (SPCR)	Read:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
	Write:								
	Reset:	0	0	1	0	1	0	0	0
SPI Status/Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
	Write:								
	Reset:	0	0	0	0	1	0	0	0
SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by reset							

Figure 16-1 SPI I/O register summary

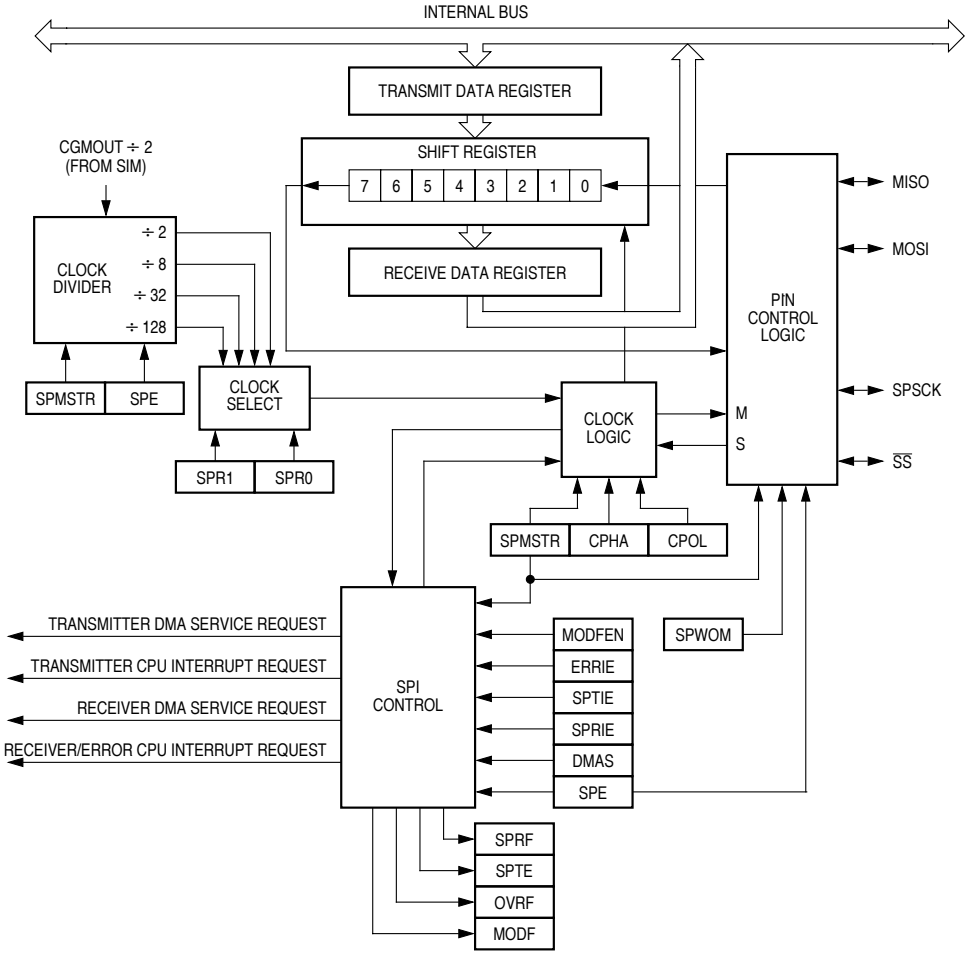


Figure 16-2 SPI module block diagram

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt-driven. All SPI interrupts can be serviced by the CPU, and the transmitter empty (SPTE) and receiver full (SPRF) flags can also be configured for DMA service.

During DMA transmissions, the DMA fetches data from memory for the SPI to transmit and/or the DMA stores received data in memory.

The following paragraphs describe the operation of the SPI module.

16.4.1 Master mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

NOTE

The SPI modules should be configured as master and slave before they are enabled. Also, the master SPI should be enabled before the slave SPI. Similarly, Disable the slave SPI should be disabled before disabling the master SPI. See **16.13.1 SPI control register (SPCR)**.

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. See Figure 16-3.

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. See **16.13.2 SPI status and control register (SPSCR)**. Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

As the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin. The transmission ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register.

16

When the DMAS bit is set, a read of the SPI data register by the CPU or by the DMA clears the SPRF bit. Regardless of the state of the DMAS bit, a write to the SPI data register by the CPU or by the DMA clears the SPTE bit.

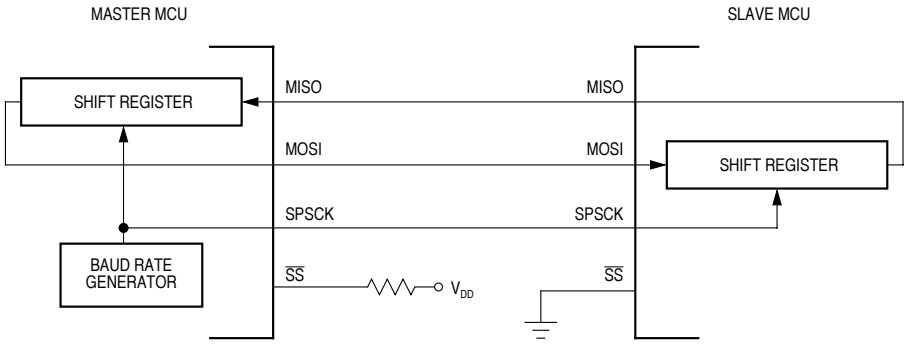


Figure 16-3 Full-duplex master-slave connections

16.4.2 Slave mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave MCU must be at '0'. $\overline{SS}$ must remain low until the transmission is complete. See **16.6.2 Mode fault error**.

In a slave SPI module, data enters the shift register under the control of the serial clock from the master SPI module. After a byte enters the shift register of a slave SPI, it transfers to the receive data register, and the SPRF bit is set. To prevent an overflow condition, slave software must then read the SPI data register before another byte enters the shift register.

The maximum frequency of the SPSCK for an SPI configured as a slave is the bus clock speed (which is twice as fast as the fastest master SPSCK clock that can be generated). The frequency of the SPSCK for an SPI configured as a slave does not have to correspond to any particular SPI baud rate. The baud rate only controls the speed of the SPSCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

A slave SPI must complete the write to the data register at least one bus cycle before the master SPI starts a transmission. When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. See **16.5 Transmission formats**.

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

NOTE

SPSCK must be in the proper idle state before the slave is enabled to prevent SPSCK from appearing as a clock edge.

16.5 Transmission formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling on the two serial data lines. A slave select line allows individual selection of a slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate a multiple-master bus contention.

16.5.1 Clock phase and polarity controls

Software can select any of four combinations of serial clock (SCK) phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transmission format.

The clock phase (CPHA) control bit selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

NOTE

Before writing to the CPOL bit or the CPHA bit, the SPI should be disabled by clearing the SPI enable bit (SPE).

16.5.2 Transmission format when CPHA = ’0’

Figure 16-4 shows an SPI transmission in which CPHA is ‘0’. The figure should not be used as a replacement for data sheet parametric information.Two waveforms are shown for SCK: one for CPOL = ‘0’ and another for CPOL = ‘1’. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only

when its slave select input ($\overline{SS}$) is at ‘0’, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general purpose I/O not affecting the SPI. See **16.6.2 Mode fault error**. When CPHA = ‘0’, the first SPSCK edge is the MSB capture strobe. Therefore the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the transmission. The $\overline{SS}$ pin must be toggled high and then low between each byte transmitted.

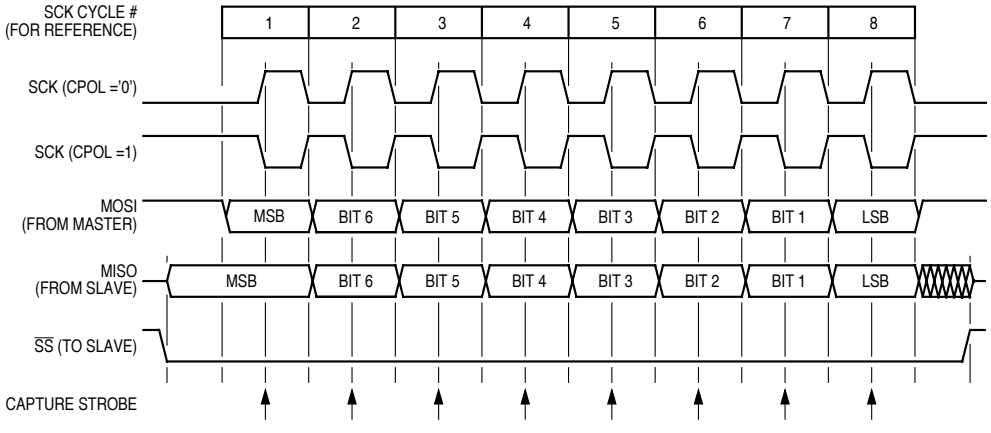


Figure 16-4 Transmission format (CPHA = ’0’)

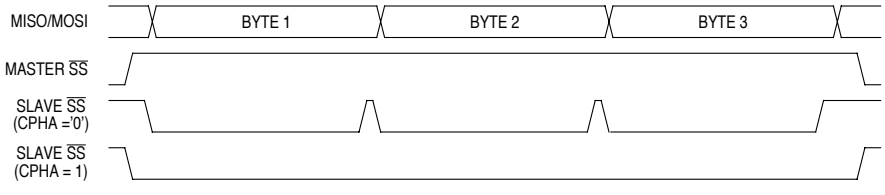


Figure 16-5 CPHA/$\overline{SS}$ timing

16.5.3 Transmission format when CPHA = ’1’

Figure 16-6 shows an SPI transmission in which CPHA is ‘1’. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = ‘0’ and another for CPOL = ‘1’. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at ’0’, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive.

The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. See **16.6.2 Mode fault error**. When CPHA = ‘1’, the master begins driving its MOSI pin on the first SPSCK edge. Therefore the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

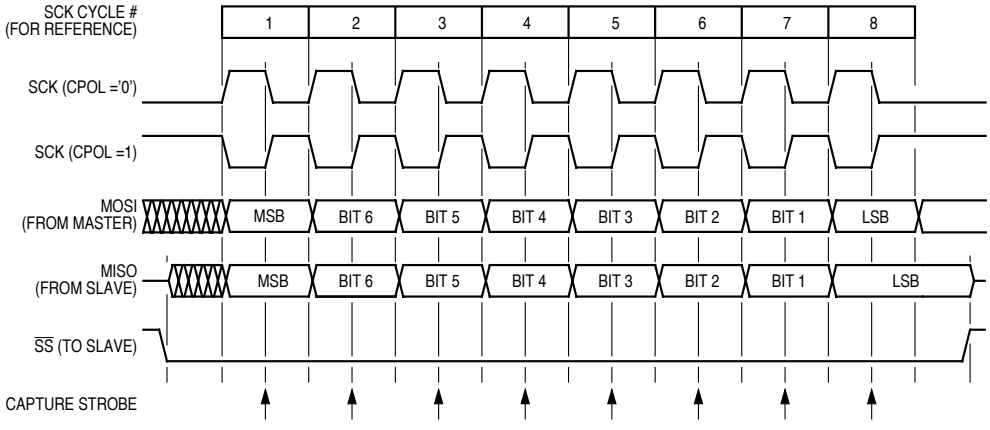


Figure 16-6 Transmission format (CPHA = ’1’)

16.5.4 Transmission initiation latency

When the SPI is configured as a master (SPMSTR = ‘1’), transmissions are started by a software write to the SPDR. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = ‘0’, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = ‘1’, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. See Figure 16-7. The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits are set to conserve power. SCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in Figure 16-7. This delay will be no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

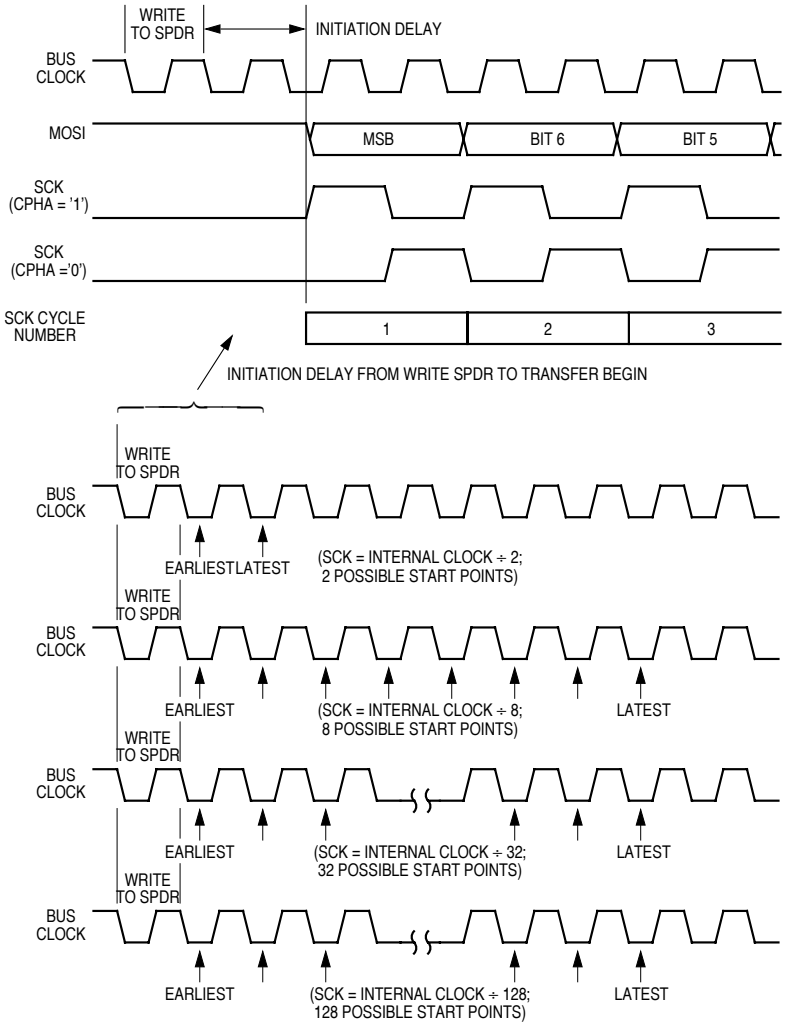


Figure 16-7 Transmission start delay (master)

16

16.6 Error conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — failing to read the SPI data register before the next byte enters the shift register results in the OVRF bit becoming set. The new byte does not transfer to the receive data register, and the unread byte still can be read by accessing the SPI data register. OVRF is in the SPI status and control register.
- Mode fault error (MODF) — the MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

16.6.1 Overflow error

The overflow flag (OVRF) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. See Figure 16-4 and Figure 16-6. If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request, and MODF and OVRF can generate a receiver/error CPU interrupt request. See Figure 16-11. It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

When the DMA is enabled to service the SPRF flag, it clears SPRF when it reads the SPI data register. The OVRF flag, however, still requires the two-step clearing mechanism of reading the flag when it is set and then reading the SPI data register. In this way, the DMA cannot directly clear the OVRF. However, if the CPU reads the SPI status and control register with the OVRF bit set, and then the DMA reads the SPI data register, the OVRF bit is cleared.

Conversely, if the CPU reads the data register to clear the OVRF flag, it could clear a pending SPRF service request to the DMA. Even if the DMA clears an SPRF, no new data will be transferred from the shift register to the data register with the OVRF high. This means that no new SPRF interrupts will be generated until this OVRF is cleared. For this reason, the OVRF interrupt to the CPU should be enabled when using the DMA to service the SPRF is there is any chance that the overflow condition might occur. See Figure 16-8.

The overflow service routine may need to disable the DMA and manually recover, since an overflow indicates the loss of data. Loss of data may prevent the DMA from reaching its byte count.

If an end-of-block transmission interrupt from the DMA was meant to pull the MCU out of WAIT, having an overflow condition without overflow interrupts enabled causes the MCU to hang in WAIT mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of WAIT mode instead.

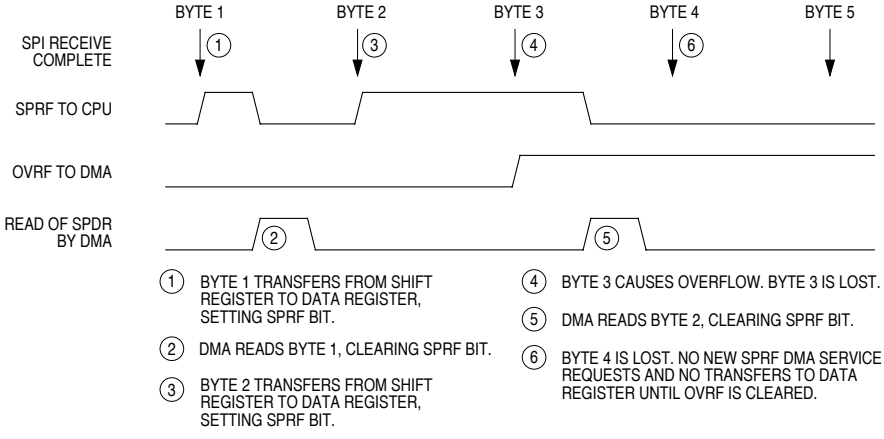


Figure 16-8 Overflow condition with DMA service of SPRF

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch??? for an overflow condition. Figure 16-9 shows how it is possible to miss an overflow.

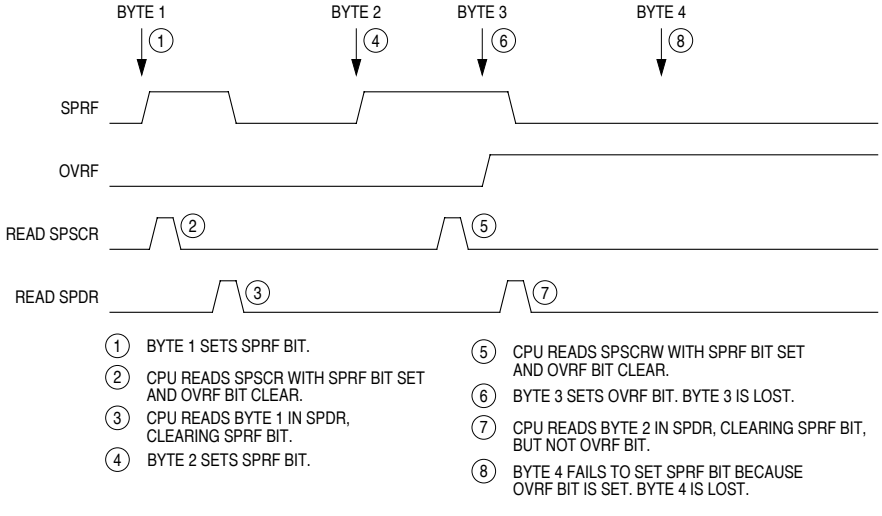


Figure 16-9 Missed read of overflow condition

The first part of Figure 16-9 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in the interval between SPSCR and SPDR being read.

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, the OVRF interrupt should be enabled, or alternatively another read of the SPSCR should be carried out following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will terminate with an SPRF interrupt. Figure 16-10 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

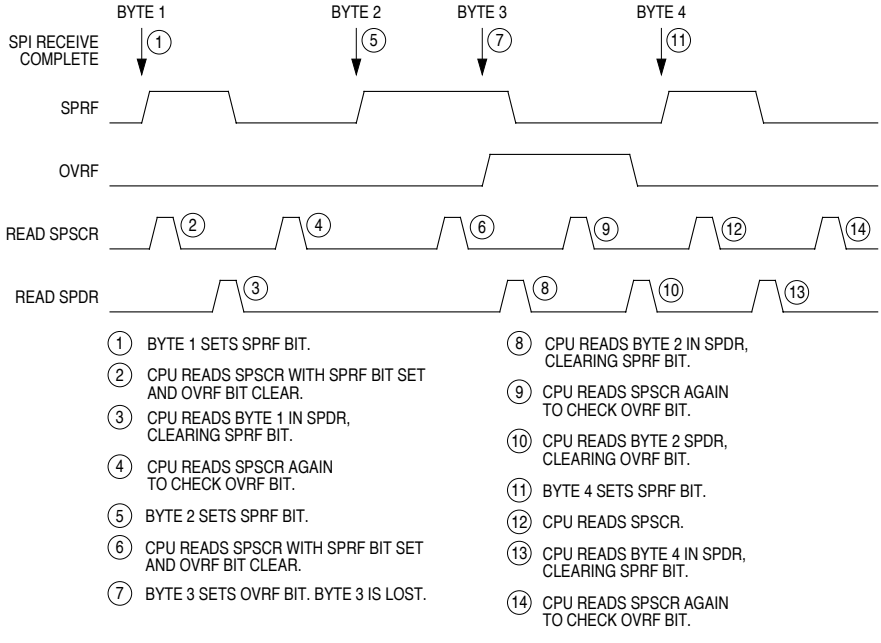


Figure 16-10 Clearing SPRF when OVRF interrupt is not enabled

16.6.2 Mode fault error

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request instead of a CPU interrupt request, but

MODF and OVRF can generate a receiver/error CPU interrupt request. See Figure 16-11. It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ becomes '0'. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = '1', the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE

To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port.

NOTE

Setting the MODF flag does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = '0'. Reading SPMSTR when MODF = '1' shows the difference between a MODF occurring when the SPI is a master and when it is a slave.

When configured as a slave (SPMSTR = '0'), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = '0', a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = '1', the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its IDLE level following the shift of the last data bit. See **16.5 Transmission formats**.

NOTE

When CPHA = ‘0’, a MODF occurs if a slave is selected ($\overline{SS}$ is at ‘0’) and later unselected ($\overline{SS}$ is ‘1’) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ at ‘0’ indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = ‘0’. When CPHA = ‘1’, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

In a slave SPI (MSTR = ‘0’), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by toggling the SPE bit of the slave.

NOTE

A ‘1’ on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

To clear the MODF flag, the SPSCR should be read with the MODF bit set and then the SPCR register should be written to. This entire clearing mechanism must occur with no MODF condition existing or else the flag will not be cleared.

16.7 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests or DMA service requests:

Table 16-3 SPI interrupts

Flag	Request
SPTE (Transmitter Empty)	SPI Transmitter CPU Interrupt Request (DMAS = 0, SPTIE = 1) SPI Transmitter DMA Service Request (DMAS = 1, SPTIE = 1)
SPRF (Receiver Full)	SPI Receiver CPU Interrupt Request (DMAS = 0, SPRIE = 1) SPI Receiver DMA Service Request (DMAS = 1, SPRIE = 1)
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1)
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = ’1’)

The DMA select bit (DMAS) controls whether SPTE and SPRF generate CPU interrupt requests or DMA service requests. When DMAS = 0, reading the SPI status and control register with SPRF set and then reading the SPI data register clears SPRF. When DMAS = 1, any read of the SPI data register clears the SPRF flag. The clearing mechanism for the SPTE flag is always just a write to the data register.

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests or transmitter DMA service requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests or receiver DMA service requests, provided that the SPI is enabled (SPE = 1).

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF flags to generate a receiver/error CPU interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF flag is enabled to generate receiver/error CPU interrupt requests.

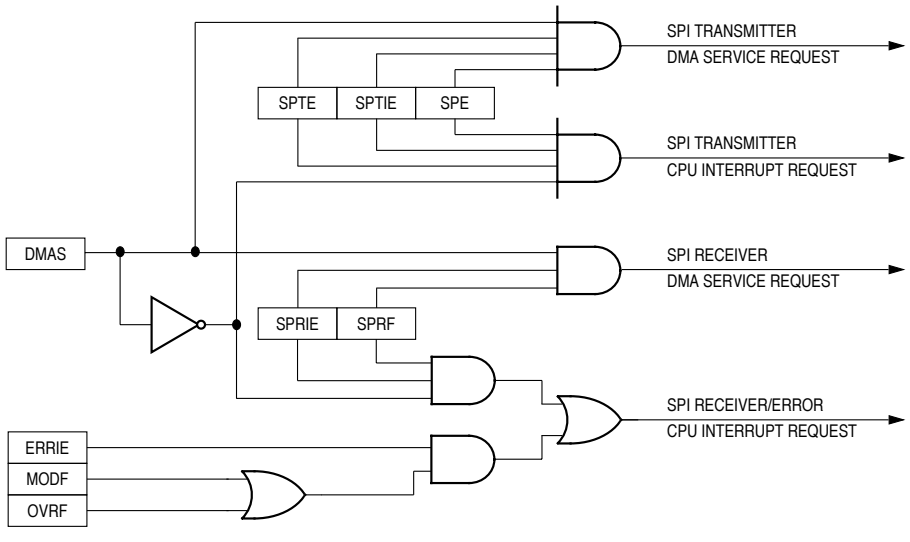


Figure 16-11 SPI interrupt request generation

The following sources in the SPI status and control register can generate CPU interrupt requests or DMA service requests:

- SPI receiver full bit (SPRF) — the SPRF bit becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF can generate either an SPI

receiver/error CPU interrupt request or an SPRF DMA service request.

If the DMA select bit, DMAS, is clear, SPRF generates an SPRF CPU interrupt request. If DMAS is set, SPRF generates an SPRF DMA service request.

- SPI transmitter empty (SPTE) — the SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE can generate either an SPTE CPU interrupt request or an SPTE DMA service request.

  If the DMAS bit is clear, SPTE generates an SPTE CPU interrupt request. If DMAS is set, SPTE generates an SPTE DMA service request.

16.8 Queuing transmission data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. Figure 16-12 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

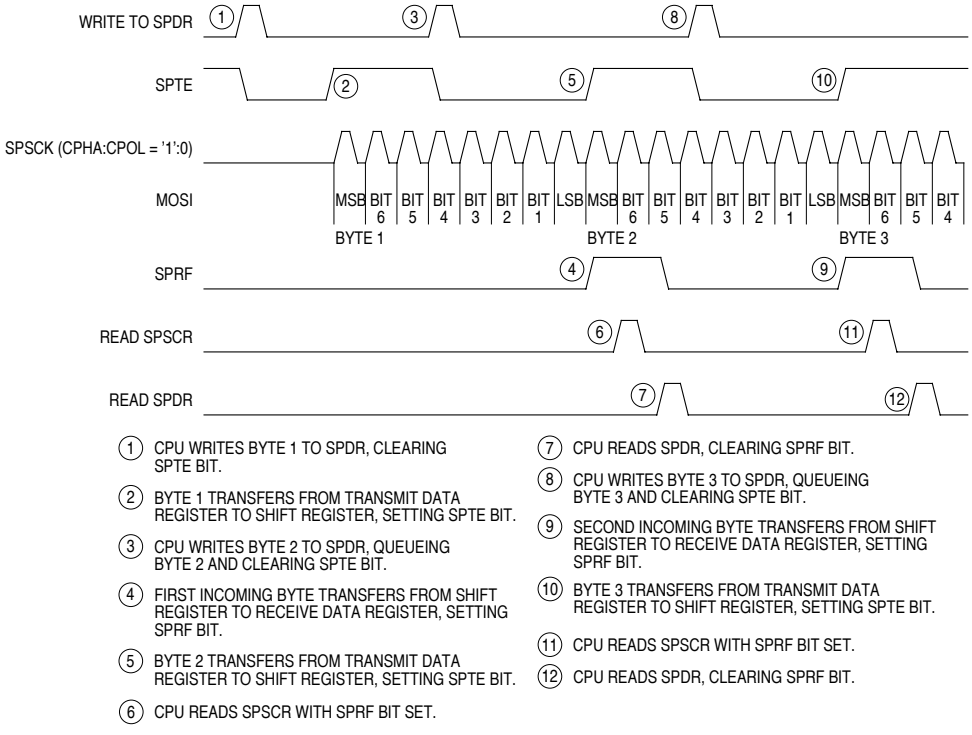


Figure 16-12 SPRF/SPTE CPU interrupt timing

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to time the write of its data between the transmissions. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.

16.9 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

- The SPTE flag is set
- Any transmission currently in progress is aborted
- The shift register is cleared
- The SPI state counter is cleared, making it ready for a new complete transmission
- All the SPI port logic is defaulted back to being general purpose I/O.

The following items are reset only by a system reset:

- All control bits in the SPCR register
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing ’0’ to the SPE bit. The SPI can also be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

16.10 Low-power modes

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

16.10.1 WAIT mode

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

If SPI module functions are not required during WAIT mode, power consumption can be reduced by disabling the SPI module before executing the WAIT instruction.

The DMA can service DMA service requests generated by the SPTE and SPRF flags without exiting WAIT mode. To exit WAIT mode when an overflow condition occurs, the OVRF bit should be enabled to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). See **16.7 Interrupts**.

16.10.2 STOP mode

The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after an external interrupt. If STOP mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

16.11 SPI during break interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **SECTION 7 SYSTEM INTEGRATION MODULE (SIM)**.

To allow software to clear status bits during a break interrupt, a ‘1’ should be written 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, a ‘0’ should be written to the BCFE bit. With BCFE at ‘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 two-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 a ‘0’. After the break, the second step clears the status bit.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the data register in break mode will not initiate a transmission, nor will this data be transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

16.12 I/O Signals

The SPI module has five I/O pins and shares four of them with a parallel I/O port.

- MISO — data received
- MOSI — data transmitted
- SPSCK — serial clock
- $\overline{SS}$ — slave select
- CGND — clock ground

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pullup resistor to V_{DD}.

16.12.1 MISO (Master in/Slave out)

MISO is one of the two SPI module pins that transmits serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is ‘0’ and its $\overline{SS}$ pin is at ‘0’. To support a multiple-slave system, a ‘1’ on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the MISO pin regardless of the state of the data direction register of the shared I/O port.

16.12.2 MOSI (Master out/Slave in)

MOSI is one of the two SPI module pins that transmits serial data. In full duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

When enabled, the SPI controls data direction of the MOSI pin regardless of the state of the data direction register of the shared I/O port.

16.12.3 SPSCK (serial clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

When enabled, the SPI controls data direction of the SPSCK pin regardless of the state of the data direction register of the shared I/O port.

16.12.4 $\overline{SS}$ (slave select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = ’0’, the $\overline{SS}$ is used to define the start of a transmission. See **16.5 Transmission formats**. Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = ‘0’ format. However, it can remain low throughout the transmission for the CPHA = ‘1’ format. See Figure 16-13.

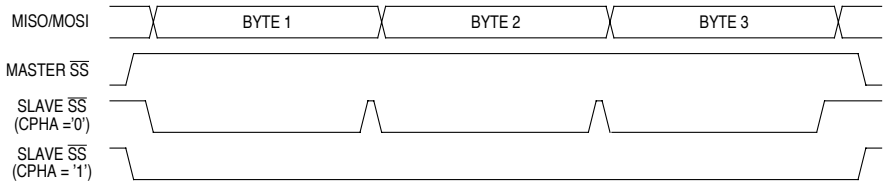


Figure 16-13 CPHA/$\overline{SS}$ timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. See **16.13.2 SPI status and control register (SPSCR)**.

NOTE

A ‘1’ on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. See **16.6.2 Mode fault error**. For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the data register. See Table 16-4.

Table 16-4 SPI configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF $\overline{SS}$ LOGIC
0	X(1)	X	Not Enabled	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

1. X = don’t care

16

16.12.5 CGND (clock ground)

CGND is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. To reduce the ground return path loop and minimize radio frequency (rf) emissions, the ground pin should be connected of the slave to the CGND pin.

16.13 I/O registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR)
- SPI status and control register (SPSCR)
- SPI data register (SPDR)

16.13.1 SPI control register (SPCR)

The SPI control register does the following:

- Enables SPI module interrupt requests
- Selects CPU interrupt requests or DMA service requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the SPSCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module

		Bit 7	6	5	4	3	2	1	Bit 0
SPCR	Read: Write:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
	Reset:	0	0	1	0	1	0	0	0

Figure 16-14 SPI control register (SPCR)

SPRIE — SPI receiver interrupt enable

This read/write bit enables CPU interrupt requests or DMA service requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit.

1 = SPRF CPU interrupt requests or SPRF DMA service requests enabled
0 = SPRF CPU interrupt requests or SPRF DMA service requests disabled

DMAS —DMA select

This read/write bit selects DMA service requests when the SPI receiver full bit, SPRF, or the SPI transmitter empty bit, SPTE, becomes set. Setting the DMAS bit disables SPRF CPU interrupt requests and SPTE CPU interrupt requests. Reset clears the DMAS bit.

1 = SPRF DMA and SPTE DMA service requests enabled
(SPRF CPU and SPTE CPU interrupt requests disabled)
0 = SPRF DMA and SPTE DMA service requests disabled
(SPRF CPU and SPTE CPU interrupt requests enabled)

SPMSTR — SPI master

This read/write bit selects master mode operation or slave mode operation. Reset sets the SPMSTR bit.

1 = Master mode
0 = Slave mode

CPOL — Clock polarity

This read/write bit determines the logic state of the SPSCK pin between transmissions. See Figure 16-4 and Figure 16-6. To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

CPHA — Clock phase

This read/write bit controls the timing relationship between the serial clock and SPI data. See Figure 16-4 and Figure 16-6. To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = ’0’, the $\overline{SS}$ pin of the slave SPI module must be set to logic one between bytes. See Figure 16-13. Reset sets the CPHA bit.

When CPHA =’0’ for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the data register. Therefore, the slave data register must be loaded with the desired transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = ’1’ for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. See **16.6.2 Mode fault error**. A ‘1’ on the $\overline{SS}$ pin does not affect the state of the SPI state machine in any way.

SPWOM — SPI wired-OR mode

This read/write bit disables the pull-up devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

1 = Wired-OR SPSCK, MOSI, and MISO pins
0 = Normal push-pull SPSCK, MOSI, and MISO pins

16

SPE — SPI enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. See **16.9 Resetting the SPI**. Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI transmit interrupt enable

This read/write bit enables CPU interrupt requests or DMA service requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit.

1 = SPTE CPU interrupt requests or SPTE DMA service requests enabled
0 = SPTE CPU interrupt requests or SPTE DMA service requests disabled

16.13.2 SPI status and control register (SPSCR)

The SPI status and control register contains flags to signal the following conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform the following functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

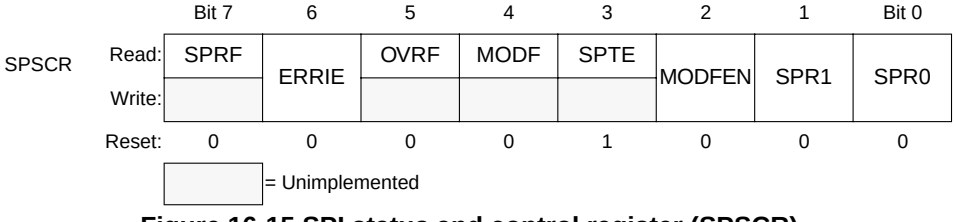


Figure 16-15 SPI status and control register (SPSCR)

SPRF — SPI receiver full

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request or a DMA service request if the SPRIE bit in the SPI control register is set also.

The DMA select bit (DMAS) in the SPI control register determines whether SPRF generates an SPRF CPU interrupt request or an SPRF DMA service request. During an SPRF CPU interrupt (DMAS = ‘0’), the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. During an SPRF DMA transmission (DMAS = ‘1’), any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.
1 = Receive data register full
0 = Receive data register not full

NOTE

When the DMA is configured to service the SPI (DMAS = ‘1’), a read by the CPU of the SPI data register can inadvertently clear the SPRF bit and cause the DMA to miss a service request.

ERRIE — Error interrupt enable

This read-only bit enables the MODF and OVRF flags to generate CPU interrupt requests. Reset clears the ERRIE bit.

1 = MODF and OVRF can generate CPU interrupt requests
0 = MODF and OVRF cannot generate CPU interrupt requests

OVRF — Overflow flag

This clearable, read-only flag is set if software does not read the byte in the receive data register before the next byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the SPI data register. Reset clears the OVRF flag.

1 = Overflow
0 = No overflow

MODF — Mode fault

This clearable, ready-only flag is set in a slave SPI if the $\overline{SS}$ pin goes high during a transmission. In a master SPI, the MODF flag is set if the $\overline{SS}$ pin goes low at any time. Clear the MODF bit by reading the SPI status and control register with MODF set and then writing to the SPI data register. Reset clears the MODF bit.

1 = $\overline{SS}$ pin at inappropriate logic level
0 = $\overline{SS}$ pin at appropriate logic level

SPTE — SPI transmitter empty

This clearable, read-only flag is set each time the transmit data register transfers a byte into the shift register. SPTE generates an SPTE CPU interrupt request or an SPTE DMA service request if the SPTIE bit in the SPI control register is set also.

NOTE

The SPI data register should not be written to unless the SPTE bit is high.

The DMA select bit (DMAS) in the SPI control register determines whether SPTE generates an SPTE CPU interrupt request or an SPTE DMA service request. During an SPTE CPU interrupt (DMAS = ‘0’), the CPU clears the SPTE bit by writing to the SPI data register. During an SPTE DMA transmission (DMAS = ‘1’), the DMA automatically clears SPTE when it writes to the SPI data register.

NOTE

When the DMA is configured to service the SPI (DMAS = ‘1’), a write by the CPU of the SPI data register can inadvertently clear the SPTE bit and cause the DMA to miss a service request.

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE will be set again within two bus cycles since the transmit buffer empties into the shift register. This allows the user to queue up a 16-bit value to send. For an already active slave, the load of the shift register cannot occur until the transmission is completed. This implies that a back-to-back write to the transmit data register is not possible. The SPTE indicates when the next write can occur.

Reset sets the SPTE bit.

1 = Transmit data register empty
0 = Transmit data register not empty

MODFEN — Mode fault enable

This read/write bit, when set to ’1’, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ pin is available as a general purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general purpose I/O regardless of the value of MODFEN. See **16.12.4 SS (slave select)**.

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. See **16.6.2 Mode fault error**.

SPR1 and SPR0 — SPI baud rate select

In master mode, these read/write bits select one of four baud rates as shown in Table 16-5. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 16-5 SPI master baud rate selection

SPR1:SPR0	Baud rate divisor (BD)
00	2
01	8
10	32
11	128

The following formula is used to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

where:

CGMOUT = base clock output of the clock generator module (CGM)
BD = baud rate divisor

16.13.3 SPI data register (SPDR)

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See Figure 16-2.

		Bit 7	6	5	4	3	2	1	Bit 0
SPDR	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Indeterminate after reset							

Figure 16-16 SPI data register (SPDR)

R7:R0/T7:T0 — Receive/Transmit data bits

NOTE

Read-modify-write instructions should not be used on the SPI data register since the buffer read is not the same as the buffer written.

SECTION 17
TIMER INTERFACE MODULE A (TIMA)

17.1 Introduction

This section describes the timer interface module (TIM4). The TIMA is a four-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 17-1 is a block diagram of the TIMA.

17.2 Features

Features of the TIMA include the following:

- Four 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 TIMA clock input
  - Seven-frequency internal bus clock prescaler selection
  - External TIMA clock input (4MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits
- Modular architecture expandable to 8 channels

17

17.3 Functional description

Figure 17-1 shows the structure of the TIMA. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH:TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

The four TIMA channels are programmable independently as input capture or output compare channels.

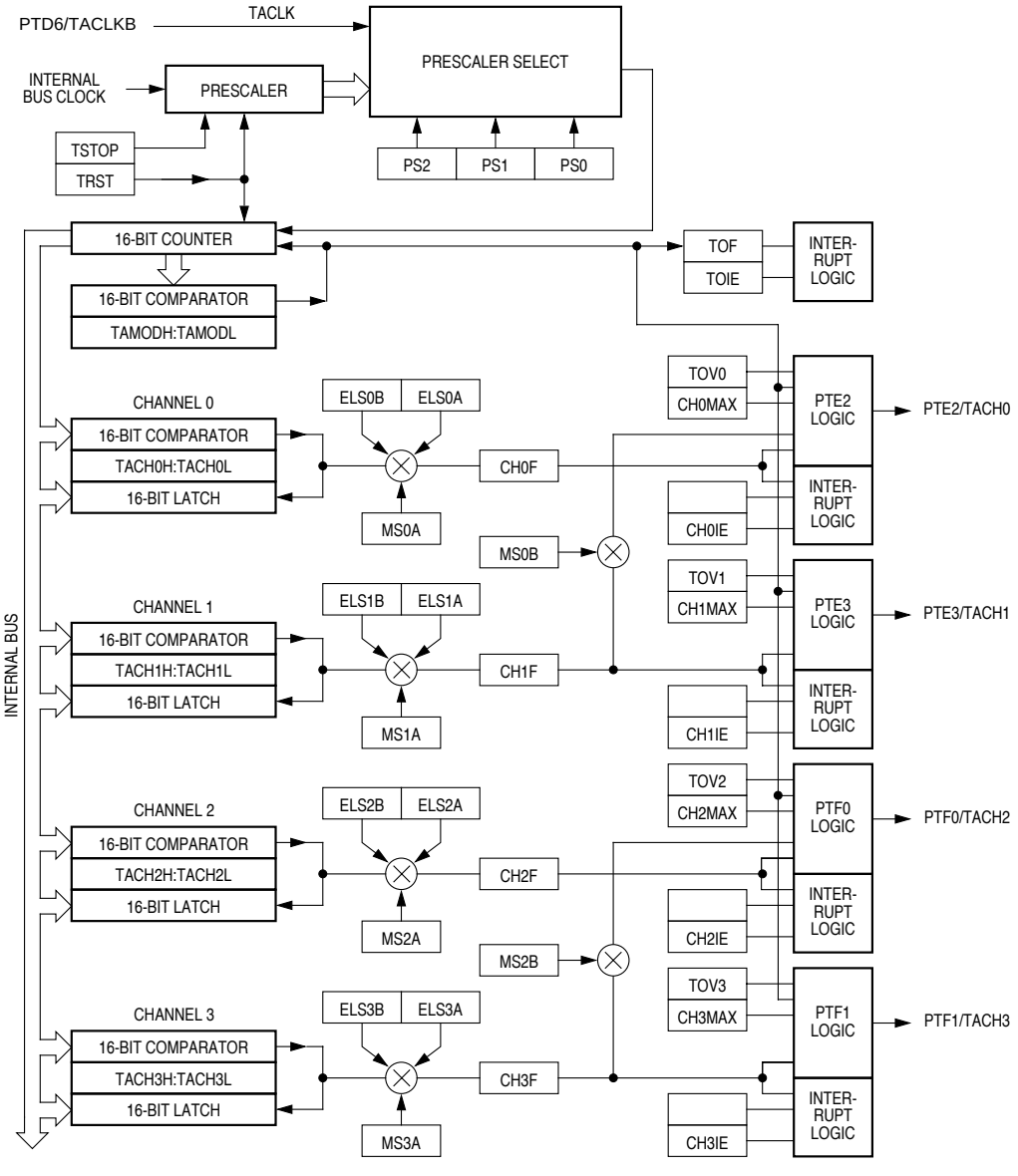


Figure 17-1 TIMA block diagram

Table 17-1 TIMA I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
TIMA status/control register (TASC)	TOF	TOIE	TSTOP	TRST	0	PS2	PS1	PS0	$0020
TIMA counter register high (TACNTH)	Bit 15	14	13	12	11	10	9	Bit 8	$0022
TIMA counter register low (TACNTL)	Bit 7	6	5	4	3	2	1	Bit 0	$0023
TIMA Counter modulo reg. high (TAMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$0024
TIMA counter modulo reg. low (TAMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$0025
TIMA Ch. 0 Status/control register (TASC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	$0026
TIMA Ch. 0 register high (TACH0H)	Bit 15	14	13	12	11	10	9	Bit 8	$0027
TIMA Ch. 0 register low (TACH0L)	Bit 7	6	5	4	3	2	1	Bit 0	$0028
TIMA Ch. 1 status/control register (TASC1)	CH1F	CH1IE		MS1A	ELS1B	ELS1A	TOV1	CH1MAX	$0029
TIMA Ch. 1 register high (TACH1H)	Bit 15	14	13	12	11	10	9	Bit 8	$002A
TIMA Ch. 1 register Low (TACH1L)	Bit 7	6	5	4	3	2	1	Bit 0	$002B
TIMA Ch. 2 status/control register (TASC2)	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX	$002C
TIMA Ch. 2 register High (TACH2H)	Bit 15	14	13	12	11	10	9	Bit 8	$002D
TIMA Ch. 2 register Low (TACH2L)	Bit 7	6	5	4	3	2	1	Bit 0	$002E
TIMA Ch. 3 status/control register (TASC3)	CH3F	CH3IE		MS3A	ELS3B	ELS3A	TOV3	CH3MAX	$002F
TIMA Ch. 3 register High (TACH3H)	Bit 15	14	13	12	11	10	9	Bit 8	$0030
TIMA Ch. 3 register Low (TACH3L)	Bit 7	6	5	4	3	2	1	Bit 0	$0031

= Unimplemented

17.3.1 TIMA counter prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/TACLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

17.3.2 Input capture

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

17.3.3 Output compare

With the output compare function, the TIMA 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 TIMA can set, clear, or toggle the channel pin. Output compares can generate TIM CPU interrupt requests.

17.3.3.1 Unbuffered output compare

Any output compare channel can generate unbuffered output compare pulses as described in **17.3.3 Output compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

An unsynchronized write to the TIMA 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 TIMA overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMA 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA 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.

17.3.3.2 Buffered output compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TASC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The output compare value in the TIMA channel 2 registers initially controls the output on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the output are the ones written to last. TASC2 controls and monitors the buffered output compare function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, 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.

17.3.4 Pulse width modulation (PWM)

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

As Figure 17-2 shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIMA to set the pin if the state of the PWM pulse is logic zero.

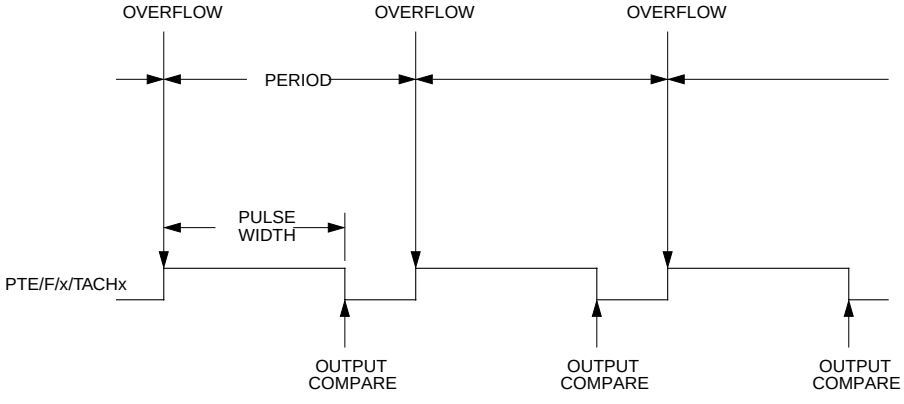


Figure 17-2 PWM period and pulse width

The value in the TIMA 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 TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000. See **17.8.1 TIMA status and control register (TASC)**.

The value in the TIMA 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 $0080 (128) to the TIMA channel registers produces a duty cycle of 128/256 or 50%.

17

17.3.4.1 Unbuffered PWM signal generation

Any output compare channel can generate unbuffered PWM pulses as described in **17.3.4 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 TIMA channel registers.

An unsynchronized write to the TIMA 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 TIMA overflow interrupt routine to write a new, smaller pulse width value may

cause the compare to be missed. The TIMA 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA 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.

17.3.4.2 Buffered PWM signal generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The TIMA channel 2 registers initially control the pulse width on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the pulse width are the ones written to last. TASC2 controls and monitors the buffered PWM function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, 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.

17.3.4.3 PWM initialization

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

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH:TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH:TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   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 17-3.
   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 17-3.

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 TIMA status control register (TASC), clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H:TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIMA channel 2 registers (TACH2H:TACH2L) initially control the PWM output. TIMA status control register 2 (TASCR2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIMA 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 **17.8.4 TIMA channel status and control registers (TASC0–TASC3)**.

17.4 Interrupts

The following TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIMA counter value rolls over to $0000 after matching the value in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH3F–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 TIMA channel x status and control register.

17.5 Low-power modes

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

17.5.1 Wait mode

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

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

17.6 TIMA during break interrupts

A break interrupt stops the TIMA 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **7.7.3 SIM break flag control register (SBFCR)**.

To allow software to clear status bits during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-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 zero. After the break, doing the second step clears the status bit.

17.7 I/O Signals

Ports E and F each share two pins with the TIM and Port D shares one. PTD6/TACLK is an external clock input to the TIMA prescaler. The four TIMA channel I/O pins are PTE2/TACH0, PTE3/TACH1, PTF0/TACH2, and PTF1/TACH3.

17.7.1 TIMA clock pin (PTD6/TACLK)

PTD6/TACLK is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTD6/TACLK input by writing logic ones to the three prescaler select bits, PS[2:0]. See **17.8.1 TIMA**

status and control register (TASC). The minimum TACLK pulse width, $TACLK_{LMIN}$ or $TACLK_{HMIN}$, is:

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

The maximum TCLK frequency is:

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

PTD6/TACLK is available as a general-purpose I/O pin when not used as the TIMA clock input. When the PTD6/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

17.7.2 TIMA channel I/O pins (PTF1/TACH3-PTE2/TACH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF0/TACH2 and PTE3/TACH1 can be configured as buffered output compare or buffered PWM pins.

17.8 I/O registers

The following I/O registers control and monitor operation of the TIMA:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH:TACNTL)
- TIMA counter modulo registers (TAMODH:TAMODL)
- TIMA channel status and control registers (TASC0, TASC1, TASC2, and TASC3)
- TIMA channel registers (TACH0H:TACH0L, TACH1H:TACH1L, TACH2H:TACH2L, and TACH3H:TACH3L)

17.8.1 TIMA status and control register (TASC)

The TIMA status and control register does the following:

- Enables TIMA overflow interrupts
- Flags TIMA overflows
- Stops the TIMA counter
- Resets the TIMA counter

- Prescales the TIMA counter clock

		Bit 7	6	5	4	3	2	1	Bit 0
TASC $0020	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
	Write:	0			TRST				
	Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 17-3 TIMA status and control register (TASC)

TOF — TIMA Overflow Flag Bit

This read/write flag is set when the TIMA counter resets to $0000 after reaching the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic zero to TOF. If another TIMA overflow occurs before the clearing sequence is complete, then writing logic zero 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 one to TOF has no effect.

1 = TIMA counter has reached modulo value
0 = TIMA counter has not reached modulo value

TOIE — TIMA Overflow Interrupt Enable Bit

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

1 = TIMA overflow interrupts enabled
0 = TIMA overflow interrupts disabled

TSTOP — TIMA Stop Bit

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

1 = TIMA counter stopped
0 = TIMA counter active

17

NOTE

Do not set the TSTOP bit before entering wait mode if the TIMA is required to exit wait mode.

TRST — TIMA Reset Bit

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

1 = Prescaler and TIMA counter cleared
0 = No effect

NOTE

Setting the TSTOP and TRST bits simultaneously stops the TIMA counter at a value of $0000.

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/TACLK pin or one of the seven prescaler outputs as the input to the TIMA counter as Table 17-2 shows. Reset clears the PS[2:0] bits.

Table 17-2 Prescaler selection

PS[2:0]	TIMA 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	PTD6/TACLK

17.8.2 TIMA counter registers (TACNTH:TACNTL)

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

NOTE

If you read TACNTH during a break interrupt, be sure to unlatch TACNTL by reading TACNTL before exiting the break interrupt. Otherwise, TACNTL retains the value latched during the break.

		Bit 7	6	5	4	3	2	1	Bit 0
TACNTH $0022	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	0	0	0	0	0	0	0	0

		Bit 7	6	5	4	3	2	1	Bit 0
TACNTL $0023	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-4 TIMA counter registers (TACNTH:TACNTL)

17.8.3 TIMA counter modulo registers (TAMODH:TAMODL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
TAMODH $0024	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	1	1	1	1	1	1	1	1

		Bit 7	6	5	4	3	2	1	Bit 0
TAMODL $0025	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	1	1	1	1	1	1	1	1

Figure 17-5 TIMA counter modulo registers (TAMODH:TAMODL)

NOTE

Reset the TIMA counter before writing to the TIMA counter modulo registers.

17.8.4 TIMA channel status and control registers (TASC0–TASC3)

Each of the TIMA channel status and control registers does the following:

- 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 TIMA overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

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 TIMA counter registers matches the value in the TIMA channel x registers.

When TIM CPU interrupt requests are enabled (CHxIE = 1), clear CHxF by reading TIMA channel x status and control register with CHxF set and then writing a logic zero to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic zero 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 one 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 TIMA CPU interrupts 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

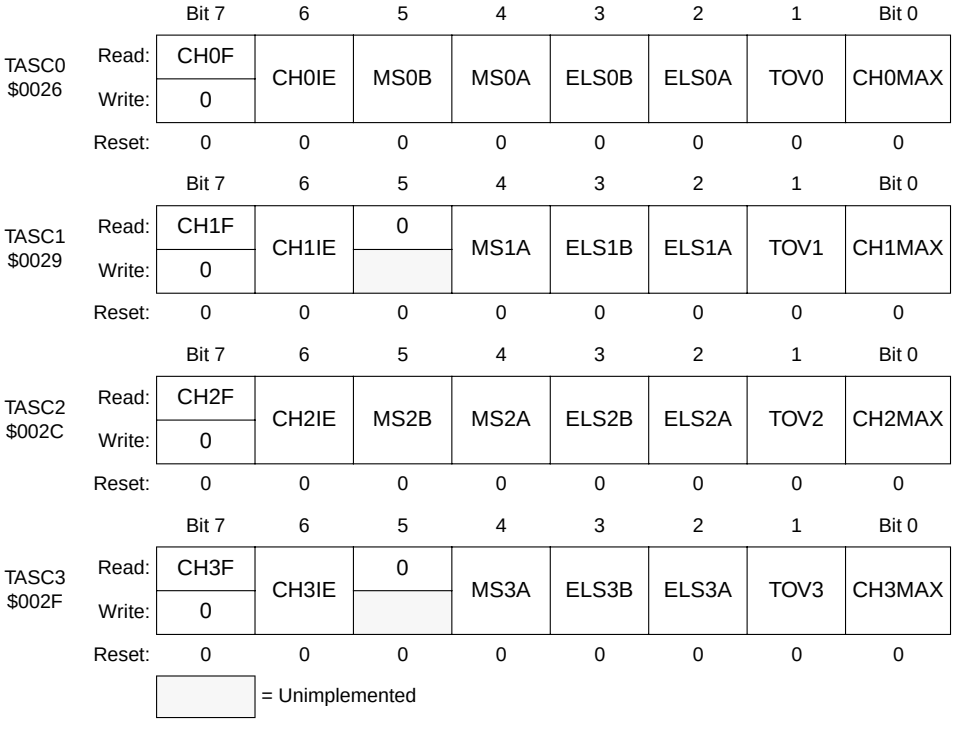


Figure 17-6 TIMA channel status and control registers (TASC0–TASC3)

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMA channel 0 and TIMA channel 2 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts TCH1B to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TCH3B 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 17-3.

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 TBCHx pin. See Table 17-3. 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 TIMA status and control register (TASC).

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 PTEx/TBCHx is available as a general-purpose I/O pin. Table 17-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 17-3 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 TIMA channel register for input capture operation, make sure that the PTE/TCHxB pin is stable for at least two bus clocks.

TOVx — 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 TIMA counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on TIMA counter overflow.
0 = Channel x pin does not toggle on TIMA counter overflow.

NOTE

When TOVx is set, a TIMA counter overflow takes precedence over a channel x output compare if both occur at the same time.

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 17-7 shows, the CHxMAX 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 CHxMAX is cleared.

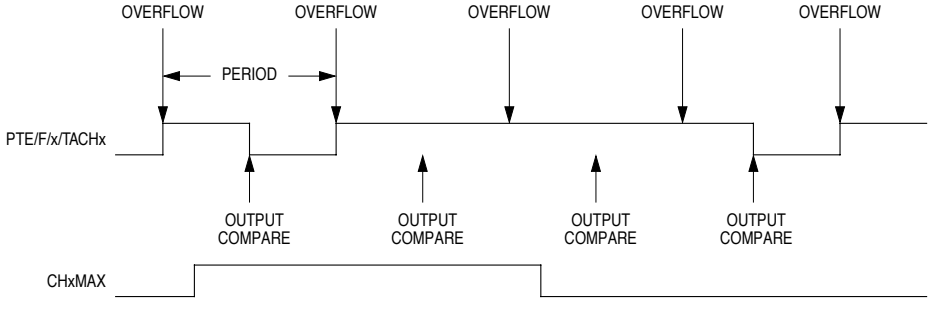


Figure 17-7 CHxMAX Latency

17.8.5 TIMA channel registers (TACH0H/L–TACH3H/L)

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

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIMA channel x registers (TACHxH) inhibits input captures until the low byte (TACHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIMA channel x registers (TACHxH) inhibits output compares until the low byte (TACHxL) is written.

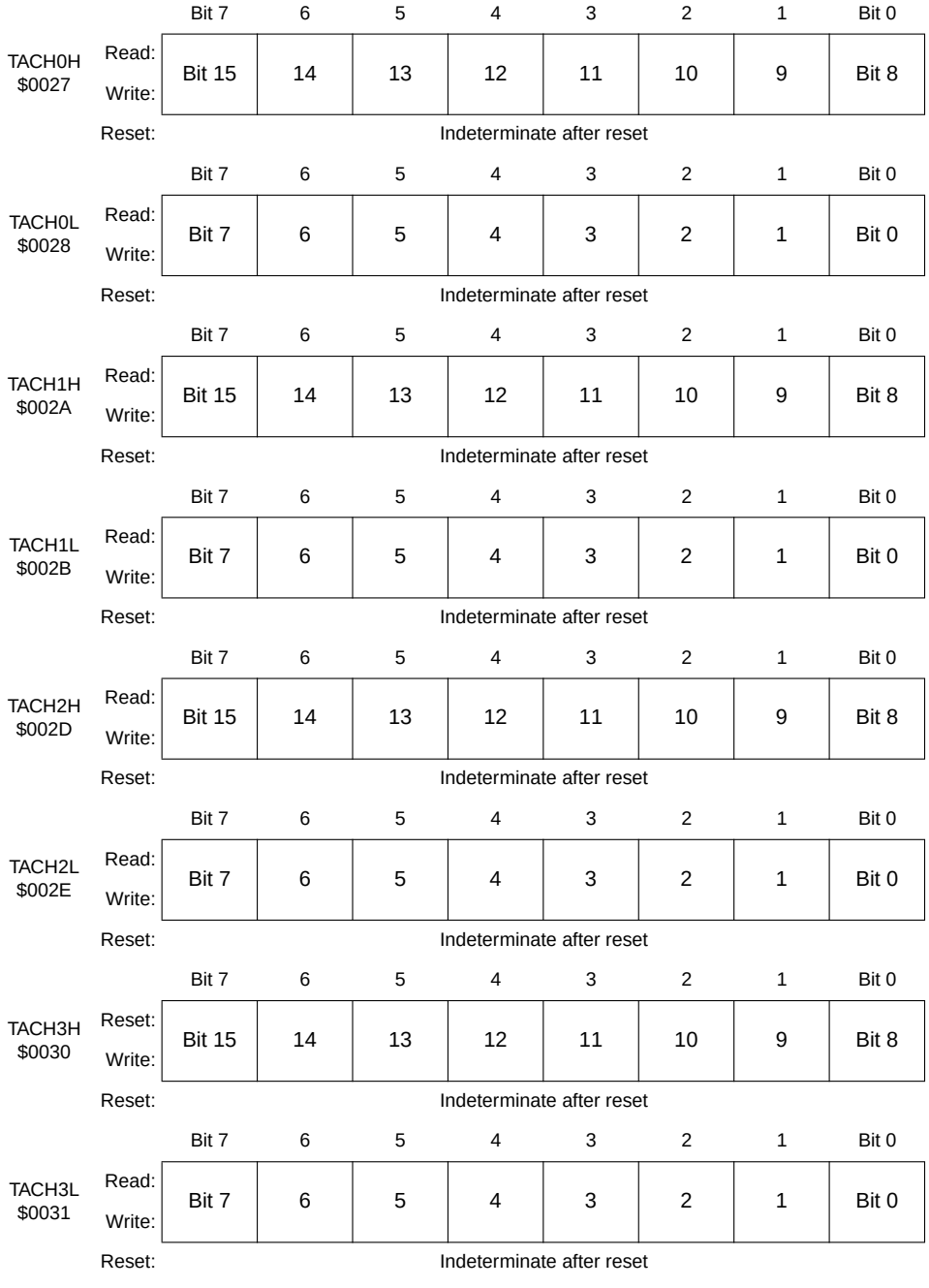


Figure 17-8 TIMA channel registers (TACH0H/L–TACH3H/L)

SECTION 18
TIMER INTERFACE MODULE B (TIMB)

18.1 Introduction

This section describes the timer interface module (TIM2). The TIMB is a two-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 18-1 is a block diagram of the TIMB.

18.2 Features

Features of the TIMB include the following:

- 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 TIMB clock input
  - Seven-frequency internal bus clock prescaler selection
  - External TIMB Clock Input (4-MHz Maximum Frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits
- Modular architecture expandable to 8 channels

18.3 Functional description

Figure 18-1 shows the structure of the TIMB. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH:TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

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

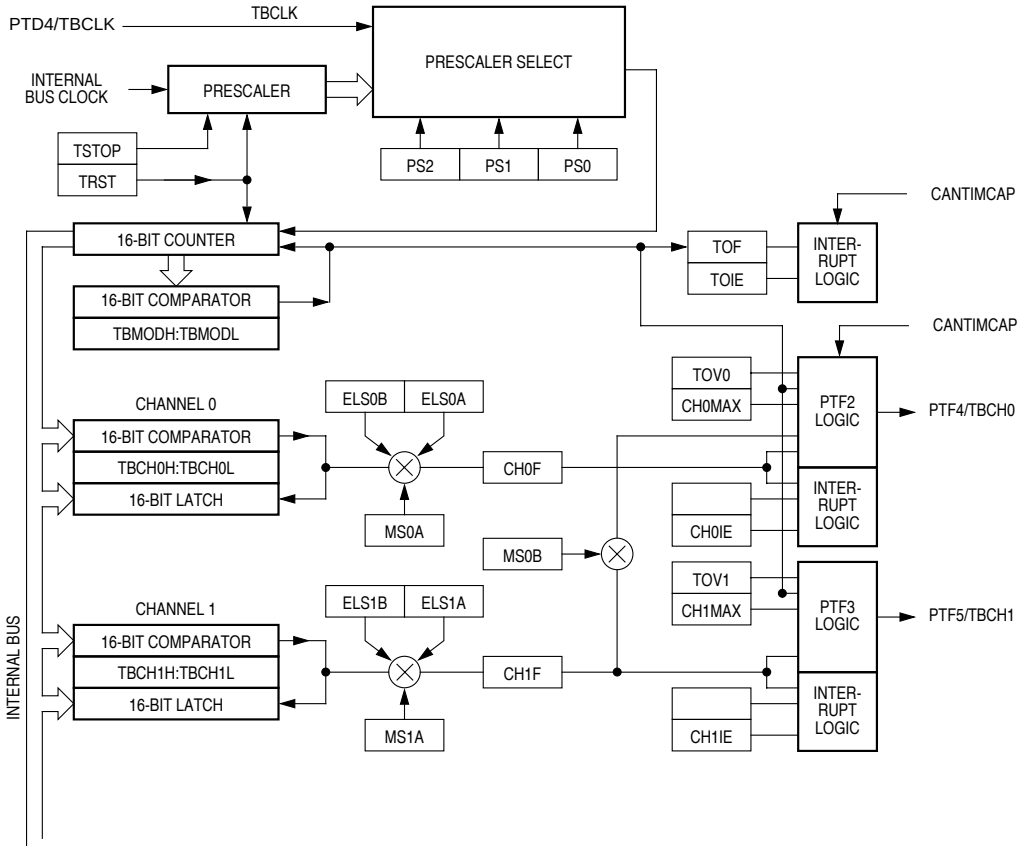


Figure 18-1 TIMB Block Diagram

18

18.3.1 TIMB counter prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTD4/TBCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

Table 18-1 TIMB I/O register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
TIMB Status/Control Register (TBSC)	TOF	TOIE	TSTOP	TRST	0	PS2	PS1	PS0	$0040
TIMB Counter Register High (TBCNTH)	Bit 15	14	13	12	11	10	9	Bit 8	$0041
TIMB Counter Register Low (TBCNTL)	Bit 7	6	5	4	3	2	1	Bit 0	$0042
TIMB Counter Modulo Reg. High (TBMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$0043
TIMB Counter Modulo Reg. Low (TBMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$0044
TIMB Ch. 0 Status/Control Register (TBSC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	$0045
TIMB Ch. 0 Register High (TBCH0H)	Bit 15	14	13	12	11	10	9	Bit 8	$0046
TIMB Ch. 0 Register Low (TBCH0L)	Bit 7	6	5	4	3	2	1	Bit 0	$0047
TIMB Ch. 1 Status/Control Register (TBSC1)	CH1F	CH1IE		MS1A	ELS1B	ELS1A	TOV1	CH1MAX	$0048
TIMB Ch. 1 Register High (TBCH1H)	Bit 15	14	13	12	11	10	9	Bit 8	$0049
TIMB Ch. 1 Register Low (TBCH1L)	Bit 7	6	5	4	3	2	1	Bit 0	$004A

= Unimplemented

18.3.2 Input capture

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

18.3.3 Output compare

With the output compare function, the TIMB 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 TIMB can set, clear, or toggle the channel pin. Output compares can generate TIM CPU interrupt requests.

18.3.3.1 Unbuffered output compare

Any output compare channel can generate unbuffered output compare pulses as described in **18.3.3 Output compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

An unsynchronized write to the TIMB 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 TIMB overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMB 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB 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.

18.3.3.2 Buffered output compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TBSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, 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.

18.3.4 Pulse Width Modulation (PWM)

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

As Figure 18-2 shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIMB to set the pin if the state of the PWM pulse is logic zero.

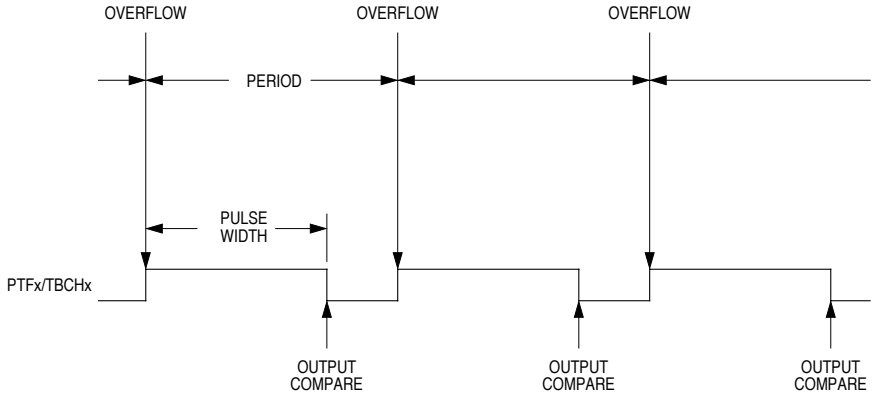


Figure 18-2 PWM period and pulse width

The value in the TIMB 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 TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000. See **18.8.1 TIMB status and control register (TBSC)**.

The value in the TIMB 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 $0080 (128) to the TIMB channel registers produces a duty cycle of 128/256 or 50%.

18.3.4.1 Unbuffered PWM signal generation

Any output compare channel can generate unbuffered PWM pulses as described in **18.3.4 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 TIMB channel registers.

An unsynchronized write to the TIMB 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 TIMB overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMB 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB 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.

18.3.4.2 Buffered PWM signal generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, 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.

18.3.4.3 PWM initialization

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

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH:TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH:TBCHxL), write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   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 18-3.
   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 18-3.)

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 TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H:TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSCR0) 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 TIMB 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 **18.8.4 TIMB channel status and control registers (TBSC0–TBSC1)**.

18.4 Interrupts

The following TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter value rolls over to $0000 after matching the value in the TIMB counter modulo registers. The TIMB overflow interrupt enable bit, TOIE, enables TIMB overflow CPU interrupt requests. TOF and TOIE are in the TIMB status and control register.
- TIMB 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 TIMB channel x status and control register.

18.5 Low-power modes

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

18.5.1 WAIT mode

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

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

18.6 TIMB during break interrupts

A break interrupt stops the TIMB 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **7.7.3 SIM break flag control register (SBFCR)**.

To allow software to clear status bits during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (its default state), software can read and write I/O registers

during the break state without affecting status bits. Some status bits have a two-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 zero. After the break, doing the second step clears the status bit.

18.7 I/O Signals

Port F Shares two of its pins with the TIMB and Port D shares one. PTD4/TBCLK is an external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTF4/TBCH0 and PTF5/TBCH1.

18.7.1 TIMB clock Pin (PTD4/TBLCK)

PTD4/TBCLK is an external clock input that can be the clock source for the TIMB counter instead of the prescaled internal bus clock. Select the PTD4/TBCLK input by writing logic ones to the three prescaler select bits, PS[2:0]. (See **18.8.1 TIMB status and control register (TBSC)**.) The minimum TBCLK pulse width, $TBCLK_{LMIN}$ or $TBCLK_{HMIN}$, is:

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

The maximum TCLK frequency is:

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

is available as a general-purpose I/O pin when not used as the TIMB clock input. When the PTD4/TBCLK pin is the TIMB clock input, it is an input regardless of the state of the DDR5 bit in data direction register D.

18.7.2 TIMB channel I/O pins (PTF5/TBCH1–PTF4/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF5/TBCH1 and PTF4/TBCH0 can be configured as buffered output compare or buffered PWM pins.

TBCH0 has an additional source for the input capture signal i.e CANTIMCAP. See Figure 18-1.

This signal is generated by the msCAN08 which generates a timer signal whenever a valid frame has been received or transmitted. The signal is routed into TBCH0

under the control of the Timer Link Enable (TLNKEN) bit in the CMCR0 see 23.12.2 msCAN08 module control register (CMCR0).

18.8 I/O Registers

The following I/O registers control and monitor operation of the TIM:

- TIMB status and control register (TBSC)
- TIMB control registers (TBCNTH:TBCNTL)
- TIMB counter modulo registers (TBMODH:TBMODL)
- TIMB channel status and control registers (TBSC0 and TBSC1)
- TIMB channel registers (TBCH0H:TBCH0L and TBCH1H:TBCH1L)

18.8.1 TIMB status and control register (TBSC)

The TIMB status and control register does the following:

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

		Bit 7	6	5	4	3	2	1	Bit 0
TBSC $0040	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
	Write:	0			TRST				
	Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 18-3 TIMB status and control register (TBSC)

18

TOF — TIMB overflow flag bit

This read/write flag is set when the TIMB counter resets to $0000 after reaching the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic zero to TOF. If another TIMB overflow occurs before the clearing sequence is complete, then writing logic zero 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 one to TOF has no effect.

1 = TIMB counter has reached modulo value
0 = TIMB counter has not reached modulo value

TOIE — TIMB overflow interrupt enable bit

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

1 = TIMB overflow interrupts enabled
0 = TIMB overflow interrupts disabled

TSTOP — TIMB stop bit

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

1 = TIMB counter stopped
0 = TIMB counter active

NOTE

Do not set the TSTOP bit before entering wait mode if the TIMB is required to exit wait mode.

TRST — TIMB reset bit

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

1 = Prescaler and TIMB counter cleared
0 = No effect

NOTE

Setting the TSTOP and TRST bits simultaneously stops the TIMB counter at a value of $0000.

PS[2:0] — Prescaler select bits

These read/write bits select either the PTD5 pin or one of the seven prescaler outputs as the input to the TIMB counter as Table 18-2 shows. Reset clears the PS[2:0] bits.

18.8.2 TIMB counter registers (TBCNTH:TBCNTL)

The two read-only TIMB counter registers contain the high and low bytes of the value in the TIMB counter. Reading the high byte (TBCNTH) latches the contents

Table 18-2 Prescaler selection

PS[2:0]	TIMB 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	PTD4/TBLCK

of the low byte (TBCNTL) into a buffer. Subsequent reads of TBCNTH do not affect the latched TBCNTL value until TBCNTL is read. Reset clears the TIMB counter registers. Setting the TIMB reset bit (TRST) also clears the TIMB counter registers

---

NOTE

If you read TBCNTH during a break interrupt, be sure to unlatch TBCNTL by reading TBCNTL before exiting the break interrupt. Otherwise, TBCNTL redbones the value latched during the break.

		Bit 7	6	5	4	3	2	1	Bit 0
TBCNTH $0041	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
TBCNTL $0042	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-4 TIMB counter registers (TBCNTH:TBCNTL)

18.8.3 TIMB counter modulo registers (TBMODH:TBMODL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
TBMODH $0043	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	1	1	1	1	1	1	1	1
		Bit 7	6	5	4	3	2	1	Bit 0
TBMODL $0044	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	1	1	1	1	1	1	1	1

Figure 18-5 TIMB counter modulo registers (TBMODH:TBMODL)

NOTE

Reset the TIMB counter before writing to the TIMB counter modulo registers.

18.8.4 TIMB channel status and control registers (TBSC0–TBSC1)

Each of the TIMB channel status and control registers does the following:

- 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 TIMB overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

		Bit 7	6	5	4	3	2	1	Bit 0
TBSC0 $0045	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
TBSC1 $0048	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-6 TIMB channel status and control registers (TBSC0–TBSC1)

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 TIMB counter registers matches the value in the TIMB channel x registers.

When TIMB CPU interrupt requests are enabled (CHxIE=1), clear CHxF by reading TIMB channel x status and control register with CHxF set and then writing a logic zero to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic zero 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 one 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 TIMB 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 TIMB 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 18-3.

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 18-3. 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 TIMB 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 F, and pin PTFx/TBCHx is available as a general-purpose I/O pin. Table 18-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 18-3 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

Table 18-3 Mode, edge, and level selection

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
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 TIMB channel register for input capture operation, make sure that the PTFx/TBCHx pin is stable for at least two bus clocks.

TOVx — 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 TIMB counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on TIMB counter overflow.
0 = Channel x pin does not toggle on TIMB counter overflow.

NOTE

When TOVx is set, a TIMB counter overflow Takes precedence over a channel x output compare if both occur at the same time.

CHxMAX — Channel x maximum duty cycle bit

When the TOVx bit is at logic zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 18-7 shows, the CHxMAX bit Takes effect in the cycle after it is set or cleared. The output stabs at the 100% duty cycle level until the cycle after CHxMAX is cleared.

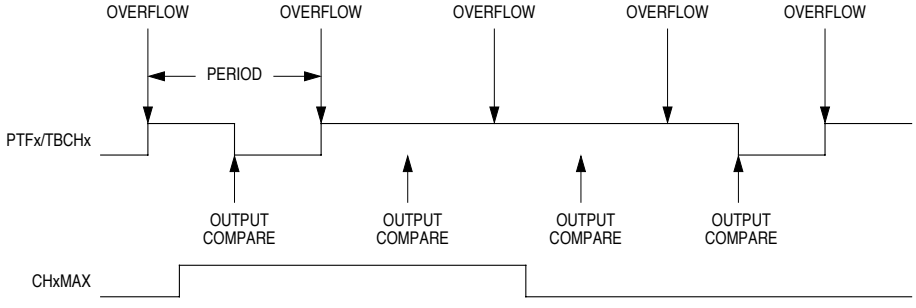


Figure 18-7 CHxMAX latency

18.8.5 TIMB channel registers (TBCH0H/L–TBCH1H/L)

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

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIMB channel x registers (TBCHxH) inhibits input captures until the low byte (TBCHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIMB channel x registers (TBCHxH) inhibits output compares until the low byte (TBCHxL) is written.

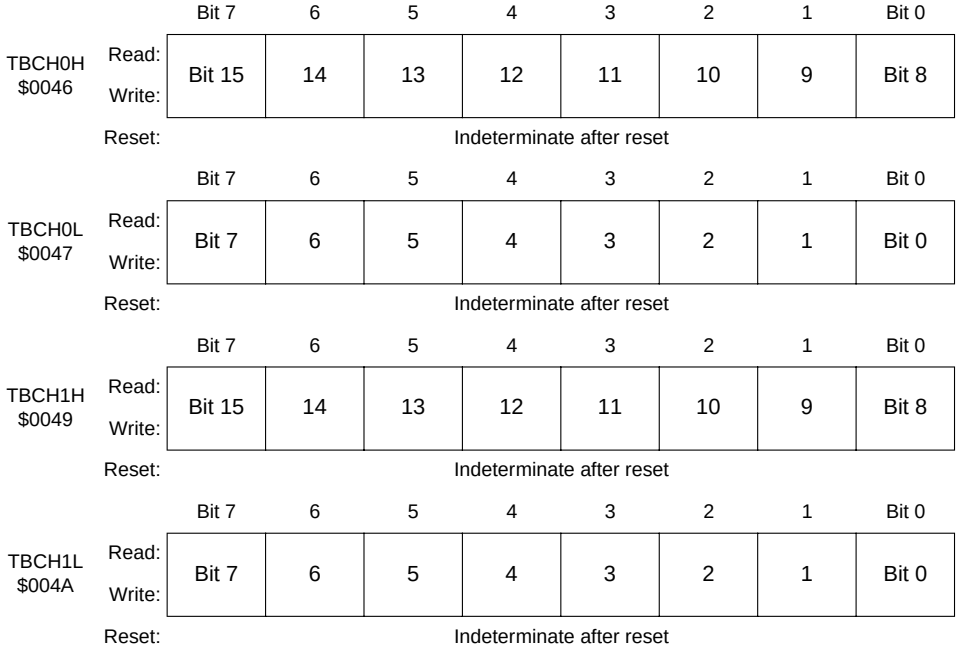


Figure 18-8 TIMB channel registers (TBCH0H/L–TBCH1H/L)

18

SECTION 19
PROGRAMMABLE INTERRUPT TIMER (PIT)

19.1 Introduction

This section describes the periodic interrupt timer module (PIT) is a block diagram of the PIT.

19.2 Features

Features of the PIT include the following:

- Programmable PIT Clock Input
- Free-Running or Modulo Up-Count Operation
- PIT Counter Stop and Reset Bits

19.3 Functional Description

Figure 19-1 shows the structure of the PIT. The central component of the PIT is the 16-bit PIT counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the interrupt. The PIT counter modulo registers, PMODH:PMODL, control the modulo value of the counter. Software can read the counter value at any time without affecting the counting sequence.

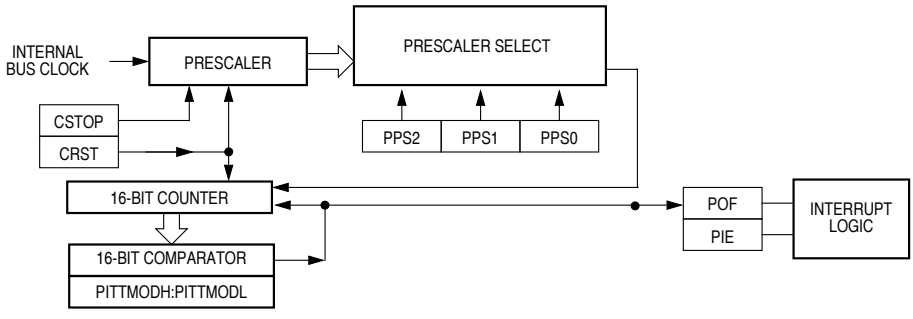


Figure 19-1 PIT Block Diagram

Table 19-1 PIT I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
PIT Status/Control Register (PSC)	POF	PIE	PSTOP	PRST	0	PPS2	PPS1	PPS0	$004B
PIT Counter Register. High (PCNTH)	Bit 15	14	13	12	11	10	9	8	$004C
PIT Counter Register. Low (PCNTL)	7	6	5	4	3	2	1	0	$004D
PIT Counter Modulo Reg. High (PMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$004E
PIT Counter Modulo Reg. Low (PMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$004F

= Unimplemented

19.3.1 PIT counter prescaler

The clock source can be one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PPS[2:0] in the status and control register select the PIT clock source.

The value in the PIT counter modulo registers and the selected prescaler output determines the frequency of the Periodic Interrupt. The PIT overflow flag (POF) is set when the PIT counter value rolls over to $0000 after matching the value in the PIT counter modulo registers. The PIT interrupt enable bit, PIE, enables PIT overflow CPU interrupt requests. POF and PIE are in the PIT status and control register.

19.4 Low-power modes

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

19.4.1 WAIT mode

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

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

19.4.2 STOP mode

The PIT is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the PIT counter. PIT operation resumes when the MCU exits stop mode after an external interrupt.

19.5 PIT during break interrupts

A break interrupt stops the PIT 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See **7.7.3 SIM break flag control register (SBFCR)**.

To allow software to clear status bits during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-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 zero. After the break, doing the second step clears the status bit.

19.6 I/O registers

The following I/O registers control and monitor operation of the PIT:

- PIT status and control register (PSC)
- PIT counter registers (PCNTH:PCNTL)
- PIT counter modulo registers (PMODH:PMODL)

19.6.1 PIT status and control register (PSC)

The PIT status and control register does the following:

- Enables PIT interrupt
- Flags PIT overflows
- Stops the PIT counter
- Resets the PIT counter
- Prescales the PIT counter clock

PSC $004B	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POF	PIE	PSTOP	0	0	PPS2	PPS1	PPS0
Write:	0			PRST				
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 19-2 PIT Status and Control Register (TSC)

POF — PIT overflow flag bit

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

1 = PIT counter has reached modulo value
0 = PIT counter has not reached modulo value

PIE — PIT overflow interrupt enable bit

This read/write bit enables PIT overflow interrupts when the POF bit becomes set. Reset clears the PIE bit.

1 = PIT overflow interrupts enabled
0 = PIT overflow interrupts disabled

PSTOP — PIT STOP bit

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

1 = PIT counter stopped
0 = PIT counter active

NOTE

Do not set the PSTOP bit before entering wait mode if the PIT is required to exit wait mode.

PRST — PIT reset bit

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

1 = Prescaler and PIT counter cleared
0 = No effect

NOTE

Setting the PSTOP and PRST bits simultaneously stops the PIT counter at a value of $0000.

PPS[2:0] — Prescaler select bits

These read/write bits select one of the seven prescaler outputs as the input to the PIT counter as Table 19-2 shows. Reset clears the PPS[2:0] bits.

Table 19-2 Prescaler selection

PS[2:0]	PIT 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

Table 19-2 Prescaler selection

PS[2:0]	PIT clock source
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	Internal Bus Clock ÷ 64

19.6.2 PIT counter registers (PCNTH:PCNTL)

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

If you read PCNTH during a break interrupt, be sure to unlatch PCNTL by reading PCNTL before exiting the break interrupt. Otherwise, PCNTL retains the value latched during the break.

		Bit 7	6	5	4	3	2	1	Bit 0
PCNTH $004C	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
PCNTL $004D	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
		0	0	0	0	0	0	0	0
	Reset:	= Unimplemented							

Figure 19-3 PIT counter registers (PCNTH:PCNTL)

19

19.6.3 PIT Counter modulo registers (PMODH:PMODL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PMODH $004E	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	1	1	1	1	1	1	1	1
		Bit 7	6	5	4	3	2	1	Bit 0
PMODL $004F	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	1	1	1	1	1	1	1	1

Figure 19-4 PIT Counter modulo registers (TMODH:TMODL)

Reset the PIT counter before writing to the PIT counter modulo registers.

19

SECTION 20
ANALOG TO DIGITAL CONVERTOR (ADC)

20.1 Introduction

This section describes the Analog to Digital Convertor. The ADC is an eight bit analog to digital convertor.

20.2 Features

Features of the ADC Module include the following:

- 8 channels with multiplexed input
- Linear successive approximation
- 8 bit resolution
- Single or continuous conversion
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock

20.3 Functional description

Eight ADC channels are available for sampling external sources at pins PTB7/ATD7-PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of eight ADC channels as ADC Voltage IN (ADCVIN). ADCVIN is converted by the successive approximation register based counter. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. See Figure 20-1.

NOTE

DMA section and associated functions are only valid if the MCU has a DMA module.

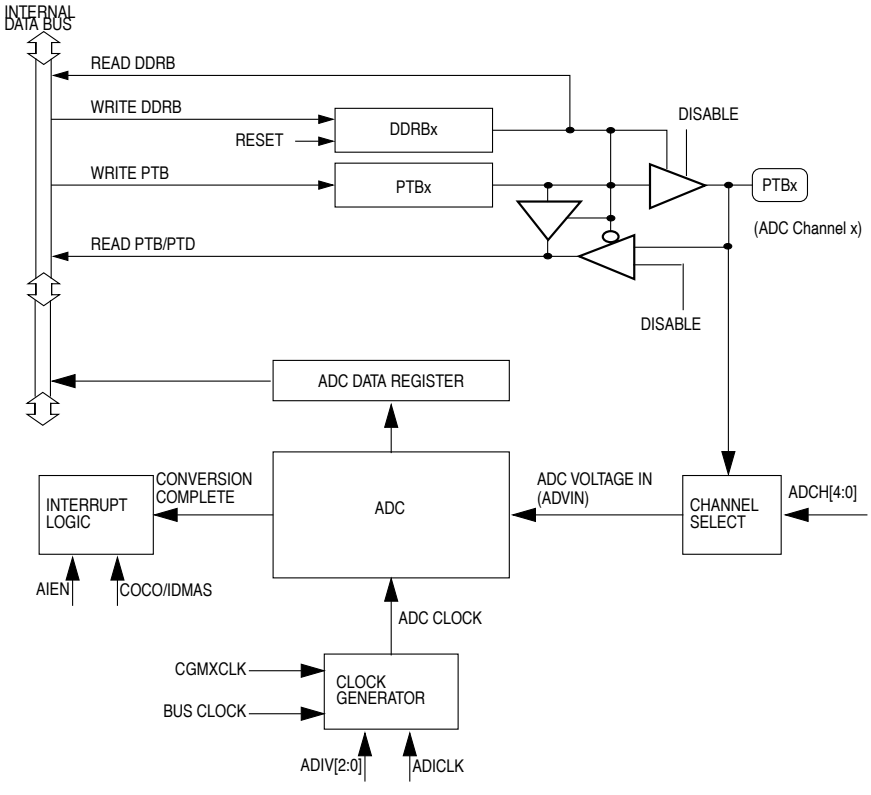


Figure 20-1 ADC block diagram

20.3.1 ADC port I/O pins

PTB7/ATD7-PTB0/ATD0 are general purpose I/O pins that shares with the ADC channels.

The Channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general purpose I/O. Writes to the port register or DDR will not have any affect on the port pin that is selected by the ADC. Read of a port pin which is in use by the ADC will return a logic zero.

20.3.2 Voltage conversion

When the input voltage to the ADC equals to VREFH, the ADC converts the signal to $FF (full scale). If the input voltage equals to A_{VSS}/VREFL, the ADC converts it to $00. Input voltages between VREFH and A_{VSS}/VREFL is a straight-line linear conversion. All other input voltages will result in $FF if greater than VREFH and $00 if less than A_{VSS}/VREFL.

NOTE

Input voltage should not exceed the analog supply voltages.

20.3.3 Conversion time

Conversion starts after a write to the ADSCR. Conversion time in terms of the number of bus cycles is a function of oscillator frequency, bus frequency, and ADIV prescaler bits. For example, with oscillator frequency of 4MHz, bus frequency of 8MHz and ADC clock frequency of 1MHz, one conversion will take between 16 ADC and 17 ADC clock cycles or between 16 and 17 μs in this case. There will be 128 bus cycles between each conversion. Sample rate is approximately 60kHz.

$$\text{Conversion Time} = \frac{\text{16-17 ADC cycles}}{\text{ADC frequency}}$$

$$\text{\# Bus Cycles} = \text{Conversion Time} \times \text{Bus Frequency}$$

20.3.4 Continuous conversion

In the continuous conversion mode, the ADC Data Register will be filled with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit is cleared. The COCO bit is set after the first conversion and will stay set for the next several conversions until the next write of the ADC status and control register or the next read of the ADC data register.

20.3.5 Accuracy and precision

The conversion process is monotonic and has no missing codes. See **20.3.5 Accuracy and precision** for accuracy information.

20.4 Interrupts

When the AIEN bit is set, the ADC module is capable of generating either CPU or DMA interrupts after each ADC conversion. A CPU interrupt is generated if the COCO/IDMAS bit is at logic zero. If COCO/IDMAS bit is set, a DMA interrupt is generated. The COCO/IDMAS bit is not used as a conversion complete flag when interrupts are enabled.

20.5 Low power modes

The WAIT and STOP instruction can put the MCU in low power consumption standby modes.

20.5.1 WAIT mode

The ADC continues normal operation during WAIT mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH[4:0] bits in the ADC Status and Control Register before executing the WAIT instruction.

20.5.2 STOP mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode after an external interrupt. Allow one conversion cycle to stabilize the analog circuitry.

20.6 I/O signals

The ADC module has eight I/O that are shared with Port B.

20.6.1 ADC analog power pin (V_{DDAREF})

The ADC analog portion uses as its power pin. Connect the V_{DDAREF} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDAREF} for good results.

NOTE

Route V_{DDAREF} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.

20.6.2 ADC analog ground pin (A_{VSS}/VREFL)

The ADC analog portion uses A_{VSS}/VREFL as its ground pin. Connect the A_{VSS}/VREFL pin to the same voltage potential as V_{SS}.

20.6.3 ADC voltage reference pin (VREFH)

VREFH is the reference voltage for the ADC.

20.6.4 ADC voltage in (ADVIN)

ADVIN is the input voltage signal from one of the eight ADC channels to the ADC module.

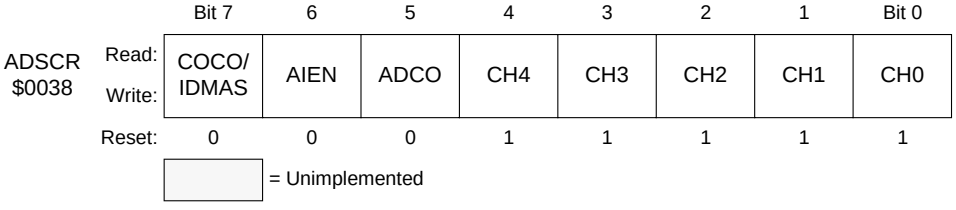
20.7 I/O registers

The following I/O registers control and monitor operation of the ADC:

- ADC status and control register (ADSCR)
- ADC data register (ADR)
- ADC clock register (ADCLK)

20.7.1 ADC status and control register (ADSCR)

The following paragraphs describe the function of the ADC Status and Control Register.



		Bit 7	6	5	4	3	2	1	Bit 0
ADSCR $0038	Read:	COCO/ IDMAS	AIEN	ADCO	CH4	CH3	CH2	CH1	CH0
	Write:								
	Reset:	0	0	0	1	1	1	1	1

= Unimplemented

Figure 20-2 ADC status and control register

COCO/IDMAS - Conversions complete/interrupt DMA select

When AIEN bit is a logic zero, the COCO/IDMAS is a read only bit which is set each time a conversion is completed except in the continuous conversion mode where it is set after the first conversion. This bit is cleared whenever the ADC Status and Control Register is written or whenever the ADC Data Register is read.

If AIEN bit is a logic one, the COCO/IDMAS is a read/write bit which selects either CPU or DMA to service the ADC interrupt request. Reset clears this bit.

1 = conversion completed (AIEN=0) / DMA interrupt (AIEN=1)
0 = conversion not completed (AIEN=0) / CPU interrupt (AIEN=1)

AIEN - ADC interrupt enable

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the Data Register is read or the Status/Control register is written. Reset clears AIEN bit.

1 = ADC Interrupt enabled
0 = ADC Interrupt disabled

ADCO - ADC continuous conversion

When set, the ADC will continuously convert samples and update the ADR register at the end of each conversion. Only one conversion is allowed when this bit is cleared. Reset clears the ADCO bit.

1 = continuous ADC conversion
0 = one ADC conversion

ADCH[4:0] - ADC channel select bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of eight ADC channels. The eight channels are detailed in the following table. Care should be taken when using a port pin as both an analog and digital input simultaneously to prevent switching noise from corrupting the analog signal. See Table 20-1.

The ADC subsystem is turned off when the channel select bits are all set to one. This feature allows for reduced power consumption for the MCU when the ADC is not used.

NOTE

Recovery from the disabled state requires one conversion cycle to stabilize.

The voltage levels supplied from internal reference nodes as specified in the table are used to verify the operation of the ADC converter both in production test and for user applications.

Table 20-1 Mux channel select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	0	0	0	0	PTB0/ATD0
0	0	0	0	1	PTB1/ATD1
0	0	0	1	0	PTB2/ATD2
0	0	0	1	1	PTB3/ATD3
0	0	1	0	0	PTB4/ATD4
0	0	1	0	1	PTB5/ATD5
0	0	1	1	0	PTB6/ATD6
0	0	1	1	1	PTB7/ATD7
0	1	0	0	0	Unused
0	1	0	0	1	Unused
0	1	0	1	0	Unused
0	1	0	1	1	Unused
0	1	1	0	0	Unused
0	1	1	0	1	Unused
0	1	1	1	0	Unused
0	1	1	1	1	Unused
1	0	0	0	0	Unused *
↓	↓	↓	↓	↓	↓
1	1	0	1	0	Unused *

* If any unused channels are selected, the resulting ADC conversion will be unknown.

20.7.2 ADC data register (ADR)

One 8-bit result register is provided. This register is updated each time an ADC conversion completes.

		Bit 7	6	5	4	3	2	1	Bit 0
ADR $0039	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 20-3 ADC data register

20.7.3 ADC clock register (ADCLKR)

This register selects the clock frequency for the ADC

		Bit 7	6	5	4	3	2	1	Bit 0
ADCLK $003A	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 20-4 ADC clock register

ADIV2:ADIV0 - ADC clock prescaler bits

ADIV2, ADIV1and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. Table 20-2 shows the available clock configurations. The ADC clock should be set to approximately 1MHz.

Table 20-2 ADC clock divide ratio

ADIV2	ADIV1	ADIV0	ADC Clock Rate
0	0	0	ADC input clock /1
0	0	1	ADC input clock / 2
0	1	0	ADC input clock / 4
0	1	1	ADC input clock / 8
1	X	X	ADC input clock / 16
X = don't care			

ADICLK – ADC input clock select

ADICLK selects either bus clock or cgmxclk as the input clock source to generate the internal ADC clock. Reset selects cgmxclk as the ADC clock source.

If the external clock (cgmxclk) is equal or greater than 1MHz, cgmxclk can be used as the clock source for the ADC. If cgmxclk is less than 1MHz, use the PLL generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1MHz, correct operation can be guaranteed. See **20.3.3 Conversion time**.

1 = Internal bus clock

0 = External clock (cgmxclk)

$$1\text{MHz} = \frac{\text{cgmxclk or bus frequency}}{\text{ADIV[2:0]}}$$

SECTION 21
KEYBOARD MODULE (KB)

21.1 Introduction

The keyboard module provides five independently maskable external interrupt pins.

21.2 Features

Features of the keyboard module include the following:

- Five Keyboard Interrupt Pins and Interrupt Masks
- Selectable triggering sensitivity

21.3 Functional description

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port G or port H pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its pull-up device. A logic zero applied to a keyboard interrupt pin can latch a keyboard interrupt request.

The keyboard interrupt latch becomes set 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 sensitivity of the keyboard interrupt latch.

- If the keyboard interrupt latch 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 former pin while it is low.
- If the keyboard interrupt latch is edge- and level-sensitive, an interrupt request is latched as long as any keyboard pin is low.

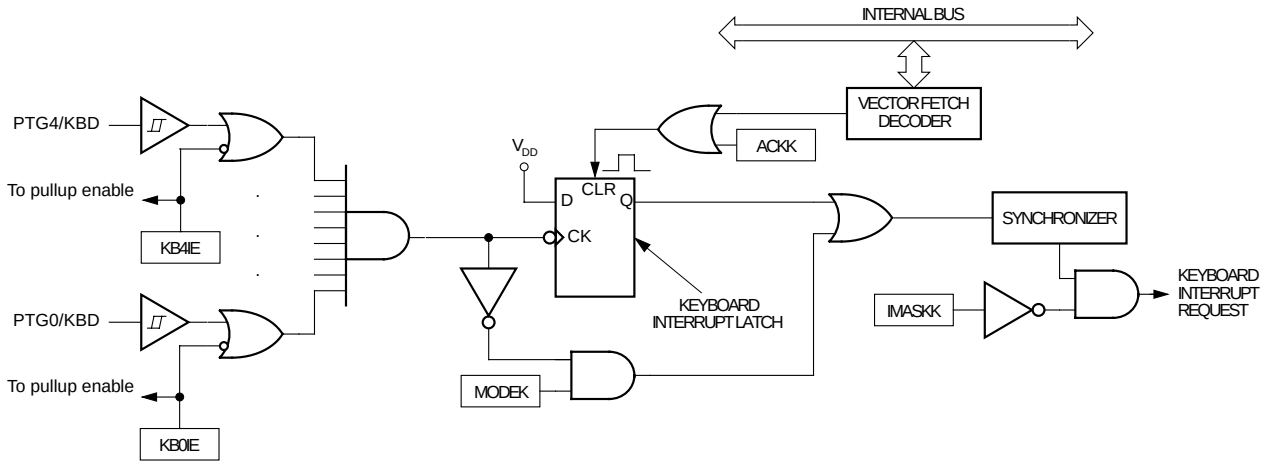


Figure 21-1. Keyboard module block diagram

Table 21-1. KB I/O register summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Keyboard Status/Control Register (KBSCR)	R:	0	0	0	0	KEYF	0	IMASKK	MODEK	$001A
	W:						ACKK			
Reset		0	0	0	0	0	0	0	0	
Keyboard Interrupt Control Register (KBICR)	R:	0	0	0	KB4IE	KB3IE	KB2IE	KB1IE	KB0IE	$001B
	W:									
Reset		0	0	0	0	0	0	0	0	

= Unimplemented

The MODEK bit in the keyboard status and control register controls the triggering sensitivity of the keyboard interrupt latch. 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 the keyboard interrupt latch:

- 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 one 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 latch. Writing to the ACKK bit can also 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 $FFD2 and $FFD3.
- Return of all enabled keyboard interrupt pins to logic one — As long as any enabled keyboard interrupt pin is at logic zero, the keyboard interrupt latch remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic one may occur in any order. The interrupt request remains pending as long as any enabled keyboard interrupt pin is at logic zero.

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 latch.

Reset clears the keyboard interrupt latch and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic zero.

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 (KBxIE) 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 zero for software to read the pin.

21.4 I/O Registers

The following registers control and monitor operation of the keyboard module:

- Keyboard status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)

21.4.1 Keyboard status and control register (KBSCR)

The keyboard status and control register performs the following functions:

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard latch triggering sensitivity

		Bit 7	6	5	4	3	2	1	Bit 0
KBSCR $001B	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
	Write:						ACKK		
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 21-2 Keyboard status and control register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as logic zeros.

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 one to this read/write bit clears the keyboard interrupt latch. ACKK always reads as logic zero. Reset clears ACKK.

IMASKK — Keyboard interrupt mask bit

Writing a logic one 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 disabled
0 = Keyboard interrupt requests enabled

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

21.4.2 Keyboard interrupt enable register (KBIER)

The keyboard interrupt enable register enables or disables each port G or port H pin to operate as a keyboard interrupt pin.

		Bit 7	6	5	4	3	2	1	Bit 0
KBIER $0021	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
	Write:								
	Reset	0	0	0	0	0	0	0	0

Figure 21-3 Keyboard interrupt enable register (KBIER)

KBIE4: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 = Pin enabled as keyboard interrupt pin
0 = Pin not enabled as keyboard interrupt pin

21.5 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 SIM break flag control register (SBFCR) enables software to clear the latch during the break state.

To allow software to clear the keyboard interrupt latch during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (its default state), writing during the break state to the keyboard acknowledge bit (ACKK) in the keyboard status and control register has no effect. See **21.4.1 Keyboard status and control register (KBSCR)**.

21

SECTION 22
I/O PORTS

22.1 Introduction

Forty-nine bidirectional input-output (I/O) pins form eight 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.

Table 22-1 I/O port register summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port A Data Register (PTA)	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0	$0000
Port B Data Register (PTB)	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	$0001
Port C Data Register (PTC)	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	$0002
Port D Data Register (PTD)	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	$0003
Data Direction Register A (DDRA)	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	$0004
Data Direction Register B (DDRB)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	$0005
Data Direction Register C (DDRC)	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	$0006
Data Direction Register D (DDRD)	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDR2	DDRD1	DDRD0	$0007
Port E Data Register (PTE)	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	$0008
Port F Data Register (PTF)	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0	$0009
Port G Data Register (PTG)	0	0	0	0	0	PTG2	PTG1	PTG0	$000A

Table 22-1 I/O port register summary (Continued)

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port H Data Register (PTH)	0	0	0	0	0	0	PTH1	PTH0	$000B
Data Direction Register E (DDRE)	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0	$000C
Data Direction Register F (DDRF)	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0	$000D
Data Direction Register G (DDRG)	0	0	0	0	0	DDRG2	DDRG1	DDRG0	$000E
Data Direction Register H (DDRH)	0	0	0	0	0	0	DDRH1	DDRH0	$000F

22.2 Port A

Port A is an 8-bit general-purpose bidirectional I/O port.

22.2.1 Port A Data Register (PTA)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTA $0000	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
	Reset:	Unaffected by reset							

Figure 22-1 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.

22.2.2 Data direction register A (DDRA)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRA $0004	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
	Reset:	0	0	0	0	0	0	0	0

Figure 22-2 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 22-3 shows the port A I/O logic.

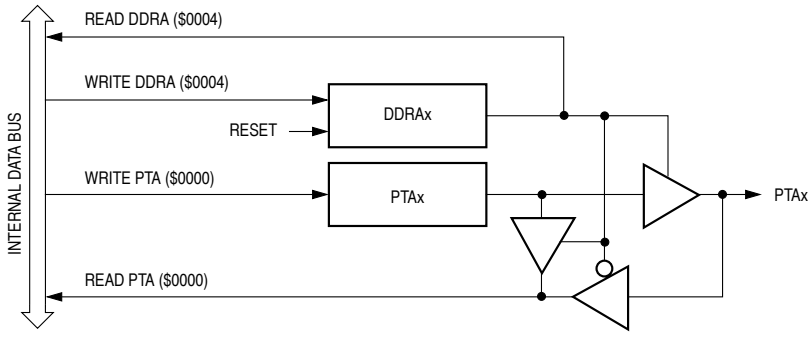


Figure 22-3 Port A I/O Circuit

When bit DDRAx is a logic one, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic zero, 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 22-2 summarizes the operation of the port A pins.

Table 22-2 Port A pin functions

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

22.3 Port B

Port B is an 8-bit special function port that shares all of its pins with the analog to digital convertor.

22.3.1 Port B data register (PTB)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTB $0001	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
	Reset:	Unaffected by reset							
ALTERNATE FUNCTIONS		ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD1	ATD0

Figure 22-4 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.

ATD[7:0] — ADC channels

NOTE

PTB7/ATD7- PTB0/ATD0 are eight of the sixteen analog to digital convertor channels. The ADC channel select bits, CH[4:0], determine whether the PTB7/ATD7- PTB0/ATD0 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be zero if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns the states of the latches or logic 0.

22.3.2 Data direction register B (DDRB)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRB $0005	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
	Reset:	0	0	0	0	0	0	0	0

Figure 22-5 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 22-6 shows the port B I/O logic.

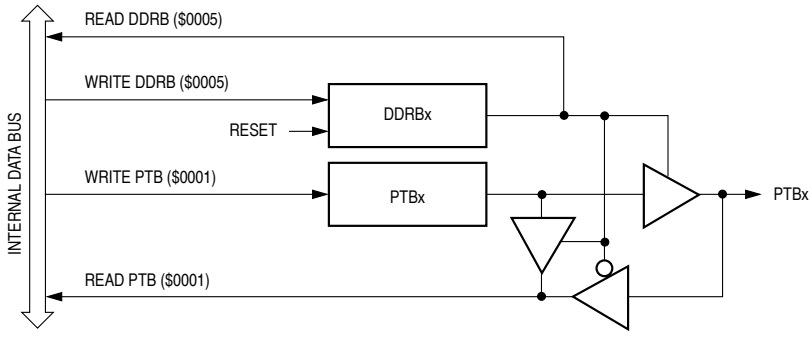


Figure 22-6 Port B I/O circuit

When bit DDRBx is a logic one, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic zero, 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 22-3 summarizes the operation of the port B pins.

Table 22-3 Port B pin functions

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

22.4 Port C

Port C is a 6-bit general-purpose bidirectional I/O port.

22.4.1 Port C data register (PTC)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTC $0002	Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
	Write:								
	Reset:	Unaffected by reset							
ALTERNATE FUNCTIONS							MCLK		

Figure 22-7 Port C data register (PTC)

PTC[5:0] — 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.

MCLK — T12 System Clock

The system clock is driven out of PTC2 when enabled by MCLKEN.

22.4.2 Data direction register C (DDRC)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRC $0006	Read:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

Figure 22-8 Data direction register C (DDRC)

MCLKEN — MCLK enable bit

This read/write bit enables mclk to be an output signal on PTC2. If MCLK is enabled, PTC2 is under the control of MCLKEN. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

DDRC[5:0] — 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 22-9 shows the port C I/O logic.

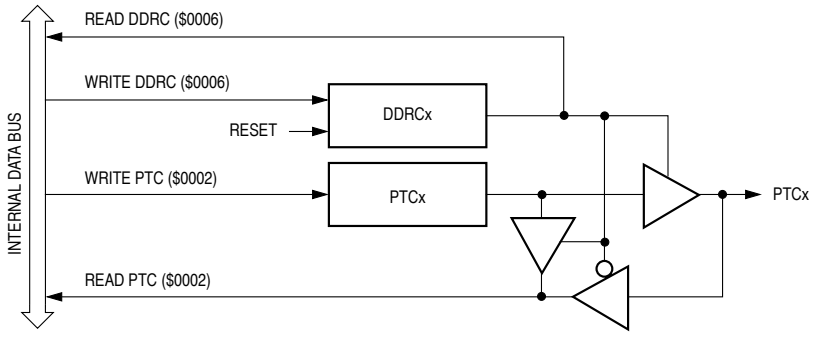


Figure 22-9 Port C I/O circuit

When bit DDRCx is a logic one, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic zero, 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 22-4 summarizes the operation of the port C pins.

Table 22-4 Port C pin functions

Bit Value	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	2	Input, Hi-Z	DDRC[7]	Pin	PTC2
1	2	Output	DDRC[7]	0	—
0	X(1)	Input, Hi-Z(2)	DDRC[5:0]	Pin	PTC[5:0](3)
1	X	Output	DDRC[5:0]	PTC[5:0]	PTC[5:0]

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

22.5 Port D

Port D is an 8-bit general-purpose I/O port.

22.5.1 Port D data register (PTD)

Port D is an 8 -bit special function port that shares two of it's pins with the TIMA and TIMB modules.

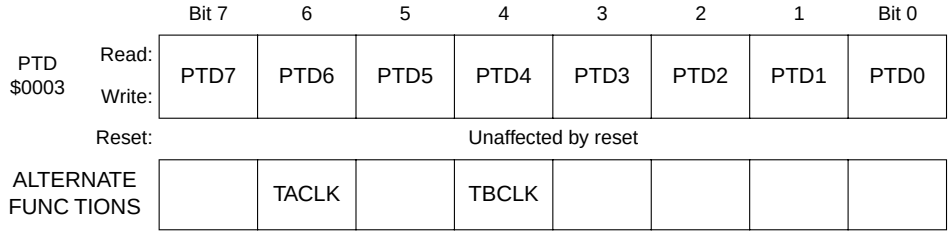


Figure 22-10 Port D data register (PTD)

PTD[7:0] — Port D data bits

PTD[7:0] are read/write, software programmable bits. Data direction of PTD[7:0] pins are under the control of the corresponding bit in data direction register D.

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

TACLK — Timer clock input

The PTD6/TACLK pin is the external clock input for the TIMA. The prescaler select bits, PS[2:0], select PTD6/TACLK as the TIMA clock input. See **SECTION 17 TIMER INTERFACE MODULE A (TIMA)**. When not selected as the TIM clock.

TBCLK — Timer Clock Input

The PTD4 pin is the external clock input for the TIMB. The prescaler select bits, PS[2:0], select PTD4 as the TIMB clock input. See **SECTION 18 TIMER INTERFACE MODULE B (TIMB)**. When not selected as the TIM clock.

22.5.2 Data direction register D (DDRD)

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

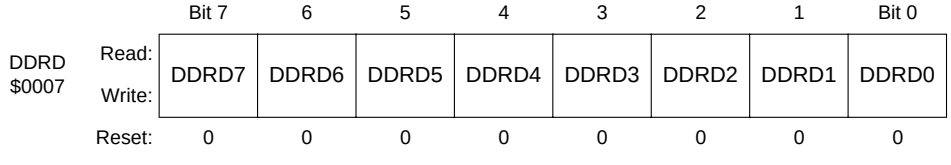


Figure 22-11 Data direction register D (DDRD)

22

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 22-12 shows the port D I/O logic.

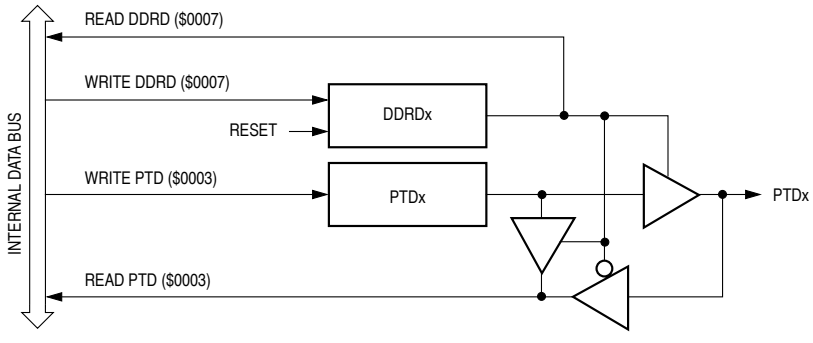


Figure 22-12 Port D I/O circuit

When bit DDRDx is a logic one, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic zero, 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 22-5 summarizes the operation of the port D pins.

Table 22-5 Port D pin functions

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

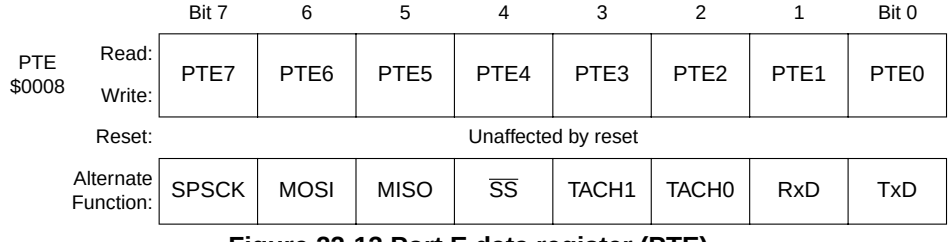
1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

22.6 Port E

Port E is an 8-bit special function port that shares two of its pins with the timer interface module (TIMA), two of its pins with the serial communications interface module (SCI) and four of its pins with the serial peripheral interface module (SPI).

22.6.1 Port E data register (PTE)

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



		Bit 7	6	5	4	3	2	1	Bit 0
PTE $0008	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
	Reset:	Unaffected by reset							
	Alternate Function:	SPSCK	MOSI	MISO	$\overline{SS}$	TACH1	TACH0	RxD	TxD

Figure 22-13 Port E data register (PTE)

PTE[7:0] — Port E data bits

PTE[7:0] 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.

SPSCK — SPI serial clock

The PTE7/SPSCK pin is the serial clock input of a SPI slave module and serial clock output of a SPI master modules. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O.

MOSI — Master Out/Slave In

The PTE6/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTE6/MOSI pin is available for general-purpose I/O. See **16.13.1 SPI control register (SPCR)**.

MISO — Master In/Slave Out

The PTE5/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTE5/MISO pin is available for general-purpose I/O. See **16.13.1 SPI control register (SPCR)**.

$\overline{SS}$ — Slave Select

The PTE4/$\overline{SS}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set, the PTE4/$\overline{SS}$ pin is available for general-purpose I/O. See **16.13.1 SPI control register (SPCR)**. When the SPI is enabled as a slave, the DDRF0 bit in data direction register E (DDRE) has no effect on the PTE4/$\overline{SS}$ pin.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 22-6.

TACH[1:0] — Timer A channel I/O bits

The PTE3/TACH1–PTE2/TACH0 pins are the TIMA input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TACH1–PTE2/TACH0 pins are timer channel I/O pins or general-purpose I/O pins. See **17.8.4 TIMA channel status and control registers (TASC0–TASC3)**.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIMA. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 22-6.

RxD — SCI Receive data input

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. See **15.7.1 SCI control register 1 (SCC1)**.

TxD — SCI transmit data output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. See **15.7.2 SCI Control Register 2 (SCC2)**.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 22-6.

22.6.2 Data direction register E (DDRE)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRE $000C	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
	Reset:	0	0	0	0	0	0	0	0

Figure 22-14 Data direction register E (DDRE)

DDRE[7:0] — Data direction register E bits

These read/write bits control port E data direction. Reset clears DDRE[7:0], 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 22-15 shows the port E I/O logic.

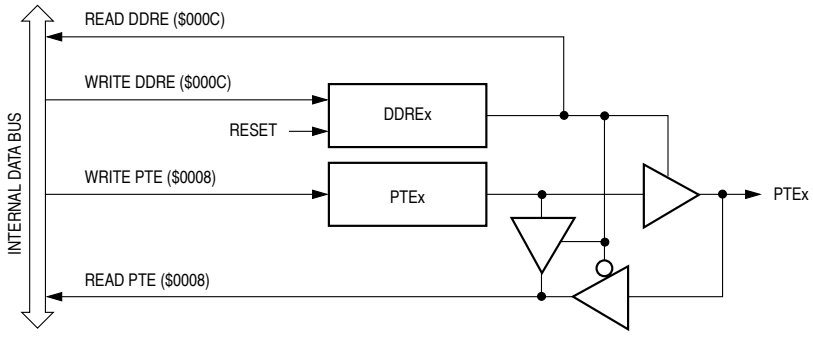


Figure 22-15 Port E I/O circuit

When bit DDREx is a logic one, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic zero, 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 22-6 summarizes the operation of the port E pins.

Table 22-6 Port E pin functions

DDRE Bit	PTE Bit	I/O Pin Mode	Accesses to DDRE	Accesses to PTE	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRE[7:0]	Pin	PTE[7:0](3)
1	X	Output	DDRE[7:0]	PTE[7:0]	PTE[7:0]

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

22.7 Port F

Port F is a 7-bit special function port that shares four of its pins with the timer interface modules (TIMA and TIMB).

22.7.1 Port F data register (PTF)

The port F data register contains a data latch for each of the six port F pins.

		Bit 7	6	5	4	3	2	1	Bit 0
PTF $0009	Read:	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
	Write:								
	Reset:	Unaffected by reset							
	Alternate Function:			TBCH1	TBCH0			TACH3	$\overline{\text{TACH2}}$

= Unimplemented

Figure 22-16 Port F data register (PTF)

PTF[6:0] — Port F data bits

These read/write bits are software programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on PTF[6:0].

TACH[3:2] and TBCH[1:0] — Timer channel I/O bits

The PTF5/TBCH1–PTF0/TACH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF5/TBCH1–PTF0/TACH2 pins are timer channel I/O pins or general-purpose I/O pins. See **17.8.4 TIMA channel status and control registers (TASC0–TASC3)** and **18.8.4 TIMB channel status and control registers (TBSC0–TBSC1)**.

NOTE

Data direction register F(DDRF) does not affect the data direction of port F pins that are being used by TIMA and TIMB. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. See Table 22-7.

22.7.2 Data direction register F (DDRF)

Data direction register F determines whether each port F pin is an input or an output. Writing a logic one to a DDRF bit enables the output buffer for the corresponding port F pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRF $000D	Read:	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
	Write:								
	Reset:		0	0	0	0	0	0	0

= Unimplemented

Figure 22-17 Data direction register F (DDRF)

DDRF[6:0] — Data direction register F bits

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

1 = Corresponding port F pin configured as output
0 = Corresponding port F pin configured as input

NOTE

Avoid glitches on port F pins by writing to the port F data register before changing data direction register F bits from 0 to 1.

Figure 22-18 shows the port F I/O logic.

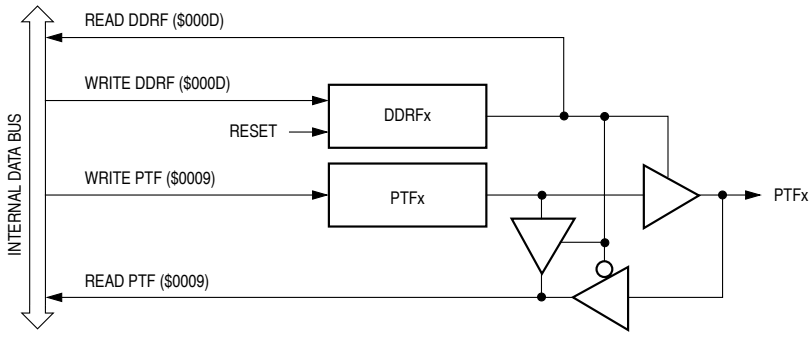


Figure 22-18 Port F I/O circuit

When bit DDRFx is a logic one, reading address $0009 reads the PTFx data latch. When bit DDRFx is a logic zero, reading address $0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-7 summarizes the operation of the port F pins.

Table 22-7 Port F pin functions

DDRF Bit	PTF Bit	I/O Pin Mode	Accesses to DDRF	Accesses to PTF	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRF[6:0]	Pin	PTF[6:0](3)
1	X	Output	DDRF[6:0]	PTF[6:0]	PTF[6:0]

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

22.8 Port G

Port G is a 3-bit general-purpose bidirectional I/O port.

22.8.1 Port G data register (PTG)

The port G data register contains a data latch for each of the three port G pins.

		Bit 7	6	5	4	3	2	1	Bit 0
PTG $000A	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
	Write:								
	Reset:	Unaffected by reset							
Alternate Function							KBD2	KBD1	KBD0
		= Unimplemented							

Figure 22-19 Port G data register (PTG)

PTG[2:0] — Port G Data Bits

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

KBD[2:0] — Keyboard Inputs

The keyboard interrupt enable bits, KBIE[2:0], in the keyboard interrupt control register (KBICR), enable the port G pins as external interrupt pins. See **SECTION 21 KEYBOARD MODULE (KB)**.

22.8.2 Data direction register G (DDRG)

Data direction register G determines whether each port G pin is an input or an output. Writing a logic one to a DDRG bit enables the output buffer for the corresponding port G pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRG $000E	Read:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
		= Unimplemented							

Figure 22-20 Data direction register G (DDRG)

DDRG[2:0] — Data direction register G bits

These read/write bits control port G data direction. Reset clears DDRG[2:0], configuring all port G pins as inputs.

1 = Corresponding port G pin configured as output
0 = Corresponding port G pin configured as input

NOTE

Avoid glitches on port G pins by writing to the port G data register before changing data direction register G bits from 0 to 1.

Figure 22-21 shows the port G I/O logic.

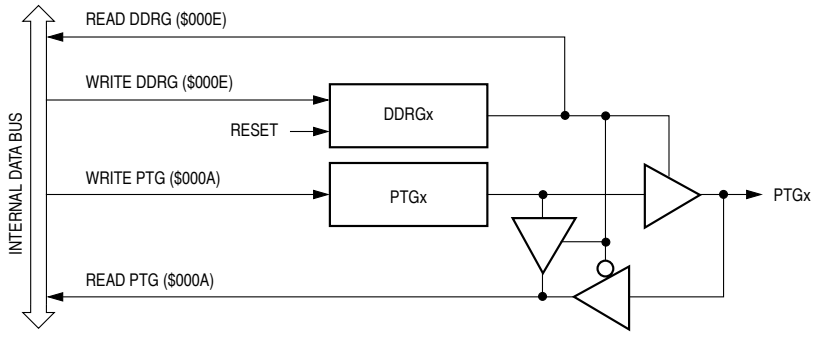


Figure 22-21 Port G I/O circuit

When bit DDRGx is a logic one, reading address $000A reads the PTGx data latch. When bit DDRGx is a logic zero, reading address $000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data.

22.9 Port H

Port H is a 2-bit general-purpose bidirectional I/O port.

22.9.1 Port H data register (PTH)

The port Hdata register contains a data latch for each of the two port H pins.

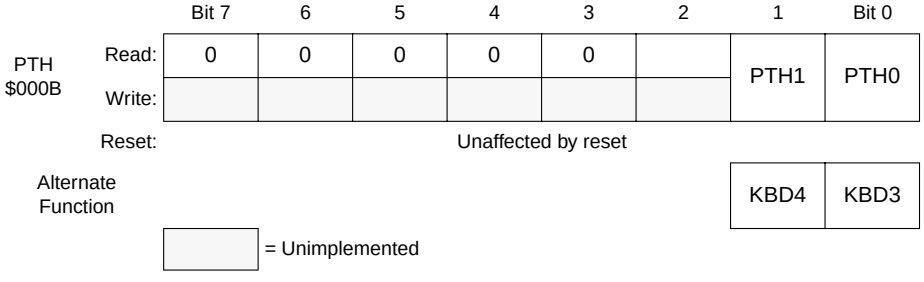


Figure 22-22 Port H data register (PTH)

PTH[1:0] — Port H data bits

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

KBD[4:3] — Keyboard inputs

The keyboard interrupt enable bits, KBIE[4:3], in the keyboard interrupt control register (KBICR), enable the port H pins as external interrupt pins See **SECTION 21 KEYBOARD MODULE (KB)**.

22.9.2 Data direction register H (DDRH)

Data direction register H determines whether each port H pin is an input or an output. Writing a logic one to a DDRH bit enables the output buffer for the corresponding port H pin; a logic zero disables the output buffer.

Figure 22-23 Data direction register H (DDRH)

DDRH[1:0] — Data direction register H bits

These read/write bits control port H data direction. Reset clears DDRH[1:0], configuring all port H pins as inputs.

1 = Corresponding port H pin configured as output
0 = Corresponding port H pin configured as input

NOTE

Avoid glitches on port H pins by writing to the port H data register before changing data direction register H bits from 0 to 1.

Figure 22-21 shows the port H I/O logic.

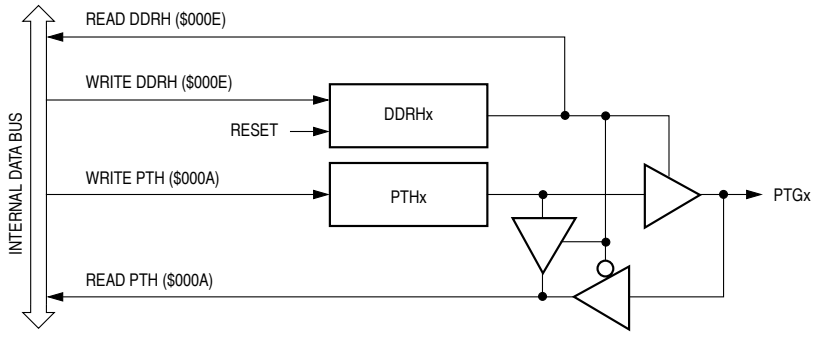


Figure 22-24 Port H I/O circuit

When bit DDRHx is a logic one, reading address $000B reads the PTHx data latch. When bit DDRHx is a logic zero, reading address $000B reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data.

SECTION 23
msCAN08 CONTROLLER

The msCAN08 is the specific implementation of the Motorola Scalable CAN (msCAN) concept targeted for the Motorola M68HC08 Microcontroller family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth.

msCAN08 utilizes an advanced buffer arrangement resulting in a predictable real-time behaviour and simplifies the application software.

23.1 Features

The basic features of the msCAN08 are as follows:

- Modular architecture
- Implementation of the CAN protocol - Version 2.0A/B
  - Standard and extended data frames.
  - 0 - 8 bytes data length.
  - Programmable bit rate up to 1 Mbps(1).
- Support for remote frames.
- Double buffered receive storage scheme.
- Triple buffered transmit storage scheme with internal prioritization using a “local priority” concept.
- Flexible maskable identifier filter supports alternatively one full size extended identifier filter or two 16 bit filters or four 8 bit filters.
- Programmable wake-up functionality with integrated low-pass filter.
- Programmable loop-back mode supports self-test operation.
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (Warning, Error Passive, Bus-Off).

1. Depending on the actual bit timing and the clock jitter of the PLL.

- Programmable msCAN08 clock source either CPU bus clock or crystal oscillator output.
- Programmable link to on-chip Timer Interface Module (TIM) for time-stamping and network synchronization.
- Low power sleep mode.

23.2 External pins

The msCAN08 uses 2 external pins, 1 input (RxCAN) and 1 output (TxCAN). The TxCAN output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state.

A typical CAN system with msCAN08 is shown in Figure 23-1 below.

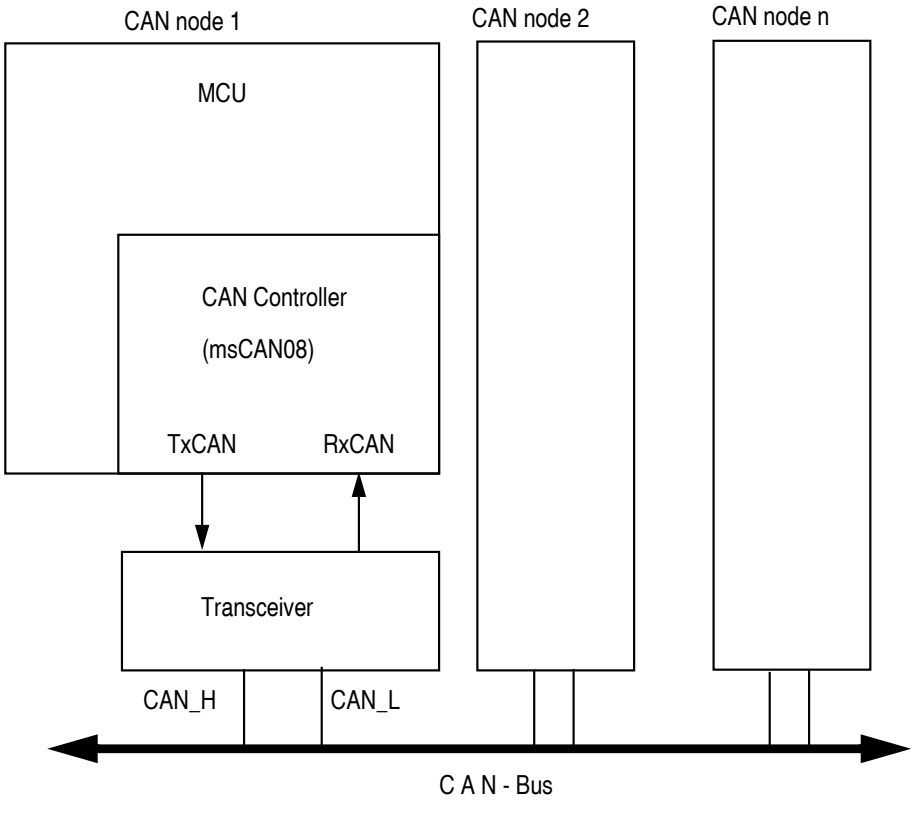


Figure 23-1 The CAN system

Each CAN station is physically connected to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection, against defected CAN or defected stations.

23.3 Message storage

msCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

23.3.1 Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behaviour can not be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the Inter-Frame Sequence (IFS) in order to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message sending and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of above requirements under all circumstances. The msCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the msCAN08 implements with the “local priority” concept described below.

23.3.2 Receive structures

The received messages are stored in a two stage input FIFO. The two message buffers are mapped using a ‘ping pong’ arrangement into a single memory area (see Figure 23-2). While the background receive buffer (RxBG) is exclusively associated to the msCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software as only one address area is applicable for the receive process.

Both buffers have a size of 13 byte to store the CAN control bits, the identifier (standard or extended) and the data content (for details see Section 23.11).

The Receiver Full flag (RXF) in the msCAN08 Receiver Flag Register (CRFLG) (see Section 23.12.6) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier this flag is set.

After the msCAN08 successfully received a message into the background buffer it copies the content of RxBG into RxFG(1), sets the RXF flag, and emits a receive interrupt to the CPU(2). A new message - which may follow immediately after the IFS field of the CAN frame - will be received into RxBG.

The user's receive handler has to read the received message from RxFG and to reset the RXF flag in order to acknowledge the interrupt and to release the foreground buffer.

An overrun conditions occurs when both, the foreground and the background receive message buffers are filled with correctly received messages and a further message is being received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. The over-writing of the background buffer is independent of the identifier filter function. While in the overrun situation, the msCAN08 will stay synchronized to the CAN bus and is able to transmit messages but will discard all incoming messages.

1. Only if the RXF flag is not set.
2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

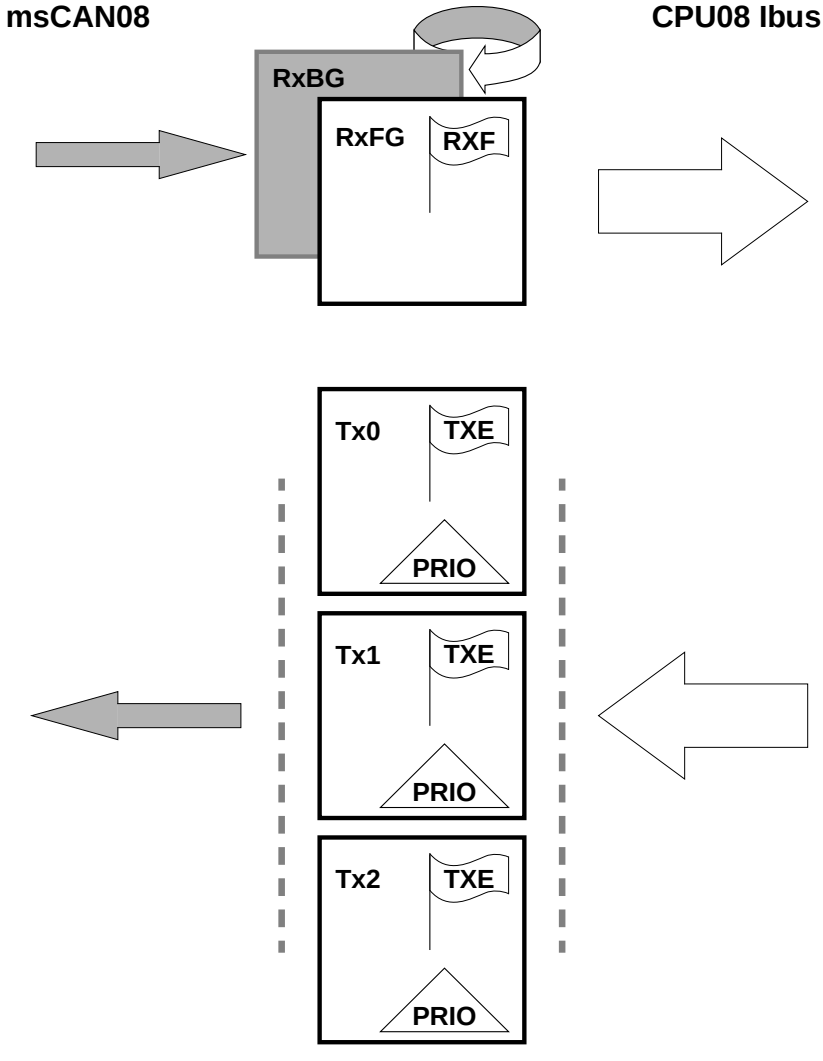


Figure 23-2 User model for Message Buffer organization

msCAN08 will receive its own messages into the background receive buffer RxBG but will NOT overwrite RxFG and will NOT emit a receive interrupt nor will it acknowledge (ACK) its own messages on the CAN bus. The exception to this rule is that when in loop-back mode msCAN08 will treat its own messages exactly like all other incoming messages.

23.3.3 Transmit structures

The msCAN08 has a triple transmit buffer scheme in order to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 23-2.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers (see Section 23.11). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit so called “Local Priority” field (PRIO) (see Section 23.11.5).

In order to transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set Transmit Buffer Empty (TXE) Flag in the msCAN08 Transmitter Flag Register (CTFLG) (see Section 23.12.8).

The CPU08 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged as being ready for transmission by clearing the TXE flag.

The msCAN08 will then schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt will be emitted(1) when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the msCAN08 uses the “local priority” setting of the three buffers for prioritization. For this purpose every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority.

The internal scheduling process takes places whenever the msCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission can not be aborted, the user has to request the abort by setting the corresponding Abort Request Flag (ABTRQ) in the Transmission Control Register (CTCR). The msCAN08 will then grant the request if possible by setting the corresponding Abort Request Acknowledge (ABTAK) and the TXE flag in order to release the buffer and by emitting a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK=1) or has been sent in the meantime (ABTAK=0).

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

23.4 Identifier acceptance filter

A very flexible programmable generic identifier acceptance filter has been introduced in order to reduce the CPU interrupt loading. The filter is programmable to operate in three different modes:

- Single identifier acceptance filter to be applied to the full 29 bits of the identifier and to the following bits of the CAN frame: RTR, IDE, SRR. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier.
- Double identifier acceptance filter to be applied to
  – the 11 bits of the identifier and the RTR bit of CAN 2.0A messages or
  – the 14 most significant bits of the identifier of CAN 2.0B messages.
- Quadruple identifier acceptance filter to be applied to the first 8 bits of the identifier. This mode implements four independent filters for the first 8 bit of a CAN 2.0A compliant standard identifier.

The Identifier Acceptance Registers (CIAR) defines the acceptable pattern of the standard or extended identifier (ID10 - ID0 or ID28 - ID0). Any of these bits can be marked ‘don’t care’ in the Identifier Mask Register (CIMR).

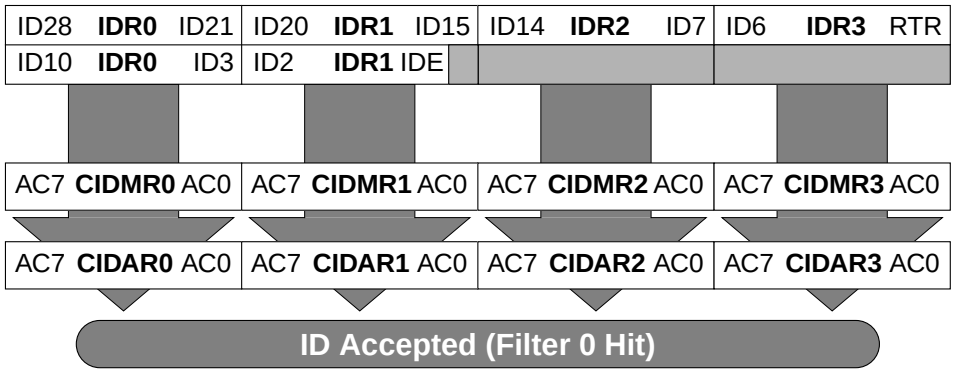


Figure 23-3 Single 32 bit maskable identifier acceptance filter

The background buffer RxBG will be copied into the foreground buffer RxFG and the RxF flag will be set only in case of an accepted identifier (an identifier acceptance filter hit). A hit will also cause a receiver interrupt if enabled.

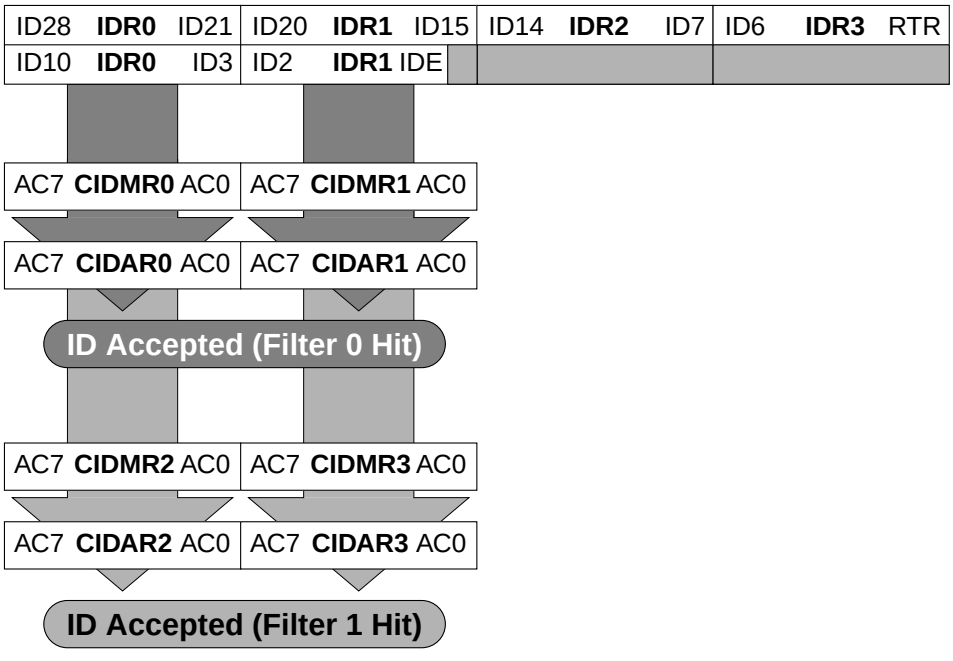


Figure 23-4 Dual 16 bit maskable acceptance filters

A filter hit is indicated to the application software by a set RXF (Receive Buffer Full Flag, see Section 23.12.6) and two bits in the Identifier Acceptance Control Register (see Section 23.12.10). These Identifier Hit Flags (IDHIT1-0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

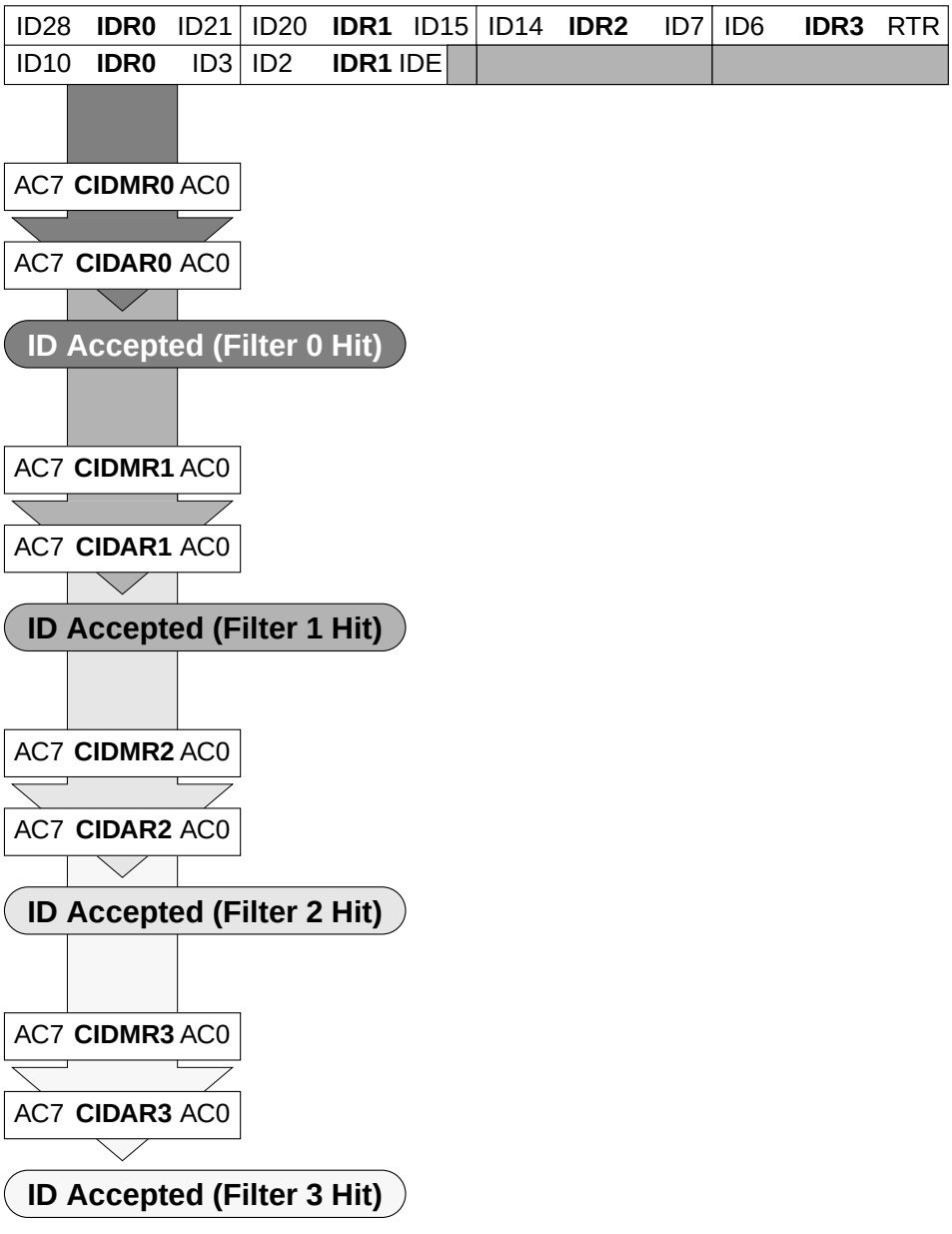


Figure 23-5 Quadruple 8 bit maskable acceptance filters

23.5 Interrupts

The msCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see Section 23.12.6 to Section 23.12.9):

- *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been successfully received and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wake-Up Interrupt*: An activity on the CAN bus occurred during msCAN08 internal sleep mode.
- *Error Interrupt*: An overrun, error or warning condition occurred. The Receiver Flag Register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in Section 23.3.2 has occurred.
  - *Receiver Warning*: The Receive Error Counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning*: The Transmit Error Counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive*: The Receive Error Counter has exceeded the Error Passive limit of 127 and msCAN08 has gone to Error Passive state.
  - *Transmitter Error Passive*: The Transmit Error Counter has exceeded the Error Passive limit of 127 and msCAN08 has gone to Error Passive state.
  - *Bus Off*: The Transmit Error Counter has exceeded 255 and msCAN08 has gone to Bus Off state.

23.5.1 Interrupt acknowledge

Interrupts are directly associated with one or more status flags in either the msCAN08 Receiver Flag Register (CRFLG) or the msCAN08 Transmitter Control Register (CTCR). Interrupts are pending as long as one of the corresponding flags is set. The flags in above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a “1” to the corresponding bit position. A flag can not be cleared if the respective condition still prevails.

CAUTION

Bit manipulation instructions (BSET) shall not be used to clear interrupt flags. The “OR” instruction is the appropriate way to clear selected flags.

23.5.2 Interrupt vectors

The msCAN08 supports four interrupt vectors as shown in Table 23-1. The vector addresses are dependent on the chip integration and to be defined. The relative interrupt priority is also integration dependent and to be defined.

Table 23-1 msCAN08 interrupt vectors

Function	Source	Local Mask	Global Mask
Wake-Up	WUPIF	WUPIE	I Bit
Error Interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

23.6 Protocol violation protection

The msCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters can not be written or otherwise manipulated.
- All registers which control the configuration of the msCAN08 can not be modified while the msCAN08 is on-line. The SFTRES bit in the msCAN08 Module Control Register (see Section 23.12.2) serves as a lock to protect the following registers:
  - msCAN08 Module Control Register 1 (CMCR1)
  - msCAN08 Bus Timing Register 0 and 1 (CBTR0, CBTR1)
  - msCAN08 Identifier Acceptance Control Register (CIDAC)
  - msCAN08 Identifier Acceptance Registers (CIDAR0-3)
  - msCAN08 Identifier Mask Registers (CIDMR0-3)
- The TxCAN pin is forced to Recessive if the CPU goes into STOP mode.

23.7 Low power modes

The WAIT and STOP instruction put the MCU in low power consumption stand-by mode.

23.7.1 msCAN08 internal sleep mode

The CPU can request the msCAN08 to enter the low-power mode by asserting the SLPRQ bit in the Module Configuration Register (see Figure 23-1). This causes the msCAN08 module internal clock to stop unless the module is active (i.e. receiving a message). The SLPAK bit indicates whether the msCAN08 successfully went into sleep mode. The application software should use this flag as a handshake indication for the request to go into sleep mode. If not set after the request, the msCAN08 is active and has not yet entered sleep mode. No wake-up interrupt will occur in that case.

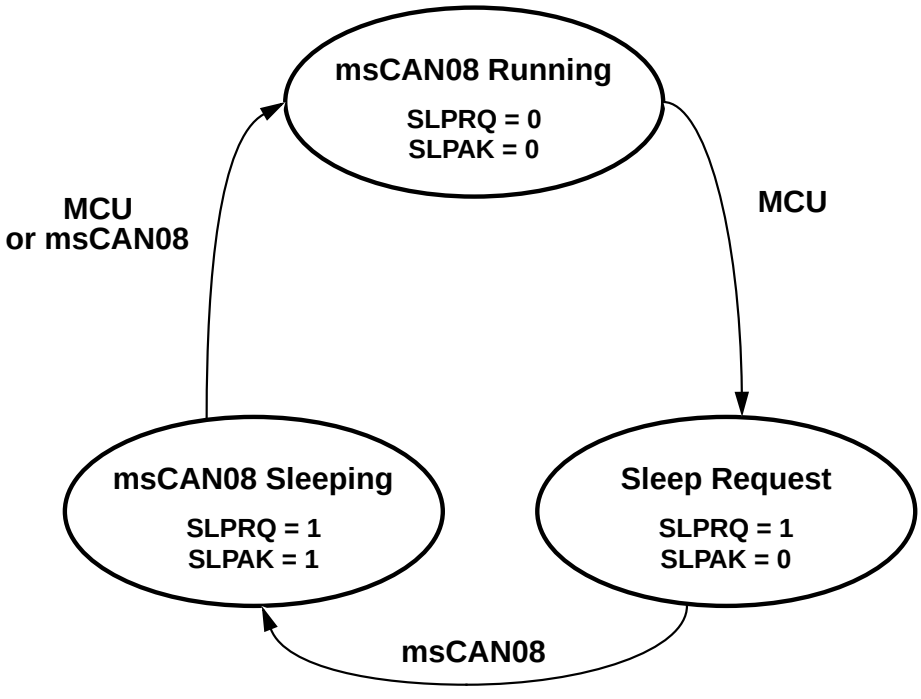


Figure 23-1. Sleep request/acknowledge cycle

When in sleep mode the msCAN08 stops its own clocks, leaving the MCU in normal run mode.

The msCAN08 will leave sleep mode (wake-up) when bus activity occurs or when the MCU clears the SLPRQ bit.

The TxCAN pin will stay in recessive state while the msCAN08 is in internal sleep mode.

The MCU can not clear the SLPRQ bit before the msCAN08 is in sleep mode (SLPAK = 1).

23.7.2 CPU WAIT mode

The msCAN08 module remains active during CPU WAIT mode. The msCAN08 will stay synchronized to the CAN bus and will generate enabled transmit, receive and error interrupts to the CPU. Any such interrupt will bring the MCU out of WAIT mode.

23.7.3 CPU STOP mode

A CPU STOP instruction will stop the crystal oscillator thus shutting down all system clocks. The user is responsible to take care that the msCAN08 is not active when the CPU goes into STOP mode. To protect the CAN bus system from fatal consequences of violations to above rule, the msCAN08 will drive the TxCAN pin into recessive state.

The recommended procedure is to bring the msCAN08 into sleep mode before the CPU STOP instruction is executed.

23.7.4 Programmable wake-up function

The msCAN08 can be programmed to apply a low-pass filter function to the RxCAN input line while in internal sleep mode (see control bit WUPM in **23.12.3 msCAN08 module control register (CMCR1)**). This feature can be used to protect the msCAN08 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

23.8 Timer link

The msCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the msCAN08 receiver engine receives also the frames being sent by itself, a timer signal will also be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip Timer Interface Module (TIM). Under the control of the Timer Link Enable (TLNKEN) bit in the CMCR0 will this signal be connected to the Timer n Channel m input(1).

After Timer n has been programmed to capture rising edge events it can be used to generate 16-bit time stamps which can be stored under software control with the received message.

1. The timer channel being used for the timer link is integration dependent.

23.9 Clock system

Figure 23-6 shows the structure of the msCAN08 clock generation circuitry and its interaction with the Clock Generation Module (CGM). With this flexible clocking scheme the msCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

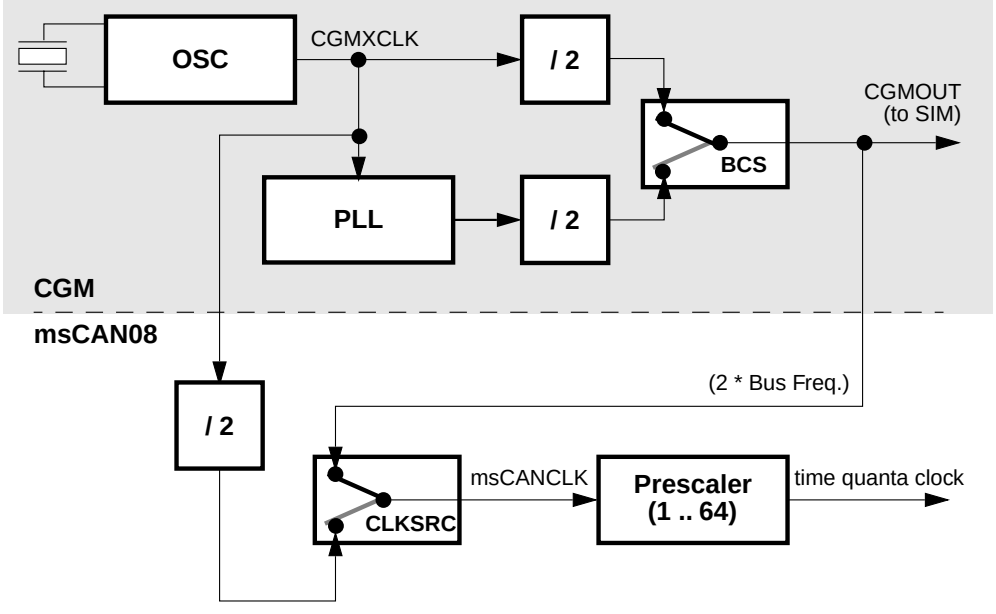


Figure 23-6 Clocking scheme

The Clock Source Flag (CLKSRC) in the msCAN08 Module Control Register (CMCR1) (see Section 23.12.3) defines whether the msCAN08 is connected to the output of the crystal oscillator or to the PLL output.

A programmable prescaler is used to generate from the msCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the msCAN08. A bit time is subdivided into three segments[1]:

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.

23

1. For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.

- Time segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

The Synchronization Jump Width can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

Above parameters can be set by programming the Bus Timing Registers (CBTR0-1, see Section 23.12.4 and Section 23.12.5).

It is the user's responsibility to make sure that his bit time settings are in compliance with the CAN standard. Figure 23-8 gives an overview on the CAN conforming segment settings and the related parameter values.

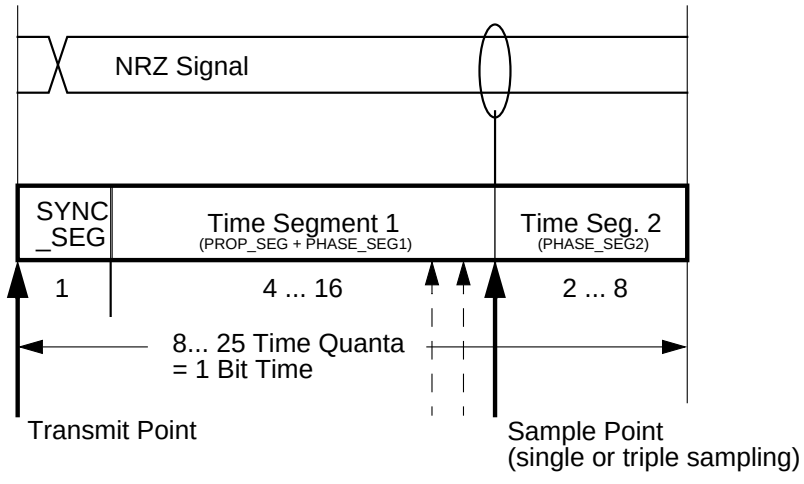


Figure 23-7 Segments within the bit time

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchron. Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

Figure 23-8 CAN standard compliant bit time segment settings

23.10 Memory map

The msCAN08 occupies 128 Byte in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128. The background receive buffer can only be read in test mode.

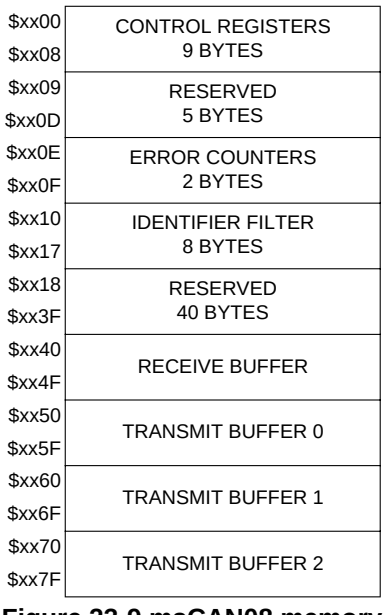


Figure 23-9 msCAN08 memory map

23.11 Programmer's model of message storage

The following section details the organisation of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification the receive and transmit message buffers have the same outline. Each message buffer allocates 16 byte in the memory map containing a 13 byte data structure. An additional Transmit Buffer Priority Register (TBPR) is defined for the transmit buffers.

Addr	Register Name
xxb0	Identifier Register 0
xxb1	Identifier Register 1
xxb2	Identifier Register 2
xxb3	Identifier Register 3
xxb4	Data Segment Register 0
xxb5	Data Segment Register 1
xxb6	Data Segment Register 2
xxb7	Data Segment Register 3
xxb8	Data Segment Register 4
xxb9	Data Segment Register 5
xxbA	Data Segment Register 6
xxbB	Data Segment Register 7
xxbC	Data Length Register
xxbD	Transmit Buffer Priority Register(1)
xxbE	unused
xxbF	unused

Figure 23-10 Message Buffer organisation

1. Not Applicable for Receive Buffers

23.11.1 Message Buffer outline

Figure 23-11 shows the common 13 byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 23-12. All bits of the 13 byte data structure are undefined out of reset.

23.11.2 Identifier registers (IDRn)

The identifiers consist of either 11 bits (ID10 – ID0) for the standard, or 29 bits (ID28 - ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR - Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

ADDR	REGISTER	R/W	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$xxb0	IDR0	R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
$xxb1	IDR1	R W	ID20	ID19	ID18	SRR (1)	IDE (1)	ID17	ID16	ID15
$xxb2	IDR2	R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
$xxb3	IDR3	R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
$xxb4	DSR0	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxb5	DSR1	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxb6	DSR2	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxb7	DSR3	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxb8	DSR4	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxb9	DSR5	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxbA	DSR6	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxbB	DSR7	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxbC	DLR	R W					DLC3	DLC2	DLC1	DLC0

Figure 23-11 Receive/Transmit Message Buffer extended identifier

ADDR	REGISTER	R/W	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$xxb0	IDR0	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
$xxb1	IDR1	R W	ID2	ID1	ID0	RTR	IDE(0)			
$xxb2	IDR2	R W								
$xxb3	IDR3	R W								

Figure 23-12 Standard identifier mapping

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer the flag indicates to the msCAN08 what type of identifier to send.

1 = Extended format (29 bit)
0 = Standard format (11 bit)

RTR — Remote transmission request

This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In case of a receive buffer it indicates the status of the received frame and allows to support the transmission of an answering frame in software. In case of a transmit buffer this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

23.11.3 Data length register (DLR)

This register keeps the data length field of the CAN frame.

DLC3 – DLC0 — Data length code bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 23-1 shows the effect of setting the DLC bits.

Table 23-1 Data length codes

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

23.11.4 Data segment registers (DSRn)

The eight data segment registers contain the data to be transmitted or being received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

23.11.5 Transmit buffer priority registers (TBPR)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TBPR	R	PRIO7	PRIO5	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
$xxbD	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-13 Transmit buffer priority register (TBPR)

PRIO7 - PRIO0— Local priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the msCAN08 and is defined to be highest for the smallest binary number. The msCAN08 implements the following internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the SOF (Start of Frame) is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case of more than one buffer having the same lowest priority the message buffer with the lower index number wins.

CAUTION

To ensure data integrity, no registers of the transmit buffers shall be written while the associated TXE flag is cleared. Also, no registers of the receive buffer shall be read while the RXF flag is cleared.

23.12 Programmer's model of control registers

23.12.1 Overview

The programmer's model has been laid out for maximum simplicity and efficiency. The following figure gives an overview on the control register block of the msCAN08:

ADDR	REGISTER	R/W	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$xx00	CMCR0	R	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		W								
$xx01	CMCR1	R	0	0	0	0	0	LOOPB	WUPM	CLKSRC
		W*								
$xx02	CBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		W*								
$xx03	CBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W*								
$xx04	CRFLG	R	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		W								
$xx05	CRIER	R	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		W								
$xx06	CTFLG	R	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		W								
$xx07	CTCR	R	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		W								
$xx08	CIDAC	R	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		W*								
$xx09-$xx0D	reserved	R								
		W								
$xx0E	CRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
$xx0F	CTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
$xx10	CIDAR0	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W*								
$xx11	CIDAR1	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W*								
$xx12	CIDAR2	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W*								
$xx13	CIDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W*								
$xx14	CIDMR0	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W*								
$xx15	CIDMR1	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W*								
$xx16	CIDMR2	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W*								
$xx17	CIDMR3	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W*								

Figure 23-14 msCAN08 Control Register Structure

23.12.2 msCAN08 module control register (CMCR0)

.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CMCR0	R	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
$xx00	W								
RESET		0	0	0	0	0	0	0	1

Figure 23-15 Module control register 0 (CMCR0)

SYNCH — Synchronized status

This bit indicates whether the msCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = msCAN08 is synchronized to the CAN bus
0 = msCAN08 is not synchronized to the CAN bus

TLNKEN - Timer enable

This flag is used to establish a link between the msCAN08 and the on-chip timer (see Section 23.8).

1 = The msCAN08 timer signal output is connected to the timer.
0 = No connection.

SLPAK — Sleep mode acknowledge

This flag indicates whether the msCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see Section 23.7.1).

1 = Sleep – The msCAN08 is in internal sleep mode.
0 = Wake-up – The msCAN08 will function normally.

SLPRQ — Sleep request, go to internal sleep mode

This flag allows to request the msCAN08 to go into an internal power-saving mode (see Section 23.7.1).

1 = Sleep – The msCAN08 will go into internal sleep mode if and as long as there is no activity on the bus.
0 = Wake-up – The msCAN08 will function normally. If SLPRQ is cleared by the CPU then the msCAN08 will wake up, but will not issue a wake-up interrupt.

SFTRES— Soft reset

When this bit is set by the CPU, the msCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers will go into the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0-3, CIDMR0-3 can only be written by the CPU when the msCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the msCAN08 will try to synchronize to the CAN bus: If the msCAN08 is not in bus-off state it will be synchronized after 11 recessive bits on the bus; if the msCAN08 is in bus-off state it continues to wait for 128 occurrences of 11 recessive bits.

1 = msCAN08 in soft reset state.

0 = Normal operation

23.12.3 msCAN08 module control register (CMCR1)

.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CMCR1	R	0	0	0	0	0	LOOPB	WUPM	CLKSRC
$xx01	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-16 Module control register 1 (CMCR1)

LOOPB - Loop back self test mode

When this bit is set the msCAN08 performs an internal loop back which can be used for self test operation: the bit stream output of the transmitter is fed back to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (1). Note that in this state the msCAN08 ignores the ACK bit to insure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

1 = Activate loop back self test mode

0 = Normal operation

WUPM - Wake-up mode

This flag defines whether the integrated low-pass filter is applied to protect the msCAN08 from spurious wake-ups (see Section 23.7.4).

1 = msCAN08 will wake up the CPU only in case of dominant pulse on the bus which has a length of at least approximately T_{wup}.

0 = msCAN08 will wake up the CPU after any recessive to dominant edge on the CAN bus.

CLKSRC - Clock source

This flag defines which clock source the msCAN08 module is driven from (see Section 23.9).

1 = THE msCAN08 clock source is CGMOUT (see Figure 23-6).

0 = The msCAN08 clock source is CGMXCLK/2 (see Figure 23-6).

NOTE

The CMCR1 register can only be written if the SFTRES bit in the msCAN08 Module Control Register is set

---

23.12.4 msCAN08 bus timing register 0 (CBTR0)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CBTR0	R	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
$xx02	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-17 Bus timing register 0

SJW1, SJW0 — Synchronization jump width

The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 23-2).

Table 23-2 Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

BRP5 – BRP0 — Baud Rate Prescaler

These bits determine the time quanta (Tq) clock, which is used to build up the individual bit timing, according to Table 23-3.

Table 23-3 Baud rate prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	1	1	64

NOTE

The CBTR0 register can only be written if the SFTRES bit in the msCAN08 Module Control Register is set.

23.12.5 msCAN08 bus timing register 1 (CBTR1)

.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
$xx03	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-18 Bus timing register 1

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. If set three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit.
0 = One sample per bit.

TSEG22 – TSEG10 — Time segment

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

Table 23-4 Time segment syntax

SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit point	A node in transmit mode will transfer a new value to the CAN bus at this point.
Sample point	A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample.

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table 23-7

.

Table 23-5 Time segment values

TSEG 13	TSEG 12	TSEG 11	TSEG 10	Time segment 1
0	0	0	0	1 Tq clock cycle
0	0	0	1	2 Tq clock cycles
0	0	1	0	3 Tq clock cycles
0	0	1	1	4 Tq clock cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 Tq clock cycles

TSEG 22	TSEG 21	TSEG 20	Time segment 2
0	0	0	1 Tq clock cycle
0	0	1	2 Tq clock cycles
.	.	.	.
.	.	.	.
1	1	1	8 Tq clock cycles

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown above).

NOTE

The CBTR1 register can only be written if the SFTRES bit in the msCAN08 Module Control Register is set

---

23.12.6 msCAN08 receiver flag register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can only be cleared when the condition which caused the setting is no more valid. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CRFLG	R	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
$xx04	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-19 Receiver flag register

WUPIF — Wake-up interrupt flag

If the msCAN08 detects bus activity whilst it is asleep, it clears the SLPAK bit in the CMCR0 register; the WUPIF bit will then be set. If not masked, a Wake-Up interrupt is pending while this flag is set.

1 = msCAN08 has detected activity on the bus and requested wake-up.
0 = No wake-up activity has been observed while in sleep mode.

RWRNIF — Receiver warning interrupt flag

This bit will be set when the msCAN08 went into warning status due to the Receive Error counter being in the range of 96 to 127. If not masked, an Error interrupt is pending while this flag is set.

1 = msCAN08 went into receiver warning status.
0 = No receiver warning status has been reached.

TWRNIF — Transmitter warning interrupt flag

This bit will be set when the msCAN08 went into warning status due to the Transmit Error counter being in the range of 96 to 127. If not masked, an Error interrupt is pending while this flag is set.

1 = msCAN08 went into transmitter warning status.
0 = No transmitter warning status has been reached.

RERRIF — Receiver error Passive Interrupt Flag

This bit will be set when the msCAN08 went into error passive status due to the Receive Error counter exceeded 127. If not masked, an Error interrupt is pending while this flag is set.

1 = msCAN08 went into receiver error passive status.
0 = No receiver error passive status has been reached.

TERRIF — Transmitter Error Passive Interrupt Flag

This bit will be set when the msCAN08 went into error passive status due to the Transmit Error counter exceeded 127. If not masked, an Error interrupt is pending while this flag is set.

1 = msCAN08 went into transmitter error passive status.
0 = No transmitter error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This bit will be set when the msCAN08 went into bus-off status, due to the Transmit Error counter exceeded 255. If not masked, an Error interrupt is pending while this flag is set.

1 = msCAN08 went into bus-off status.
0 = No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This bit will be set when a data overrun condition occurred. If not masked, an Error interrupt is pending while this flag is set.

1 = A data overrun has been detected.
0 = No data overrun has occurred.

RXF — Receive Buffer Full

The RXF flag is set by the msCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be handshaken to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. In that case the msCAN08 will signal an overload condition. If not masked, a Receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.
0 = The receive buffer is released (not full).

23.12.7 msCAN08 Receiver Interrupt Enable Register (CRIER)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CRIER	R	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
$xx05	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-20 Receiver Interrupt Enable Register

WUPIE — Wake-up Interrupt Enable
1 = A wake-up event will result in a wake-up interrupt.
0 = No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable
1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable
1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable
1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable
1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

BOFFIE — Bus-Off Interrupt Enable
1 = A bus-off event will result in an error interrupt.
0 = No interrupt will be generated from this event.

OVRIE — Overrun Interrupt Enable
1 = An overrun event will result in an error interrupt.
0 = No interrupt will be generated from this event.

RXFIE — Receiver Full Interrupt Enable
1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
0 = No interrupt will be generated from this event.

23.12.8 msCAN08 Transmitter Flag Register (CTFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CTCR register. A hard or soft reset will clear the register.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CTFLG	R	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
$xx06	W								
RESET		0	0	0	0	0	1	1	1

Figure 23-21 Transmitter Flag Register

ABTAK2 - ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent in the meantime. The flag is reset implicitly whenever the associated TXE flag is set to 0.

1 = The message has been aborted.
0 = The massage has not been aborted, thus has been sent out.

TXE2 - TXE0 —Transmitter Buffer Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The msCAN08 will set the flag after the message has been sent successfully. The flag will also be set by the msCAN08 when the transmission request was successfully aborted due to a pending abort request (Section 23.12.9). If not masked, a Transmit interrupt is pending while this flag is set.
A reset of this flag will also reset the Abort Acknowledge (ABTAK, see above) and the Abort Request (ABTRQ, see Section 23.12.9) flags of the particular buffer.

1 = The associated message buffer is empty (not scheduled).
0 = The associated message buffer is full (loaded with a message due for transmission).

23.12.9 msCAN08 Transmitter Control Register (CTCR)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CTCR	R	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
$xx07	W								
RESET		0	0	0	0	0	0	0	0

Table 23-6 Transmitter Control Register

ABTRQ2 - ABTRQ0 — Abort Request

The CPU sets this bit to request that an already scheduled message buffer (TXE = 0) shall be aborted. The msCAN08 will grant the request when the message is not already under transmission. When a message is aborted the associated

TXE and the Abort Acknowledge flag (ABTAK, see Section 23.12.8) will be set and an TXE interrupt will occur if enabled. The CPU can not reset ABTRQx. ABTRQx is reset implicitly whenever the associated TXE flag is set.

1 = Abort request pending.
0 = No abort request.

TXEIE2 - TXEIE0 — Transmitter Empty Interrupt Enable

1 = A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.
0 = No interrupt will be generated from this event.

23.12.10 msCAN08 Identifier Acceptance Control Register (CIDAC)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CIDAC	R	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
$xx08	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-22 Identifier Acceptance Control Register

IDAM1- IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organization (see Section 23.4). Table 23-7 summarizes the different settings. In “Filter Closed” mode no messages will be accepted such that the foreground buffer will never be reloaded.

Table 23-7 Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Single 32 bit Acceptance Filter
0	1	Two 16 bit Acceptance Filter
1	0	Four 8 bit Acceptance Filters
1	1	Filter Closed

IDHIT1- IDHIT0— Identifier Acceptance Hit Indicator

The msCAN08 sets these flags to indicate an identifier acceptance hit (see Section 23.4). Table 23-7 summarizes the different settings.

Table 23-8 Identifier Acceptance Hit Indication

IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	Filter 0 Hit
0	1	Filter 1 Hit
1	0	Filter 2 Hit
1	1	Filter 3 Hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer the indicators are updated as well.

NOTE

The CIDAC register can only be written if the SFTRES bit in the msCAN08 Module Control Register is set.

23.12.11 msCAN08 Receive Error Counter (CRXERR)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CRXERR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
$xx0E	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-23 Receive Error Counter

This register reflects the status of the msCAN08 receive error counter. The register is read only.

23.12.12 msCAN08 Transmit Error Counter (CTXERR)

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CTXERR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
$xx0F	W								
RESET		0	0	0	0	0	0	0	0

Figure 23-24 Transmit Error Counter

This register reflects the status of the msCAN08 transmit error counter. The register is read only.

For both error counters there is no hardware synchronization between the write accesses to those registers from the msCAN08 side and the read accesses by the

CPU. It is the user's responsibility to verify that a stable value has been read by executing a second validation read and comparing the two values.

23.12.13 msCAN08 Identifier Acceptance Registers (CIDAR0-3)

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the msCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers all four acceptance and mask registers are applied. For standard identifiers only the first two (IDAR0, IDAR1) are applied. In the latter case it is required to program the mask register CIDMR1 in the three last bits (AC2 - AC0) to "don't care".

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CIDAR0 $xx10	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR1 $xx11	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR2 $xx12	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
CIDAR3 $xx13	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
	W								
RESET		U	U	U	U	U	U	U	U

Figure 23-25 Identifier Acceptance Registers

AC7 – AC0 — Acceptance Code Bits

AC7 – AC0 comprise a user defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE

The CIDAR0-3 registers can only be written if the SFTRES bit in the msCAN08 Module Control Register is set

---

23.12.14 msCAN08 Identifier Mask Registers (CIDMR0-3)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CIDMR0 $xx14	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR1 $xx15	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR2 $xx16	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
CIDMR3 $xx17	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
	W								
RESET		U	U	U	U	U	U	U	U

Figure 23-26 Identifier Mask Registers

AM7 – AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit, before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

Bit description:

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

NOTE

The CIDMR0-3 registers can only be written if the SFTRES bit in the msCAN08 Module Control Register is set

---

APPENDIX A
Devices Similar to the HC08AZ32

A.1 HC08AZ0

The HC08AZ0 is a device similar to the HC08AZ32, but with an External Bus Interface (EBI) instead of on-chip user ROM. The entire HC08AZ32 Technical Summary applies to the HC08AZ0, with the exceptions outlined in this appendix.

A.1.1 Features

- External Bus Interface (EBI)
- No on-chip user ROM or mask options
- Available in 100-pin TQFP (Thin Quad Flat Pack) package

A.1.2 MCU block diagram

A block diagram of the HC08AZ32 is shown in Figure A-1.

A.1.3 Pin assignments

The pin assignments for the HC08AZ32 are shown in Figure A-2.

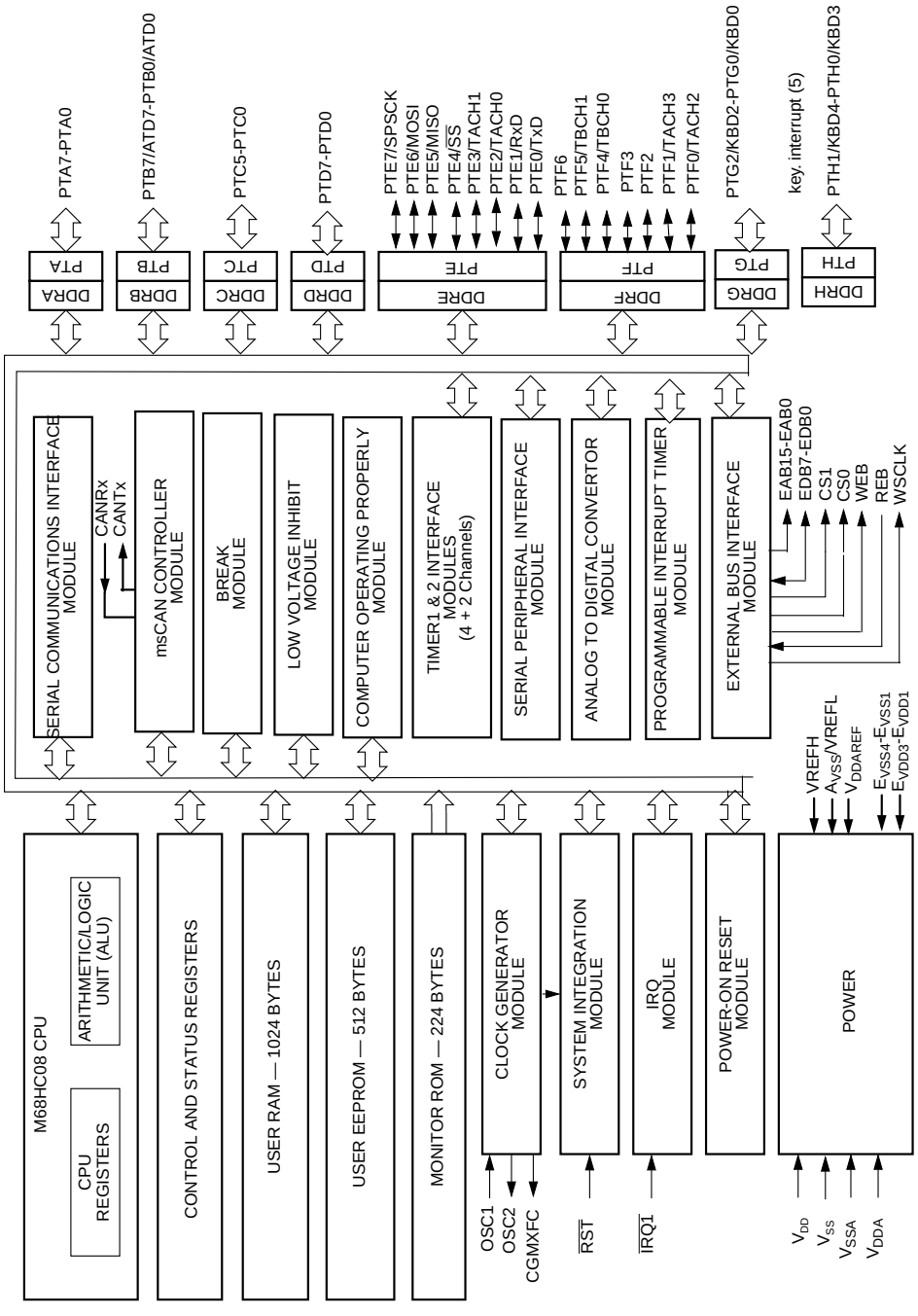


Figure A-1. MCU Block Diagram

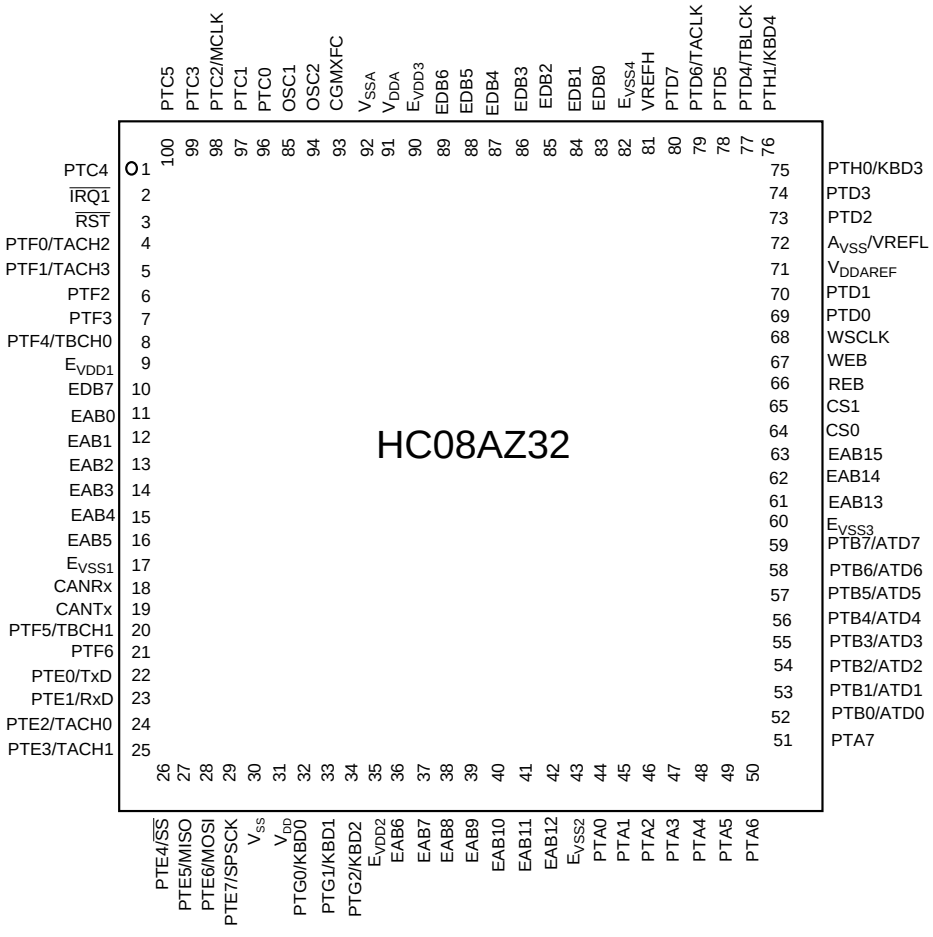


Figure A-2 100TQFP pin assignments (Top View)

A.1.4 Pin descriptions

A.1.4.1 Power supply pins(E_{VDD3}/E_{VDD1} and E_{VSS4}/E_{VSS1})

The E_{VDD} and E_{VSS} pins are for the sole use of the I/O pins. This will help reduce the effect of the noise induced into the V_{SS} power supply.

A.1.4.2 External data pins (EDB7-EDB0)

EDB7-EDB0 are the bidirectional data lines for connection to external peripherals.

A

A.1.4.3 External address pIns (EAB15-EAB0)

EAB15-EAB0 are the address lines for connection to external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.1.4.4 Write control pin (WEB)

WEB is the write control signal for external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.1.4.5 Read control pin (REB)

REB is the read control signal for external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.1.4.6 WAIT states clock pin (WSCLK)

WSCLK is the External WAIT State control signal. See **A.1.14 External Bus Interface (EBI) Module**.

A.1.4.7 External chip-select pins (CS1,CS0)

CS1 and CS0 are the chip-select lines for connection to external peripherals. See **A.1.14 External Bus Interface (EBI) Module**. The external pins are summarized in Table A-1 and the clock sources are shown in Table A-2.

Table A-1 External pins summary

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
EAB15-EAB0	External address bus	Output	NA	Output
EDB7-EDB0	External data bus	Dual state	NO	Input (Hi-Z)
REB	External read enable	Output	NA	Output
WEB	External write enable	Output	NA	Output
WSCLK	WAIT state clock	Output	NA	Hi-Z
CS1	Chip-select 1	Output	NA	Output
CS0	Chip-select 0	Output	NA	Output

Table A-2 Clock source summary

Module	Clock source
EBI	Bus clock

A

A.1.5 Memory Section

The HC08AZ0 operates in Expanded mode, where all memory space not occupied by internal peripherals or memory is available externally through the External Bus Interface (EBI).

A.1.5.1 Memory Map

Figure A-3 shows the memory map of the HC08AZ32.

A.1.6 ROM security

The ROM security feature is disabled.

A.1.7 LVI reset

The reset signal from the LVI module is enabled.

A.1.8 LVI power

The LVI module power is enabled.

A.1.9 STOP mode recovery delay

STOP mode recovery after 4096 CGMXCLK cycles.

A.1.10 STOP instruction

The STOP instruction is enabled.

A.1.11 COP module

The COP module is enabled.

A.1.12 Reset vector source

The reset vector source on the HC08AZ32 is external.

A.1.13 EEPROM security

The EEPROM security function on the HC08AZ32 is enabled. This can be used to prevent program/erase access to locations $08F0–$08FF of the EEPROM array and also to the EEACR/EENVR configuration registers. See **5.3.7 HC08AZ32 EEPROM security**.

NOTE

The settings described in Section A.1.6 to Section A.1.13 are subject to change.

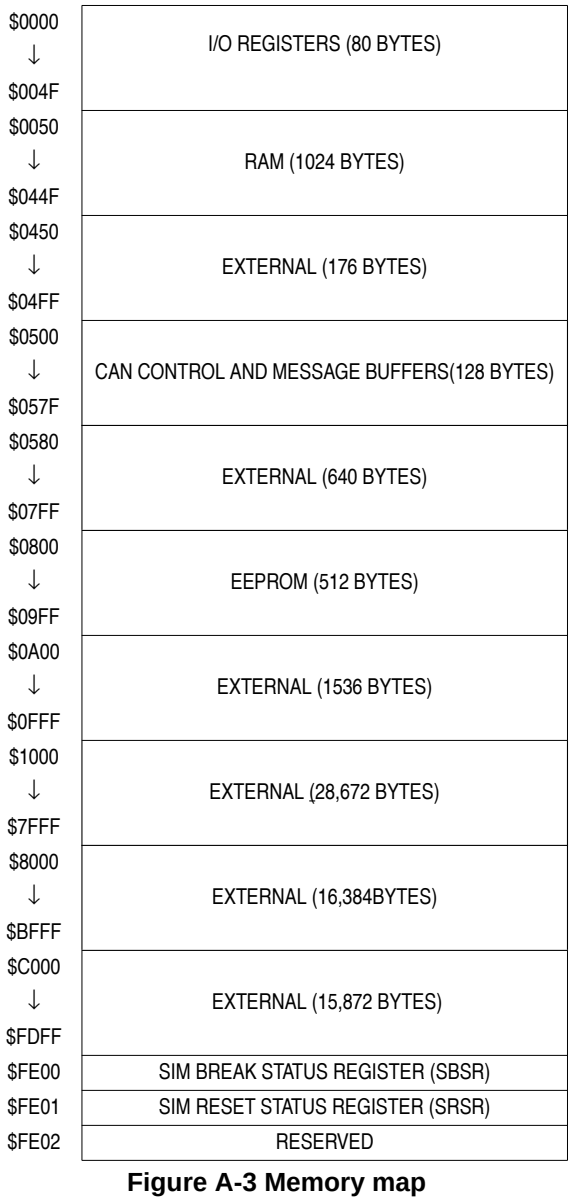


Figure A-3 Memory map

A

$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04	RESERVED
$FE05	RESERVED
$FE06	UNIMPLEMENTED
$FE07	RESERVED
$FE08	RESERVED
$FE09	RESERVED
$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	EXTERNAL (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	EXTERNAL (192 BYTES)
$FFC0 ↓ $FFCF	EXTERNAL (16 BYTES)
$FFD0 ↓ $FFFF	EXTERNAL VECTORS (48 BYTES)

Figure A-3 Memory map

A.1.14 External Bus Interface (EBI) Module

A.1.14.1 Introduction

This section describes the External Bus Interface Module (EBI) specification for the HC08AZ0 MCU. This module handles the transfer of information between the MCU

and the external address space. The external bus provides up to 16 address lines, 8 data lines, 2 chip-selects and 3 control signals to the external devices.

A.1.14.2 Features

- Up to 64K byte of Address Space.
- Low Noise or High Performance Modes of operation.
- 2 Pre-determined Chip-Select Lines (CS1 and CS0).
- Separate Read and Write Enable Signals
- 0,1,2 or 3 WAIT States for Slow Memory Access associated with CS1
- Up to 7 WAIT States for Slow Device Access associated with CS0.

Figure A-4 shows the structure of the EBI.

The EBI has two basic modes of operation; Low Noise or High Performance. In Low Noise mode the Address and Data lines are only driven out during external accesses to reduce RF emissions

In High Performance mode the EBI operates at its fastest. This mode of operation requires the external address and data signals from the HC08AZ32 **before** they are stable.

The chip-select regions are controlled by software providing the user with a choice of four combinations.The chip-selects have a control register which allows the chip selects to be enabled, their polarity selected and for the appropriate number of WAIT States to be defined for CS0 and CS1.

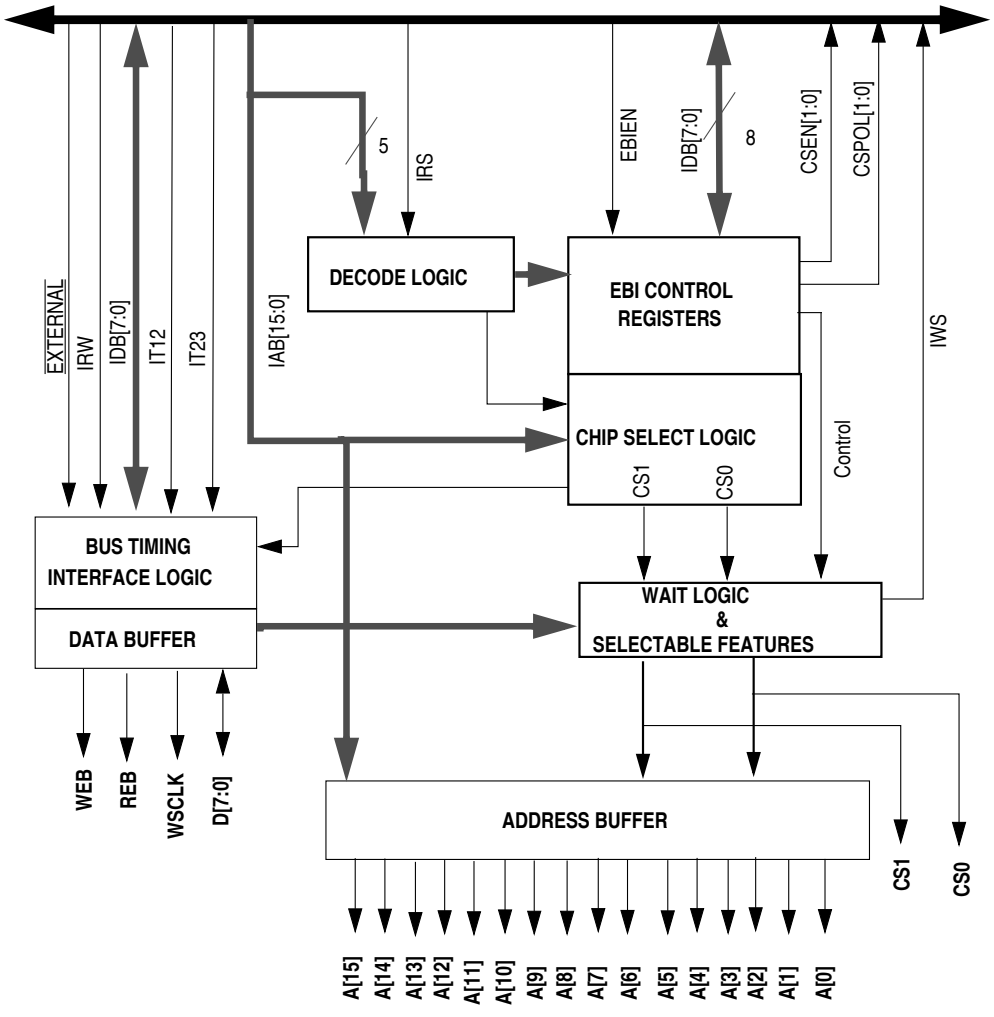


Figure A-4 EBI Block Diagram

A.1.14.3 Module I/O signal descriptions

Below is a description of the I/O signals shown in Figure A-4. The External Bus Interface module is connected to the HC08 Bus (IBUS) with data bus IDB[7:0], address bits IAB[15:0] and IRW, IWS for control. It is connected to the external devices with Address Bus A[15:0], Data Bus D[7:0], and Control Bits WEB, REB, CS1 and CS0. In addition WSCLK is provided for cycle-by-cycle external WAIT state selection.

A.1.14.3.1Internal Address - IAB[15:0]

The internal address bits are an input to the EBI module.

A.1.14.3.2Internal data - IDB[7:0]

The internal data bits are bidirectional signals to the EBI module. They are used to transfer data in and out of the EBI via the IBUS. The direction of the data flow is controlled by the IRW signal. When IRW is high (read mode), the data bus is an output of the EBI and the data in D[7:0] is transferred to IDB[7:0] (the internal data bus). When IRW is low (write mode), the data bus is an input to the EBI and the data in IDB[7:0] is transferred to D[7:0] (the external data bus). An exception to this occurs when the IRV bit is set. In that mode, when internal accesses occur, data on IDB[7:0] is transferred out on D[7:0].

A.1.14.3.3Internal read/write - IRW

The internal read/write bit is an input control signal to the EBI module. It allows the CPU (or DMA) to read and write to the external devices. With the proper interface logic, the IRW generates the external read signal REB and external write signal WEB.

A.1.14.3.4External access (External)

The External signal identifies those cycles that are not internal to the part. This is used to keep the EBI outputs quiet for internal accesses in Low Noise Mode.

A.1.14.3.5EBIEN - EBI enable

The EBI enable signal is provided by physical Mask Option. It is used to disable the EBI and therefore reduce power and RF emissions in non-expanded mode. (64QFP package).

A.1.14.3.6Internal WAIT state - IWS

The internal WAIT state signal is an output from the EBI module. It is connected to the IMREQB (in the IBUS), which suspends the CPU state (but not the internal clocks). The IMREQB signal delay is software controlled for CSI and either software or hardware controlled for CS0.This signal will be disabled for internal accesses.

A.1.14.3.7External address bus - A[15:0]

The external address bus are the output address signals from the EBI module. They provide addressing information to the external devices. In Low Noise mode

the values on these lines remain at the last driven state between external accesses.

A.1.14.3.8Chip-selects - CS1, CS0

The chip-selects are output control signals from the EBI module. They enable external devices at their programmed addresses.

A.1.14.3.9External data bus - DB[7:0]

The external data bus are three-state bidirectional data signals to the EBI module. These signals provide the data path between the MCU and all external devices. In Low Noise mode the values on these lines remain at the last driven state between external accesses.

A.1.14.3.10External read - REB

The external read is an output control signal from the EBI module. When REB is asserted, a read cycle starts and data is transferred from an external device to the internal data bus. This signal is asserted only for external accesses, or when the IRV bit is set.

A.1.14.3.11External write - WEB

The external write is an output control signal to the EBI module. When WEB is asserted, a write cycle starts and data is transferred to an external device from the internal data bus. This signal is asserted only for external accesses or when the IRV bit is set.

A.1.14.3.12WAIT state clock - WSCLK

The WAIT state clock is used to insert the correct number of WAIT states into each bus cycle when the ‘external WAIT states’ operating mode is selected for CS0.

A.1.14.4 Functional description

The user has two optional chip-select lines to define areas within the 64K byte address range to be occupied by external devices. The range of each address chip-select region can be modified by software. The software option provides the user with a choice of regions as shown in Figure A-5.

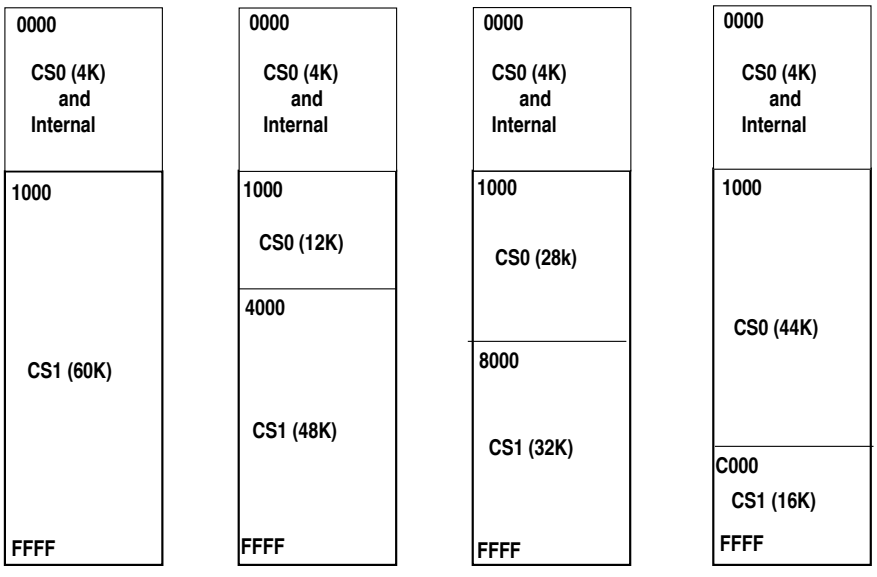


Figure A-5 Software controlled chip-select combinations

CS1 is intended for program space as it always includes the vector space. CS0 may be used for any purpose and is in effect equivalent to $\overline{CS1}$. Extending CS0 address space below $1000 allows the user to place external devices (e.g. I/O) in unused address locations below $1000 without the use of an external decoder (the CS0 pin would otherwise be redundant). Internal accesses will always have priority.

When a chip-select is enabled, it is active for all memories and I/O cycles within its defined (external) area. Each chip-select has control bits for enabling, polarity setting and for inserting the correct number of WAIT states in each bus cycle. Out of reset CS0 and CS1 are configured with the maximum number (3) of software controlled WAIT states.

All bus timing interface signals are handled by the bus timing interface logic that generates the proper signals for reads and writes required to interface the internal and external buses.

A.1.14.5 Externally controlled WAIT states

The EBI generates an IWS signal which can be controlled either internally or externally for CS0. The external option allows the user to further decode the CS0 address space into smaller address ranges for multiple external devices, and assign a different number of WAIT states to each address range. The number of WAIT states associated with CS1 address space is determined internally by the CS1WS 1:0 bits.

During T4 the HC08 data bus is not driven and during this clock phase the number of WAIT states for the cycle in progress is determined. When CS0 is asserted, the value on the External Data bus at the end of T4 is used to determine the number of WAIT States according to Table A-3 This WAIT state value is encoded in the first 3 bits of the data bus, D2:0. This mode of operation is selected by enabling the function in the EBI control register and by enabling the WSCLK pin according to Table A-4 The WAIT state value on the data bus is only latched when CS0 is asserted. Therefore, it is not required that the bus be driven during T4 when accessing addresses outside the CS0 range (i.e. when CS1 is asserted).

The WAIT state value driven onto the External Data bus may be derived directly from the address lines or indirectly from a decoded chip select signal(s). In all cases, the WAIT state value must only be allowed to drive the External Data bus during the period WSCLK is asserted by using a tri-statable buffer (e.g. 74AC240/244, 74AC367A/368A, 74AC125). See examples later in this section.

External Databus Value D2 - D0	Number of WAIT States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Table A-3 Data bus values corresponding to number of WAIT states

The pin WSCLK provides the T4 signal to synchronize driving the WAIT state value onto the External Data lines. Table A-4 shows the options available for the WSCLK pin. The WSCLK can also be disabled.

When external WAIT-state decoding is enabled, the low RF emission data bus freeze function is disabled for data bus lines D2:0. The address bus freeze function remains unaffected.

WSCLK1	WSCLK0	WSCLK Pin Function
0	0	Disabled, tri-state
0	1	$\overline{T4}$ + CS0, push/pull
1	0	T4, push/pull
1	1	$\overline{T4}$, push/pull

Table A-4 WSCLK pin function

CS0 used in the WSCLK pin functions is active low, irrespective of the state of the CS0 pin polarity bit. CS0 and WSCLK are not asserted during internal access bus cycles. The term <$\overline{T4}$ + CS0> is therefore an active low signal.

Examples of external WAIT state selection are shown in Figure A-6, Figure A-7 and Figure A-8

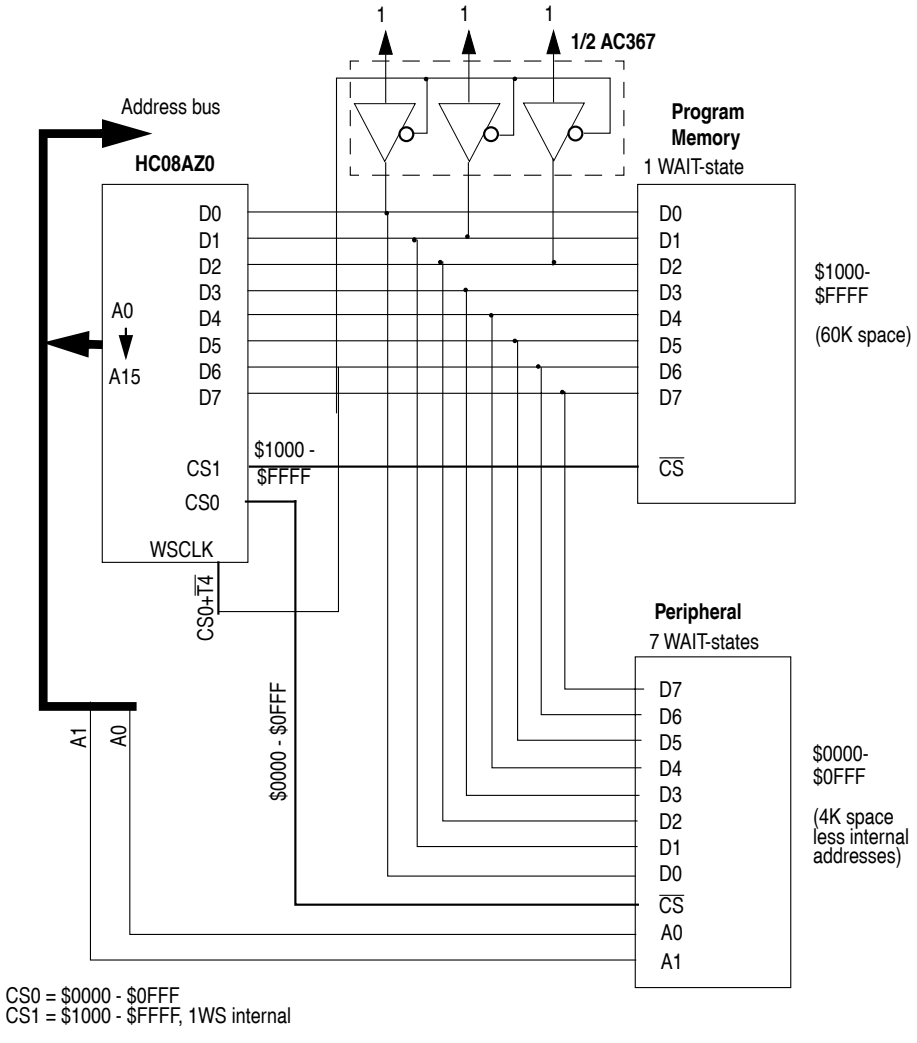


Figure A-6 Defining WAIT states for Ext addresses below $1000

In Figure A-6, the user wants to maximize the address space allocated to the program memory but requires to decode a slow external peripheral with a 4byte address space. To avoid driving the data bus unnecessarily, WSCLK is programmed to generate CS0+$\overline{T4}$ which will only go low during T4 whenever the address is within the CSo range as defined by CSC[1:0]. The peripheral will be multiply mapped within this address space. Whenever CS0 is asserted and the CS0 WAIT state control is configured for external control (C0WS =1), internal logic

A

will direct the WAIT state generator to use the data bus as the source of the number of WAIT states to be inserted. In this case, the value $111 will be driven onto D2:0 during T4 which will instruct the WAIT state generator to insert 7 WAIT states (equivalent to a 1uS bus cycle for an 8MHz bus clock). The number of WAIT states for CS1 is selected internally based on the contents of CS1WS1:0 bits and may between 0 and 3 bus cycles.

In Figure A-7, the application requires different WAIT states for program memory, RAM and peripheral bus cycles. One of the chip selects must therefore be subdivided into two address spaces, each with a different number of WAIT states associated with it. A simple decode of the upper 2 address lines provides an address range of $0000-$1FFF for the peripheral and drives External Data bus line D1 high during T4 when within that address range, thus requesting 2 WAIT states. In this implementation, the peripheral map is duplicated with the address range $0000 to $0FFF as well as the required $1000 to $1FFF. In order to prevent internal accesses effecting the peripheral, CS0 is used as a further enable to both devices within this address space. CS0 is not asserted for internal accesses within its address space.

WSCLK is programmed to generate CS0+$\overline{T4}$ which will go low during T4 whenever the address is within the CS0 range as defined by CSC[1:0]. The number of WAIT states for CS1 is selected internally based on the contents of CS1WS1:0 bits and may between 0 and 3 bus cycles.

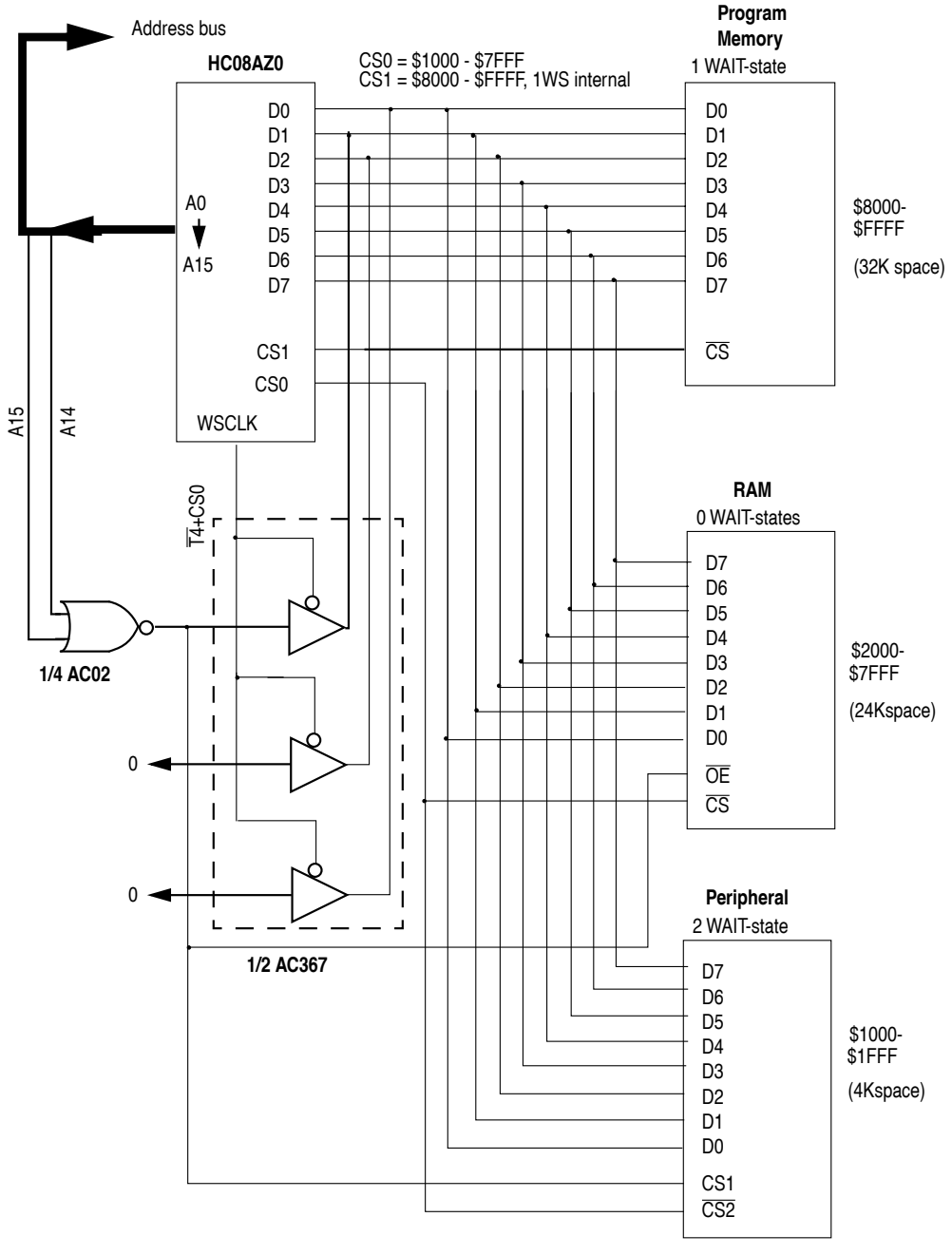


Figure A-7 Simple address space decode with different WAIT states

A

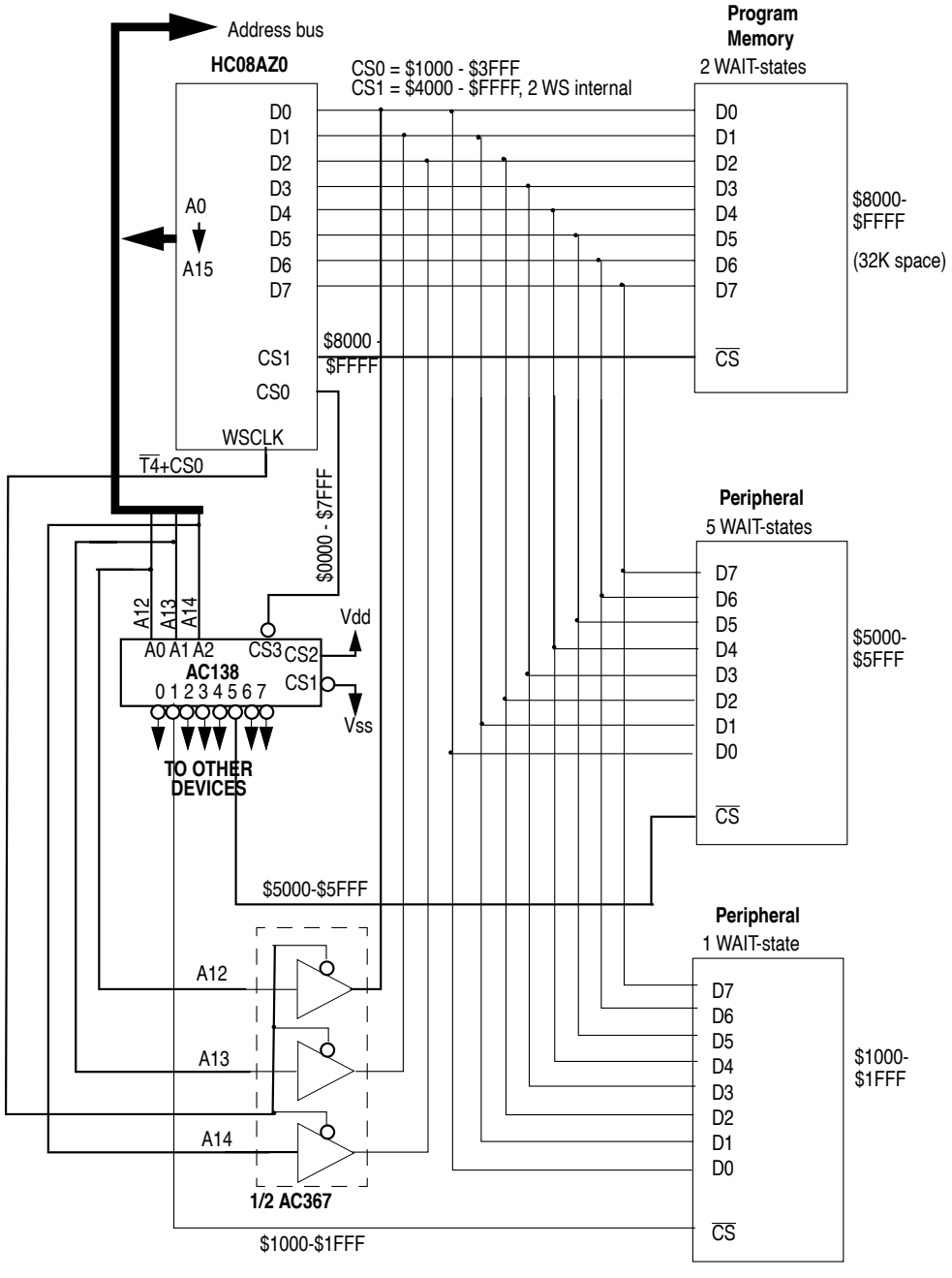


Figure A-8 WAIT state selection for multiple I/O peripherals

In Figure A-8, the application has several external peripheral devices which are decoded into the CS0 address space using an external 3-to-8 decoder (e.g. 74AC138). The addition of a tri-stateable buffer (e.g. 74AC367, hex buffer) represents the hardware overhead to provide a unique number of WAIT states for each decoded address space. Decoder outputs provide the peripheral chip selects. In this implementation, the same addresses used by the decoder are driven back onto the data bus during T4. Therefore, as the address increments, so does the number of WAIT states assigned to each address space. WSCLK is configured to drive CS0+$\overline{T4}$, which drives the buffer enables during T4 when the address is within the CS0 range.

Choosing to drive different addresses onto the data bus during T4 allow the user to provide a different mix of WAIT states throughout the decoded CS0 address space. Alternatively, the 3-8 line decoder could be replaced with a PAL which would provide for a more complex decode and assignment of WAIT states. The number of WAIT states for CS1 is selected internally based on the contents of CS1WS1:0 bits and may between 0 and 3 bus cycles.

A.1.14.6 EBI control registers

The following I/O registers control and monitor operation of the EBI:-

- EBI Control Register (EBIC)
- Chip-Select Control Register (EBICS)

A.1.14.6.13EBI control register

		Bit 7	6	5	4	3	2	1	0
EBIC $003B	Read:	0	IRV	MODE	C0WS	WSCLK1	WSCLK0	CSC1	CSC0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure A-9 EBI control register (EBIC)

IRVIRV - Internal read visibility bit

This function is included for easy-debug of the customer application.

1 = The REB and WEB are active during IRV, to allow creation of an ECLK. Enabled chip selects are active as well, and all internal bus activity is externally visible.

0 = In normal user operation IRV should be off to prevent possible bus contention.

MODE

MODE- EBI operating mode
1 = Low Noise
0 = High Performance

C0WS

C0WS- Chip select 0 WAIT state control
1 = Externally Controlled
0 = Internally Controlled.

WSCLK1:0

WSCLK1:0 - WAIT state clock select

These bits control the WSCLK operating mode according to Table A-5

WSCLK1	WSCLK0	WSCLK Pin Function
0	0	Disabled, Tri-State
0	1	$\overline{T4}$ + CS0, Open Drain
1	0	T4, Push/Pull
1	1	$\overline{T4}$, Push/Pull

Table A-5 WSCLK pin function

A.1.14.7 CSC1:0

CSC1:0 - Chip-Select Combination

These bits control the chip-select combination according to table Table A-6

CSC1	CSC0	Chip-Select 1	Chip-Select 0
0	0	$1000 - $FFFF	$0000 - $0FFF
0	1	$4000 - $FFFF	$0000 - $3FFF
1	0	$8000 - $FFFF	$0000 - $7FFF
1	1	$C000 - $FFFF	$0000 - $BFFF

Table A-6 Chip-select combinations

A.1.14.8 EBI Chip-Select Register

		Bit 7	6	5	4	3	2	1	Bit 0
EBICS $003C	Read:	CS1WS1	CS1WS0	CS1POL	CS1EN	CS0WS1	CS0WS0	CS0POL	CS0EN
	Write:								
	Reset:	1	1	0	1	1	1	0	1

= Unimplemented

Figure A-10 EBI control register (EBICS)

CS1WS1:0

These bits control the number of WAIT States for Chip-Select 1 during external accesses when the WAIT states are **internally** controlled. WAIT states are inserted according to Table A-7

CS1WS1:0	Number of WAIT States
00	0
01	1
10	2
11	3

Table A-7 WAIT states for external accesses

CS1EN Chip-select enable

This bit controls whether the Chip-Select is active.
0 = Chip-Select disabled.
1 = Chip-Select enabled

CS1POL Chip-select polarity

This bit controls the polarity of the Chip-Select line.
0 = Chip-Select Active LOW
1 = Chip-Select Active HIGH

CS0WS1:0

These bits control the number of WAIT States for Chip-Select 0 during external accesses according to Table A-7

CS0WS1:0	Number of WAIT States
00	0
01	1
10	2
11	3

Table A-8 WAIT states for external accesses

CS0EN Chip-select enable

This bit controls whether the Chip-Select is active.
0 = Chip-Select disabled.
1 = Chip-Select enabled

CS0POL Chip-select polarity

This bit controls the polarity of the Chip-Select line.
0 = Chip-Select Active LOW
1 = Chip-Select Active HIGH

A.1.14.9 Electrical information

The following figures show the Read and Write timings for both High Performance and Low Noise operating modes.

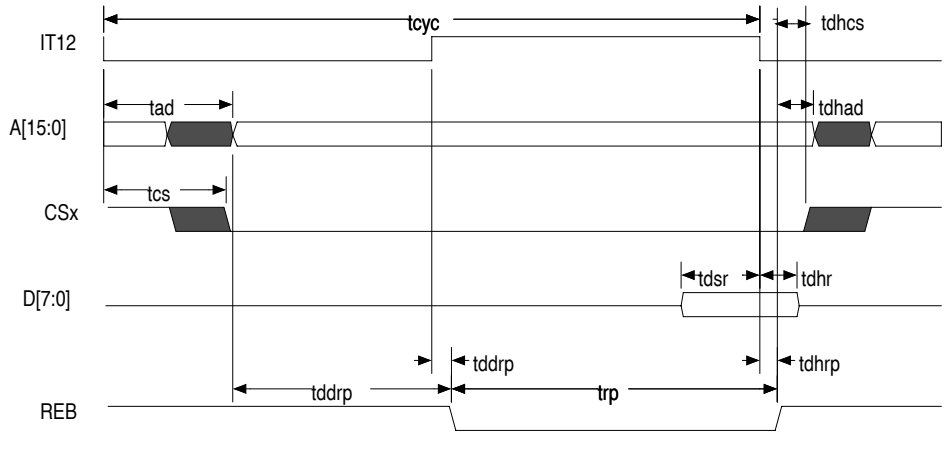


Figure A-11 High performance read timings

A

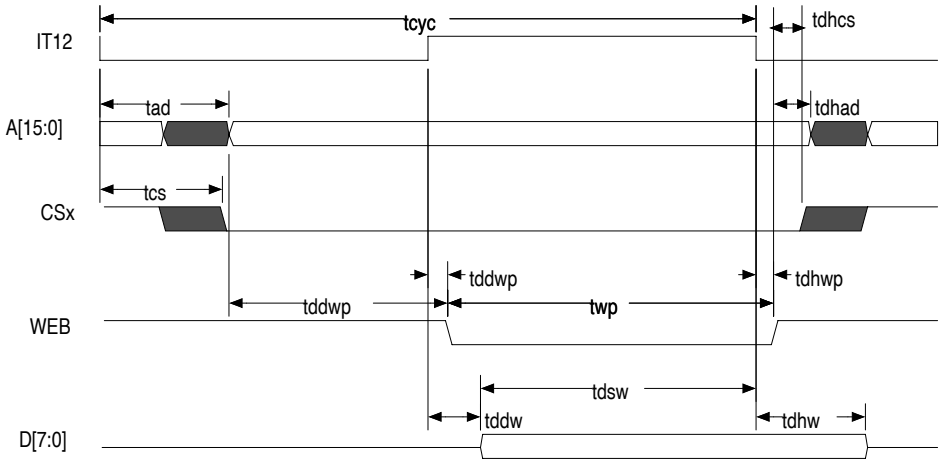


Figure A-12 High performance write timings

To operate at maximum bus frequency i.e 8MHz (125ns) without WAIT States:-

Bus Frq = Memory Access(max) + CSDelay(max) + Data Setup(min)

==> Memory Access(max) = 125 - 22 -16

==> Maximum Memory Access Time = 87ns

With an external memory access time of 120ns the maximum bus speed without WAIT States is :-

Bus Frq = Memory Access(max) + CSDelay(max) + Data Setup(min)

==> Maximum Bus Frequency = 120 + 22 + 16

==> Maximum Bus Frequency = 158ns = 6.33MHz

NOTE

Every effort has been made to ensure the above timings and calculations are accurate. More accurate information will be supplied once the silicon has been characterized.

A

Characteristic	Symbol	Min	Max	Unit
Cycle Time	tcyc	125		ns
IT12 to Valid External Address	tad		22	ns
IT12 to Valid Chip-Select	tcs		22	ns
External Address Hold Time	tdhad	tbd		ns
Chip-Select Hold Time	tdhcs	tbd		ns
Read Data Setup Time	tdsr	16		ns
Read Data Hold Time	tdhr	0		ns
Valid Address to Read Enable	tddrp	tbd		ns
Read Enable Pulse Width	trp	62		ns
Read Enable Hold Time	tdhrp	0		ns
Valid Address to Write Enable	tddwp	tbd		ns
Write Enable Pulse Width	twp	62		ns
Write Enable Hold Time	tdhwp	0		ns
Write Data Delay Time	tddw		16	ns
Write Data Setup Time	tdsw	tbd		ns
Write Data Hold Time	tdhw	62		ns
Write Data Hold Time (External WAIT states)		32		ns

Table A-9 High performance EBI timings

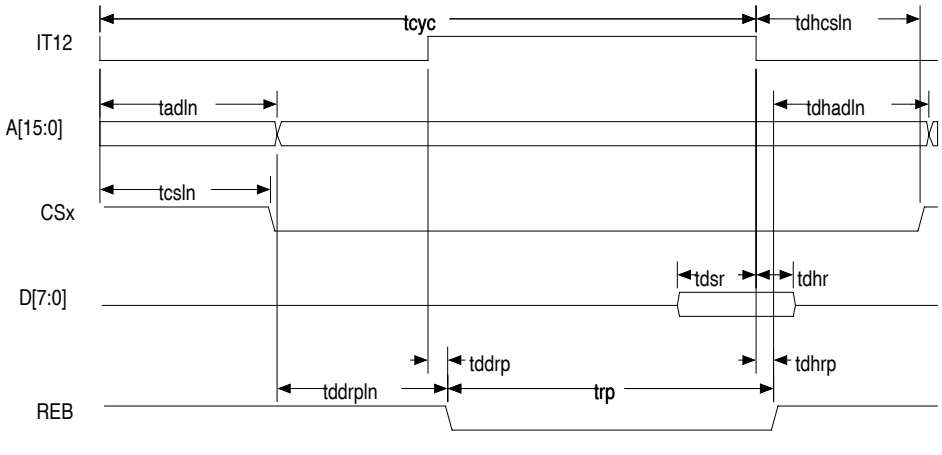


Figure A-13 Low noise mode read timings

A

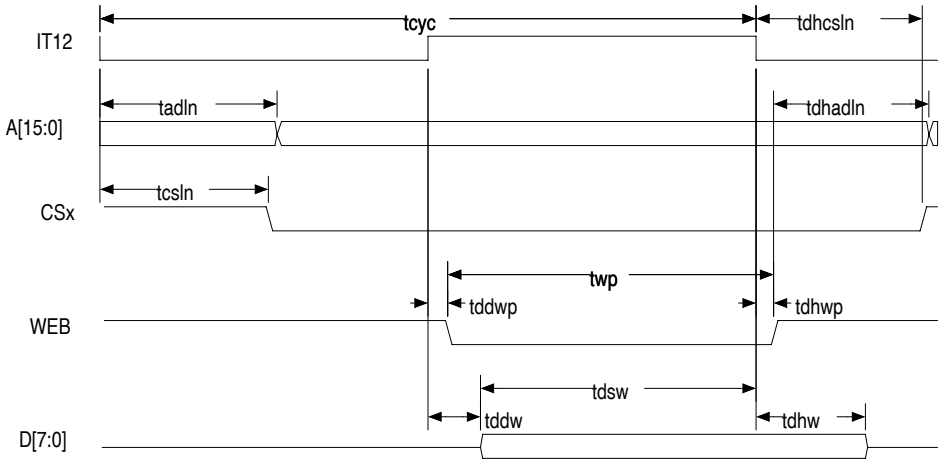


Figure A-14 Low-noise mode write timings

Characteristic	Symbol	Min	Max	Unit
Cycle Time	tcyc	125		ns
IT12 to Valid External Address	tadln		32	ns
IT12 to Valid Chip-Select	tcsln		32	ns
External Address Hold Time	tdhadln	tbd		ns
Chip-Select Hold Time	tdhcsln	tbd		ns
Read Data Setup Time	tdsr	16		ns
Read Data Hold Time	tdhr	0		ns
Valid Address to Read Enable	tddrpln	tbd		ns
Read Enable Pulse Width	trp	62		ns
Read Enable Hold Time	tdhrp	0		ns
Valid Address to Write Enable	tddwp	tbd		ns
Write Enable Pulse Width	twp	62		ns
Write Enable Hold Time	tdhwp	0		ns
Write Data Delay Time	tddw		16	ns
Write Data Setup Time	tdsw	tbd		ns
Write Data Hold Time	tdhw	62		ns

Table A-10 Low noise mode EBI timings

A

Characteristic	Symbol	Min	Max	Unit
Write Data Hold Time (External WAIT states)	tdhws	32		ns

Table A-10 Low noise mode EBI timings

To operate at maximum bus frequency i.e 8MHz (125ns) without WAIT States :-

Bus Frq = Memory Access(max) + CS Delay(max) + Data Setup(min)

==> Memory Access(max) = 125 - 32 -16

==> Maximum Memory Access Time = 77ns

Also, with an external memory access time of 120ns the maximum bus speed (without WAIT States) is :-

Bus Frq = Memory Access(max) + CSDelay(max) + Data Setup(min)

==> Maximum Bus Frequency = 120 + 32+ 16

==> Maximum Bus Frequency = 168ns = 5.9MHz

NOTE

Every effort has been made to ensure the above timings and calculations are accurate. More accurate information will be supplied once the silicon has been characterized.

A.2 HC08AZ16

The HC08AZ16 is a device similar to the HC08AZ32, but with 16,384 bytes of on-chip user ROM, and 512 bytes of RAM. The entire HC08AZ32 Technical Summary applies to the HC08AZ16, with the exceptions outlined in this appendix.

A.2.1 Features

- 16,384 bytes of on-chip user ROM
- 512 bytes of RAM
- Available in 64-pin QFP (Quad Flat Pack) package

A.2.2 Block diagram

A block diagram of the HC08AZ16 is shown in Figure A-15.

A.2.3 Pin assignments

The pin assignments for the HC08AZ16 are shown in Figure A-16.

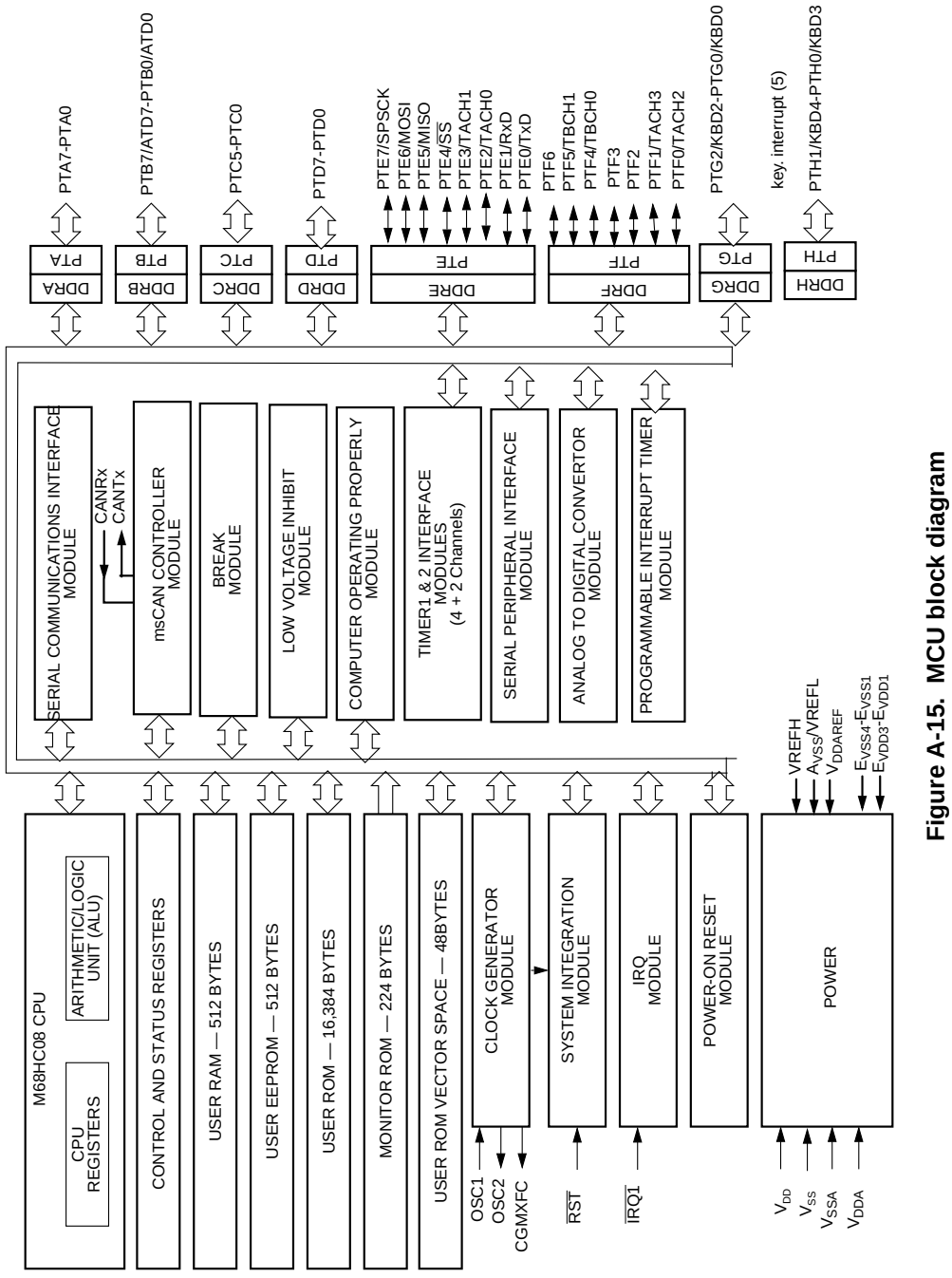


Figure A-15. MCU block diagram

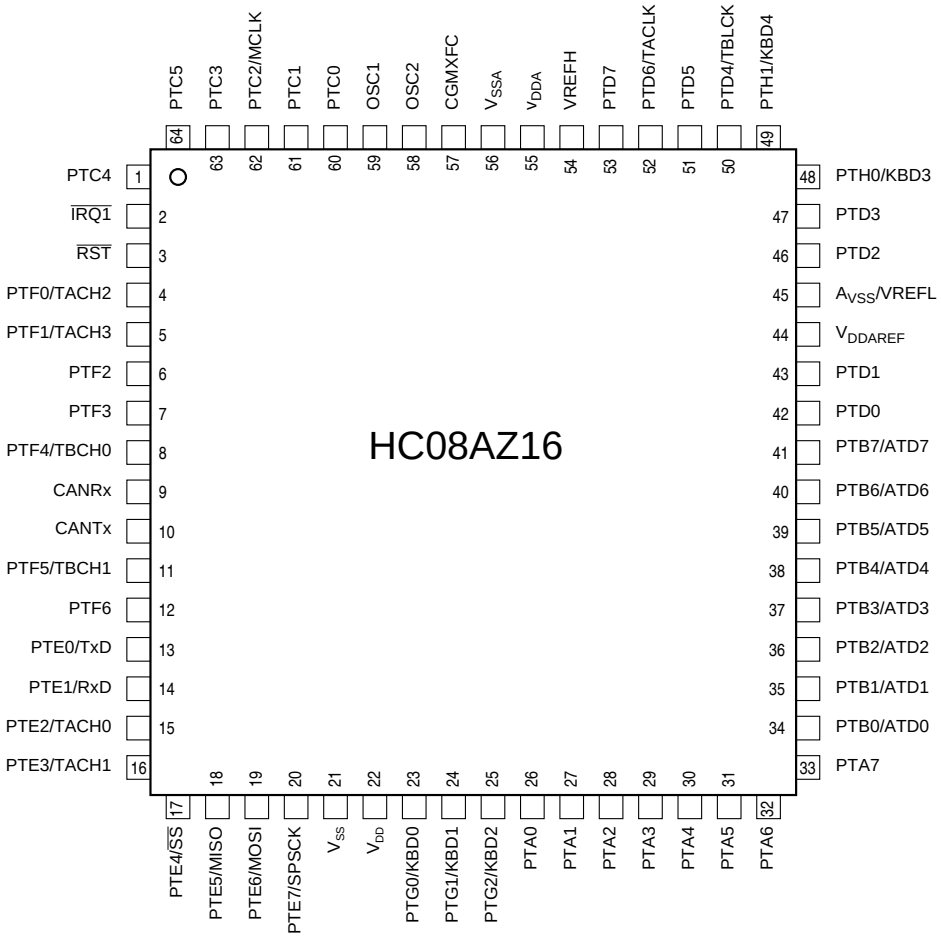


Figure A-16 QFP pin assignments

A.2.4 Memory section

The HC08AZ16 operates in Non-Expanded mode, where 16,384 bytes of on-chip user ROM is available.

A.2.5 Memory map

The memory map for the HC08AZ16 is shown in Figure A-17.

A.2.6 RAM

The HC08AZ16 has 512 bytes of user RAM, from locations $0050 to $024F.

A.2.7 ROM

The HC08AZ16 has 16,384 bytes of user ROM, from locations $BE00 to $FDFF. The monitor ROM and vectors are located from $FE20 - $FEFF.

Forty-eight user Vectors, $FFD0-$FFFF, are dedicated to user-defined reset and interrupt vectors.

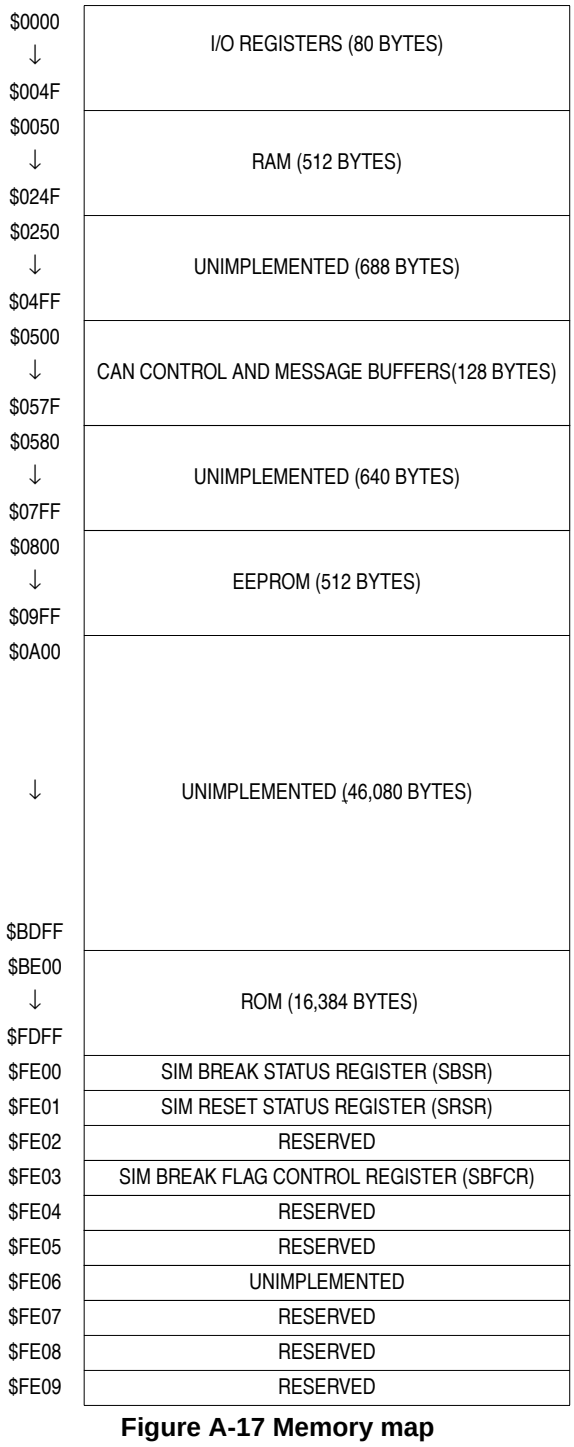


Figure A-17 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (48 BYTES)

Figure A-17 Memory map

A.3 HC08AZ24

The HC08AZ24 is a device similar to the HC08AZ32, but with 24,576 bytes of on-chip user ROM, and 768 bytes of RAM. The entire HC08AZ32 Technical Summary applies to the HC08AZ24, with the exceptions outlined in this appendix.

A.3.1 Features

- 24,576 bytes of on-chip user ROM
- 768 bytes of RAM
- Available in 64-pin QFP (Quad Flat Pack) package

A.3.2 Block diagram

The HC08AZ24 block diagram is shown in Figure A-18.

A.3.3 Pin assignments

The pin assignments for the HC08AZ24 are shown in Figure A-19.

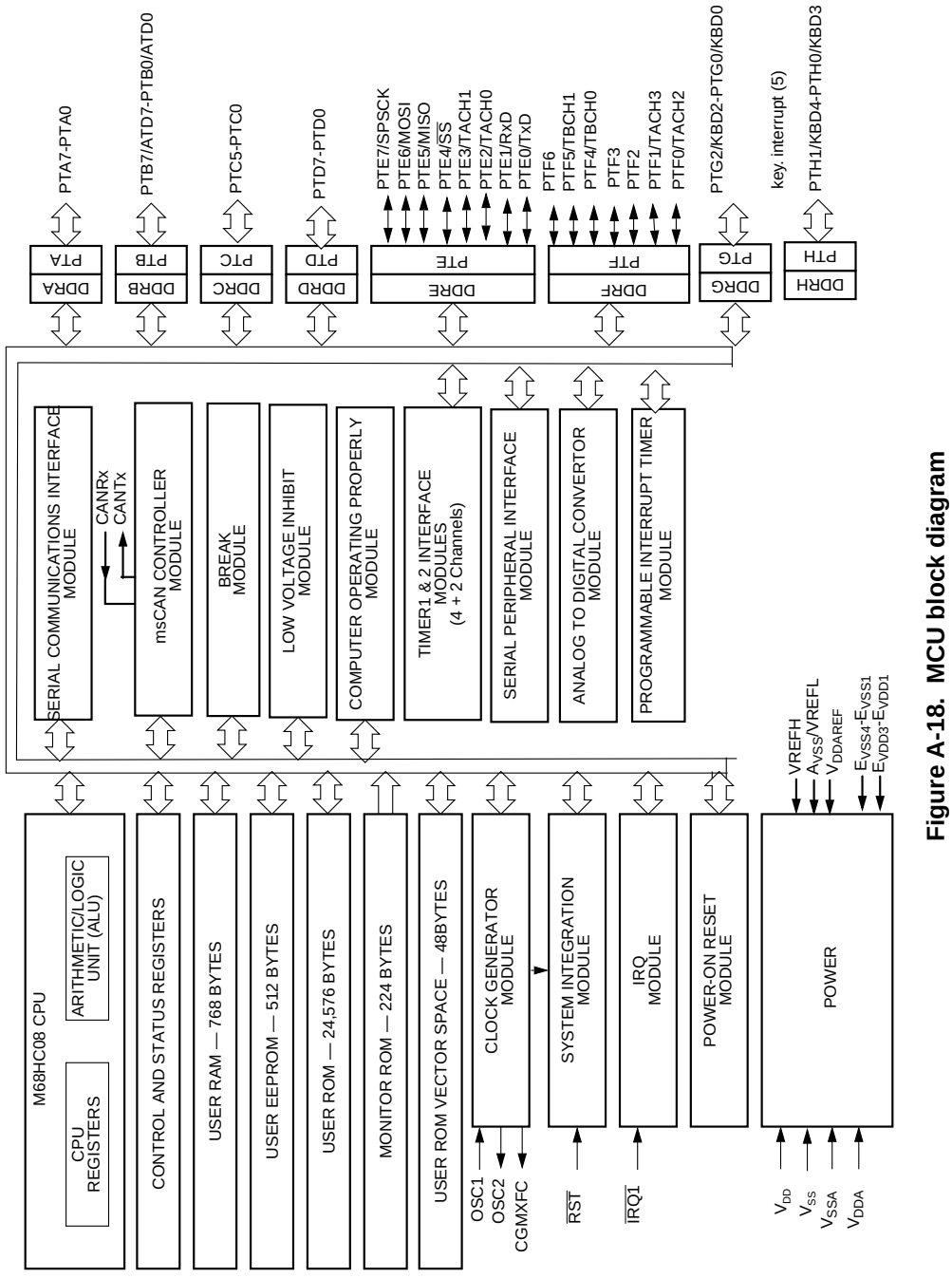


Figure A-18. MCU block diagram

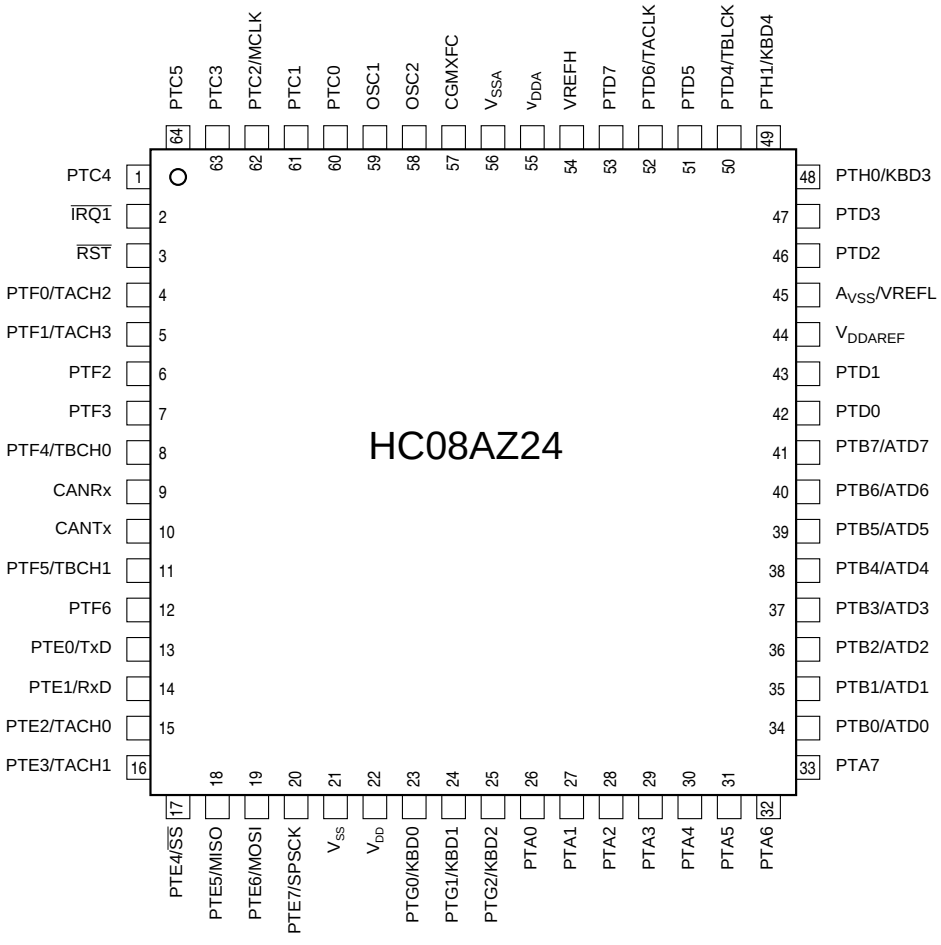


Figure A-19 QFP pin assignments

A

A.3.4 Memory section

The HC08AZ24 operates in Non-Expanded mode, where 24,576 bytes of on-chip user ROM is available.

A.3.5 Memory map

The memory map for the HC08AZ24 is shown in Figure A-20.

A.3.5.1 RAM

The HC08AZ24 has 768 bytes of user RAM, from locations $0050 to $034F.

A.3.5.2 ROM

The HC08AZ24 has 24,576 bytes of user ROM, from locations $9E00 to $FDFF. The monitor ROM and vectors are located from $FE20 - $FEFF.

Forty-eight user Vectors, $FFD0-$FFFF, are dedicated to user-defined reset and interrupt vectors.

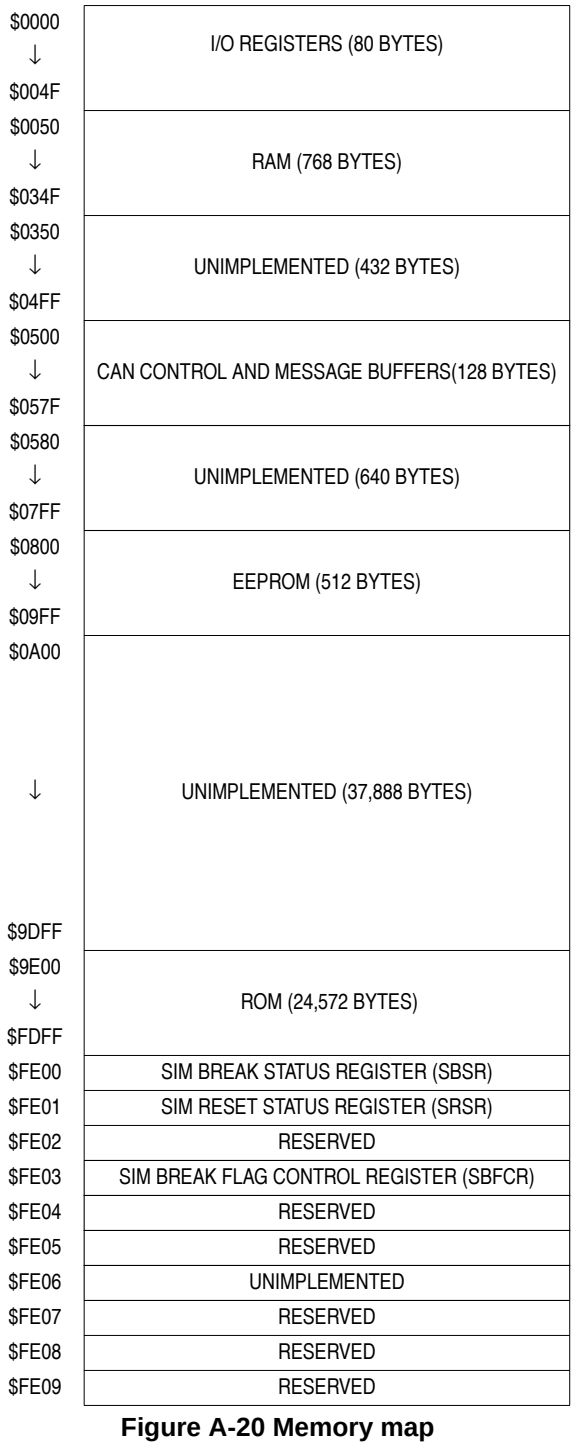


Figure A-20 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (48 BYTES)

Figure A-20 Memory map

A.4 HC08AB32

The HC08AB32 is a device similar to the HC08AZ32, but without the msCAN08 controller. The entire HC08AZ32 Technical Summary applies to the HC08AB32, with the exceptions outlined in this appendix.

A.4.1 Features

- No msCAN08 controller
- 32,255 bytes of on-chip user ROM
- 1024 bytes of RAM
- Available in 64-pin QFP (Quad Flat Pack) package

A.4.2 Block diagram

The HC08AB32 block diagram is shown in Figure A-21.

A.4.3 Pin assignments

The pin assignments for the HC08AB32 are shown in Figure A-22.

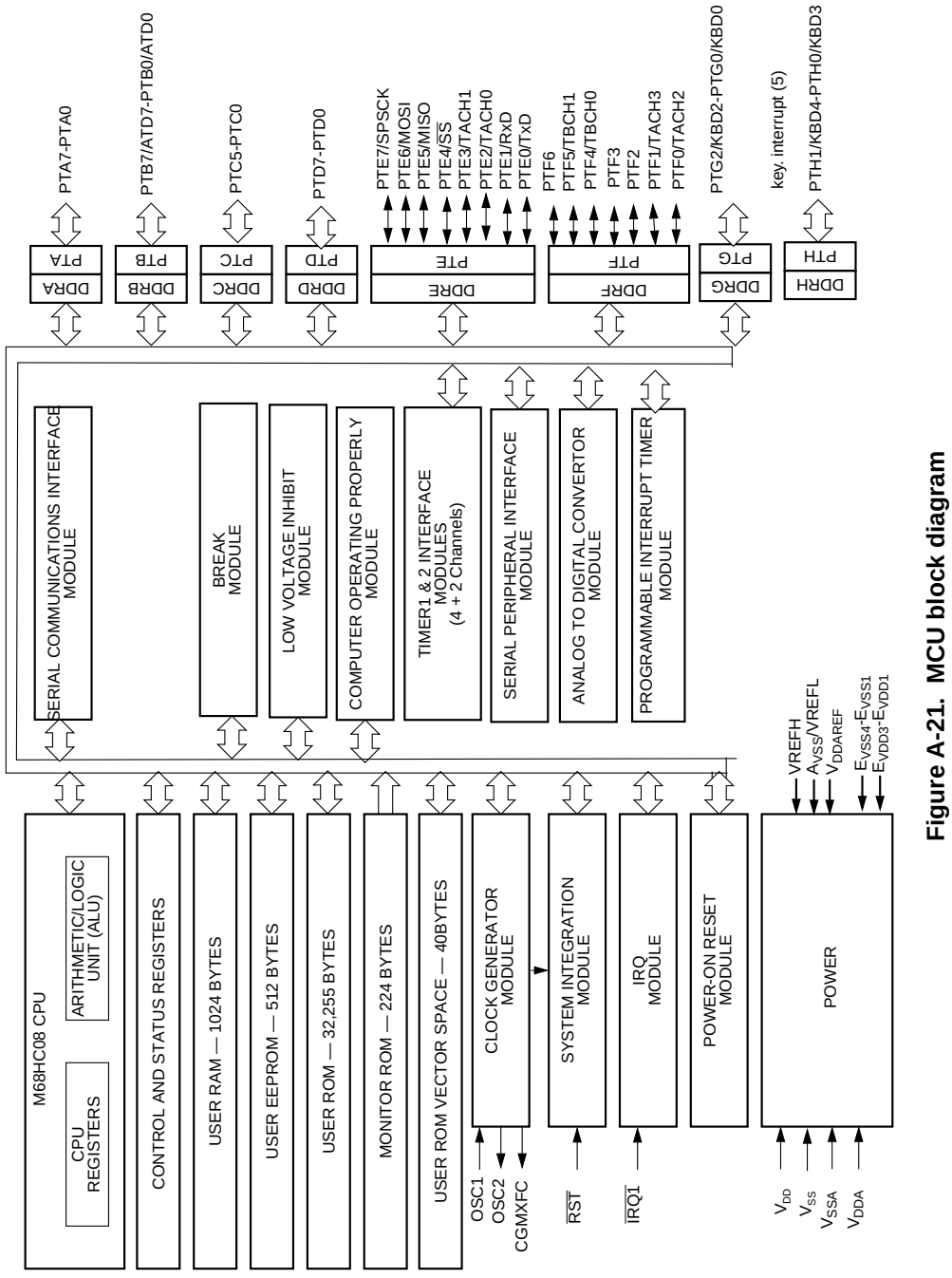


Figure A-21. MCU block diagram

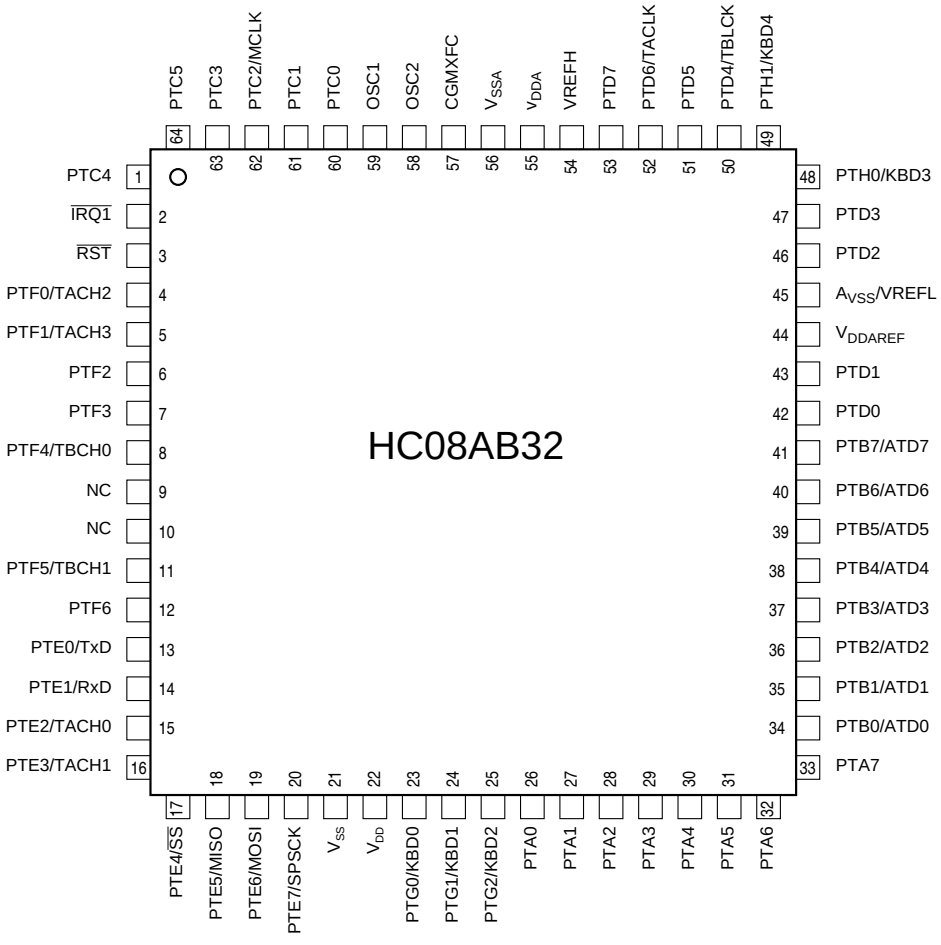


Figure A-22 QFP pin assignments

A

A.4.4 Memory section

The HC08AB32 operates in Non-Expanded mode, where 32,255 bytes of on-chip user ROM is available.

A.4.4.1 Memory map

The memory map for the HC08AB32 is shown in Figure A-23 .

A.4.4.2 RAM

The HC08AB32 has 1024 bytes of user RAM, from locations $0050 to $044F.

A.4.4.3 ROM

The HC08AB32 has 32,255 bytes of user ROM, from locations $8000 to $FDFF. The monitor ROM and vectors are located from $FE20 - $FEFF.

Forty user Vectors, $FFD0–$FFD9 and $FFE2–$FFFF, are dedicated to user-defined reset and interrupt vectors.

A.4.5 I/O section

The locations from $0500–$057F are unimplemented.

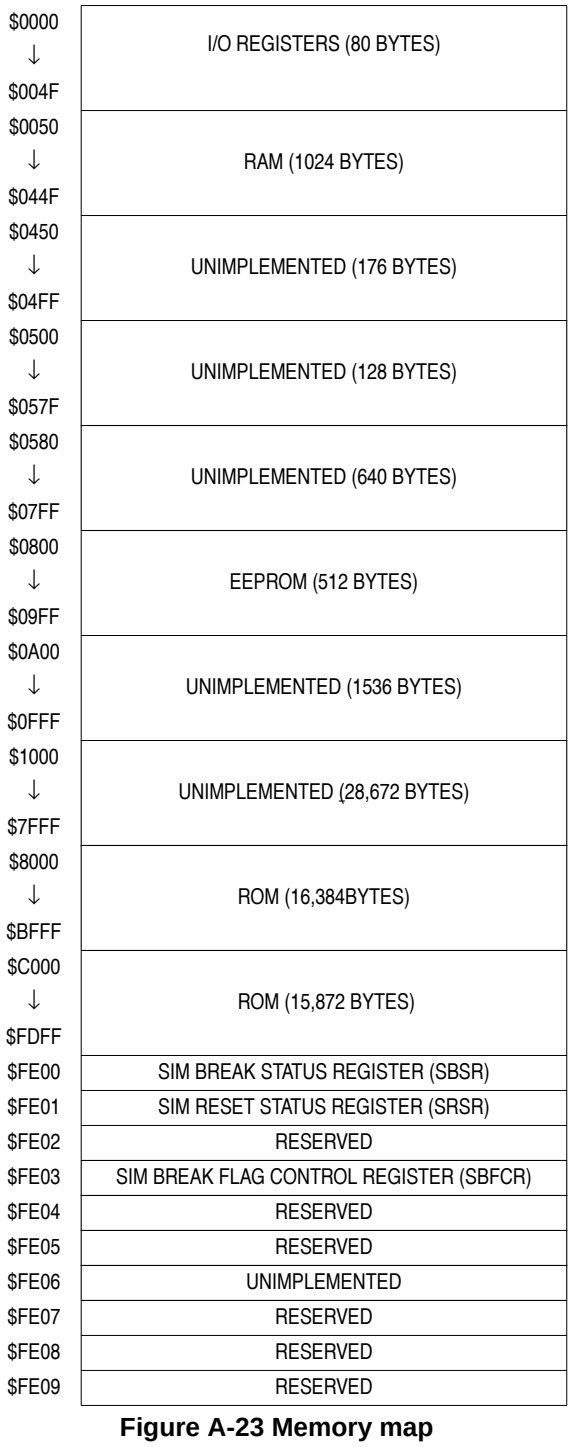


Figure A-23 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (40 BYTES)

Figure A-23 Memory map

A.5 HC08AB0

The HC08AB0 is a device similar to the HC08AZ0, but without the msCAN08 controller. The entire HC08AZ32 Technical Summary applies to the HC08AB0, with the exceptions outlined in this appendix.

A.5.1 Features

- No msCAN08 controller
- External Bus Interface (EBI)
- No on-chip user ROM or mask options
- Available in 100-pin TQFP (Thin Quad Flat Pack) package

A.5.2 MCU block diagram

A block diagram of the HC08AB0 is shown in Figure A-24.

A.5.3 Pin assignments

The pin assignments for the HC08AB0 are shown in Figure A-25.

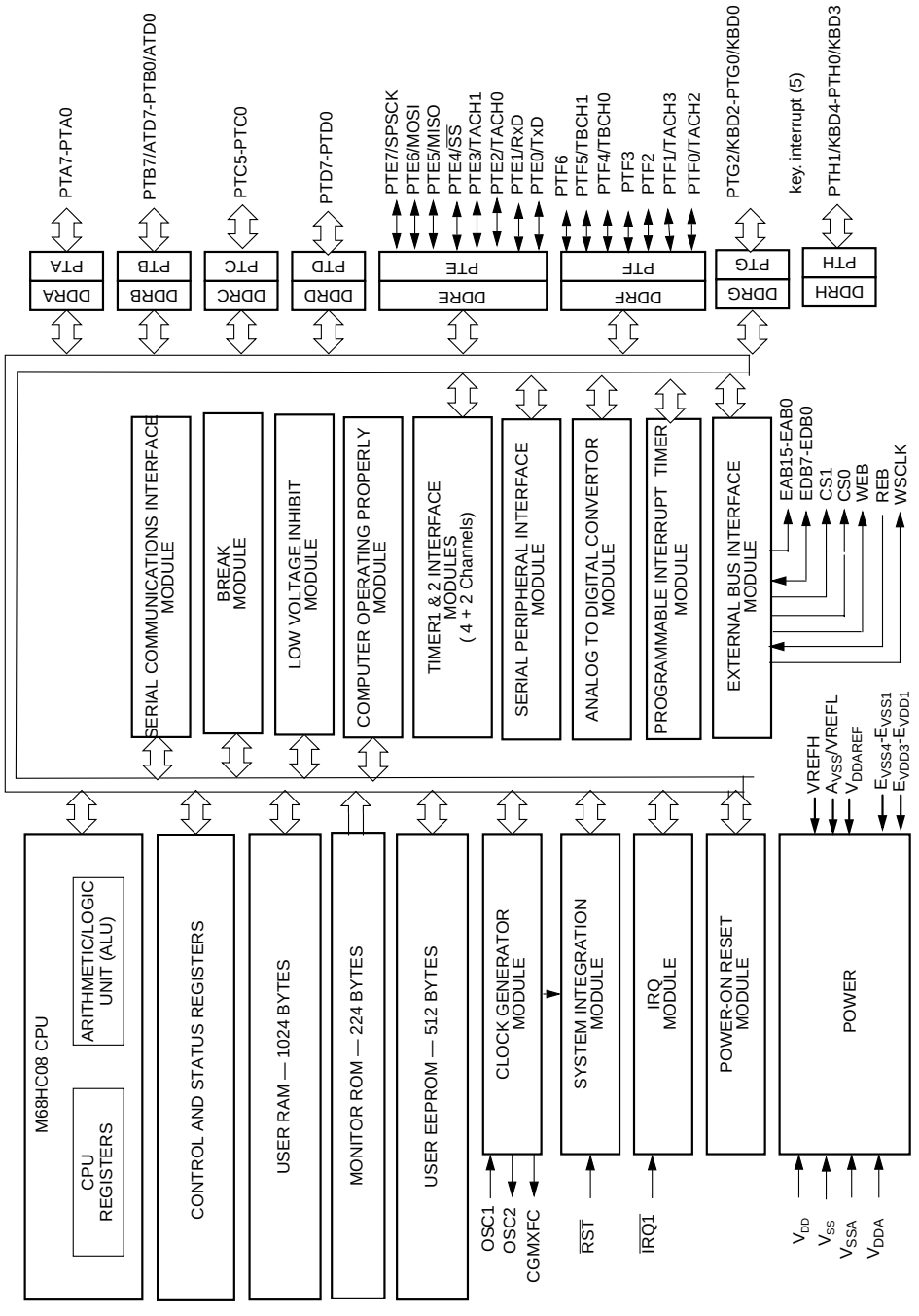


Figure A-24. MCU Block Diagram

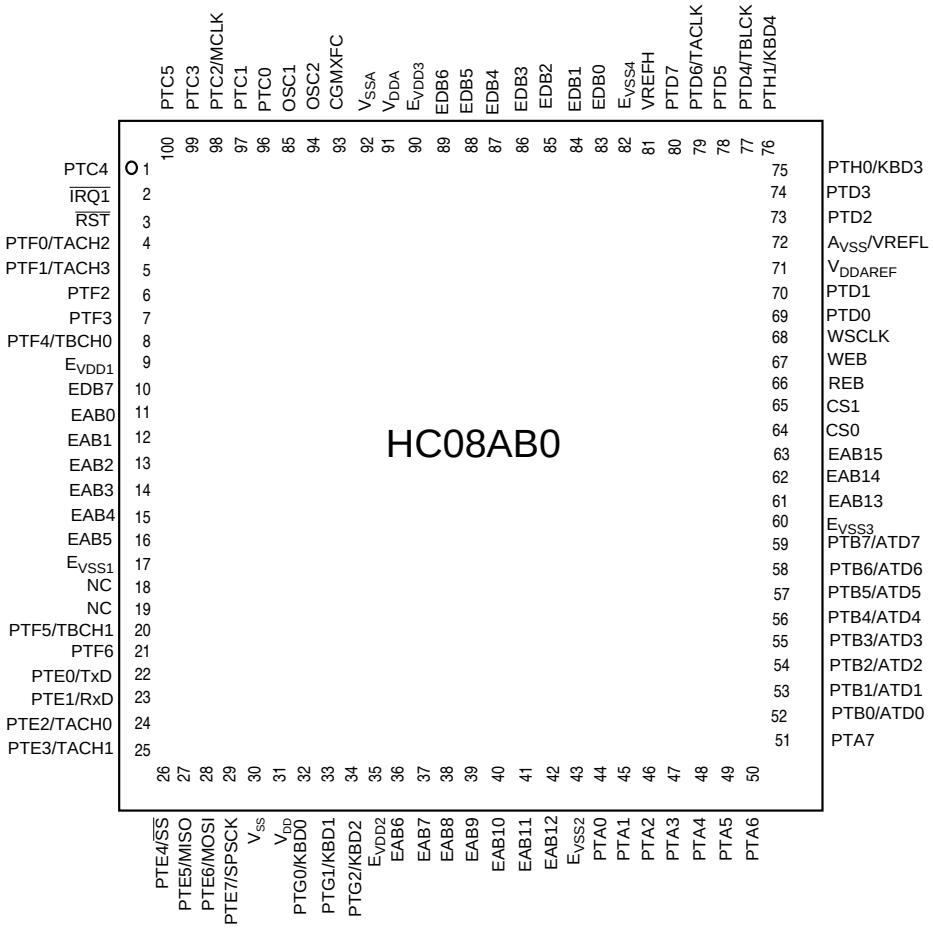


Figure A-25 100TQFP Pin Assignments (Top View)

A.5.4 Pin descriptions

A.5.4.1 Power supply pins(E_{VDD3}/E_{VDD1} and E_{VSS4}/E_{VSS1})

The E_{VDD} and E_{VSS} pins are for the sole use of the I/O pins. This will help reduce the effect of the noise induced into the V_{SS} power supply.

A.5.4.2 External data pins (EDB7-EDB0)

EDB7-EDB0 are the bidirectional data lines for connection to external peripherals.

A.5.4.3 External address pIns (EAB15-EAB0)

EAB15-EAB0 are the address lines for connection to external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.5.4.4 Write control pin (WEB)

WEB is the write control signal for external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.5.4.5 Read control pin (REB)

REB is the read control signal for external peripherals. See **A.1.14 External Bus Interface (EBI) Module**.

A.5.4.6 WAIT states clock pin (WSCLK)

WSCLK is the External WAIT State control signal. See **A.1.14 External Bus Interface (EBI) Module**.

A.5.4.7 External chip-select pins (CS1,CS0)

CS1 and CS0 are the chip-select lines for connection to external peripherals. See **A.1.14 External Bus Interface (EBI) Module**. The external pins are summarized in Table A-1 and the clock sources are shown in Table A-2.

Table A-11 External pins summary

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
EAB15-EAB0	External address bus	Output	NA	Output
EDB7-EDB0	External data bus	Dual state	NO	Input (Hi-Z)
REB	External read enable	Output	NA	Output
WEB	External write enable	Output	NA	Output
WSCLK	WAIT state clock	Output	NA	Hi-Z
CS1	Chip-select 1	Output	NA	Output
CS0	Chip-select 0	Output	NA	Output

Table A-12 Clock source summary

Module	Clock source
EBI	Bus clock

A.5.5 Memory section

The HC08AB0 operates in Expanded mode, where all memory space not occupied by internal peripherals or memory is available externally through the External Bus Interface (EBI).

A.5.5.1 Memory map

Figure A-26 shows the memory map of the HC08AB0.

A.5.5.2 RAM

The HC08AB0 has 1024 bytes of RAM, from locations $0050 to $044F.

A.5.6 I/O section

The locations from $0500–$057F are unimplemented.

A.5.7 ROM security

The ROM security feature is disabled.

A.5.8 LVI reset

The reset signal from the LVI module is enabled.

A.5.9 LVI power

The LVI module power is enabled.

A.5.10 STOP mode recovery delay

STOP mode recovery after 4096 CGMXCLK cycles.

A.5.11 STOP instruction

The STOP instruction is enabled.

A.5.12 COP module

The COP module is enabled.

A.5.13 Reset vector source

Forty user vectors, $FFD0–$FFD9 and $FFE2–$FFFF, are dedicated to user-defined reset and interrupt vectors. The reset vector source on the HC08AB0 is external.

A.5.14 EEPROM security

The EEPROM security function on the HC08AB0 is enabled. This can be used to prevent program/erase access to locations $08F0–$08FF of the EEPROM array and also to the EEACR/EENVR configuration registers. See **5.3.7 HC08AZ32 EEPROM security**.

NOTE

The settings described in Section A.5.7 to Section A.5.14 are subject to change.

A.5.15 External bus interface module

The external bus interface module (EBI), is identical to the EBI described in the HC08AZ0 section.

$0000 ↓ $004F	I/O REGISTERS (80 BYTES)
$0050 ↓ $044F	RAM (1024 BYTES)
$0450 ↓ $04FF	EXTERNAL (176 BYTES)
$0500 ↓ $057F	UNIMPLEMENTED
$0580 ↓ $07FF	EXTERNAL (640 BYTES)
$0800 ↓ $09FF	EEPROM (512 BYTES)
$0A00 ↓ $0FFF	EXTERNAL (1536 BYTES)
$1000 ↓ $7FFF	EXTERNAL (28,672 BYTES)
$8000 ↓ $BFFF	EXTERNAL (16,384BYTES)
$C000 ↓ $FDFF	EXTERNAL (15,872 BYTES)
$FE00	SIM BREAK STATUS REGISTER (SBSR)
$FE01	SIM RESET STATUS REGISTER (SRSR)
$FE02	RESERVED
$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04	RESERVED
$FE05	RESERVED
$FE06	UNIMPLEMENTED
$FE07	RESERVED
$FE08	RESERVED
$FE09	RESERVED

Figure A-26 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	EXTERNAL (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	EXTERNAL (192 BYTES)
$FFC0 ↓ $FFCF	EXTERNAL (16 BYTES)
$FFD0 ↓ $FFFF	EXTERNAL VECTORS (40 BYTES)

Figure A-26 Memory map

A.6 HC08AB16

The HC08AB16 is a device similar to the HC08AZ16, but without the msCAN08 controller. The entire HC08AZ32 Technical Summary applies to the HC08AB16, with the exceptions outlined in this appendix.

A.6.1 Features

- No msCAN controller
- 16,384 bytes of on-chip user ROM
- 512 bytes of RAM
- Available in 64-pin QFP (Quad Flat Pack) package

A.6.2 Block diagram

A block diagram of the HC08AB16 is shown in Figure A-27

A.6.3 Pin assignments

The pin assignments for the HC08AB16 are shown in Figure A-28.

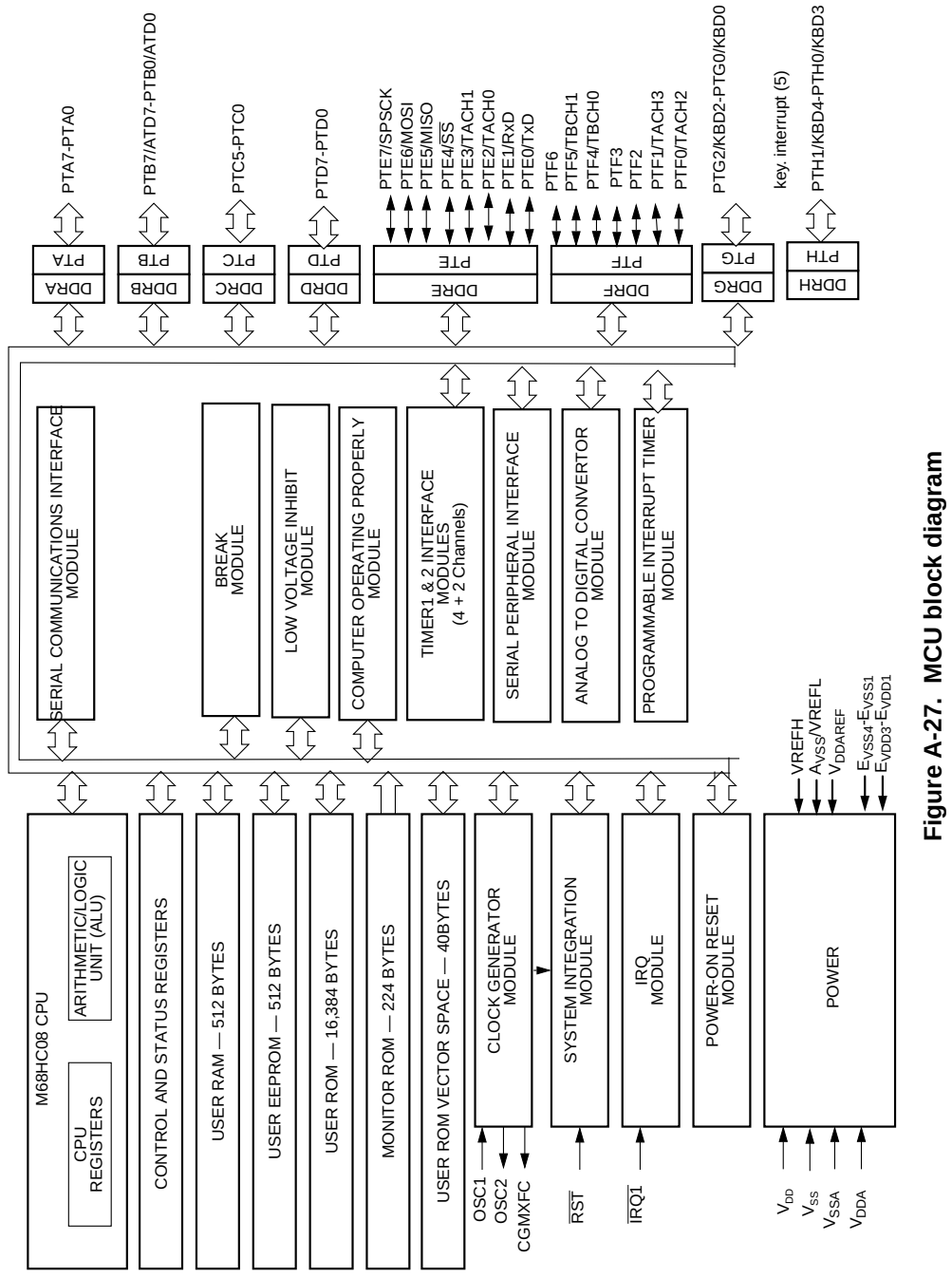


Figure A-27. MCU block diagram

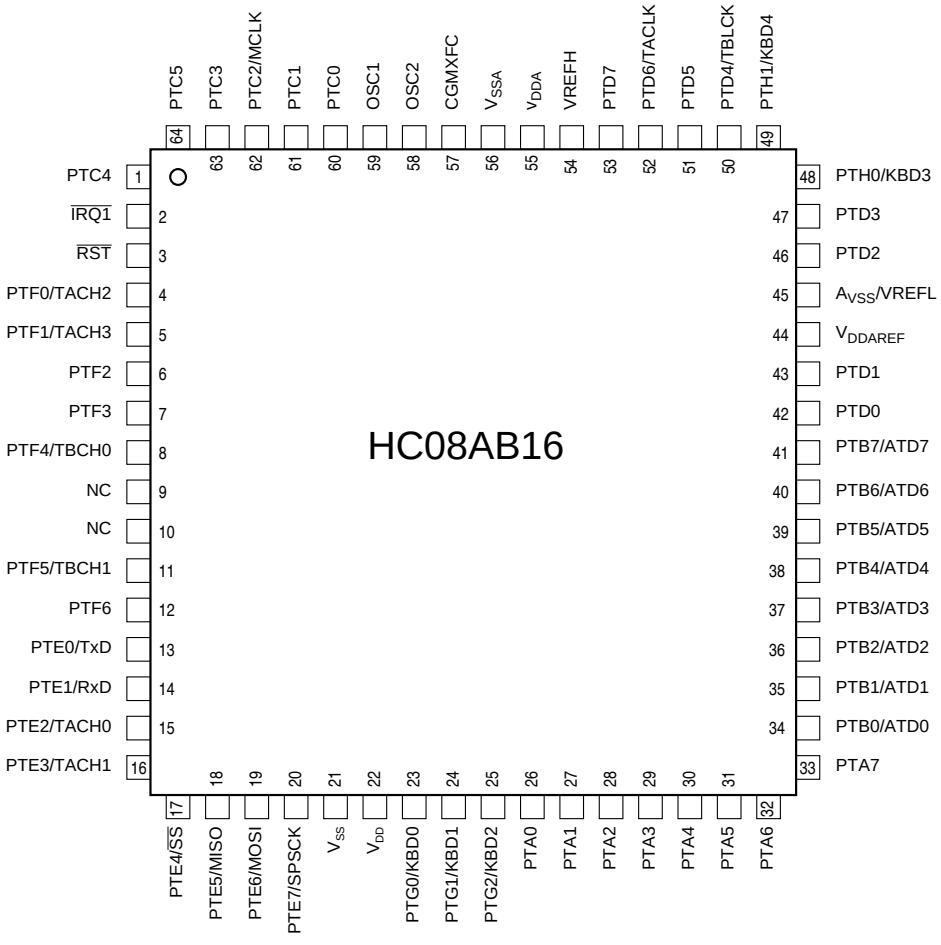


Figure A-28 QFP pin assignments

A.6.4 Memory section

The HC08AB16 operates in Non-Expanded mode, where 16,384 bytes of on-chip user ROM is available.

A.6.4.1 Memory map

The memory map for the HC08AB16 is shown in Figure A-29.

A.6.4.2 RAM

The HC08AB16 has 512 bytes of user RAM, from locations $0050 to $024F.

A.6.4.3 ROM

The HC08AB16 has 16,384 bytes of user ROM, from locations $BE00 to $FDFF. The monitor ROM and vectors are located from $FE20 - $FEFF.

Forty user Vectors, $FFD0–$FFD9 and $FFE2–$FFFF, are dedicated to user-defined reset and interrupt vectors.

A.6.5 I/O section

The locations from $0500–$057F are unimplemented.

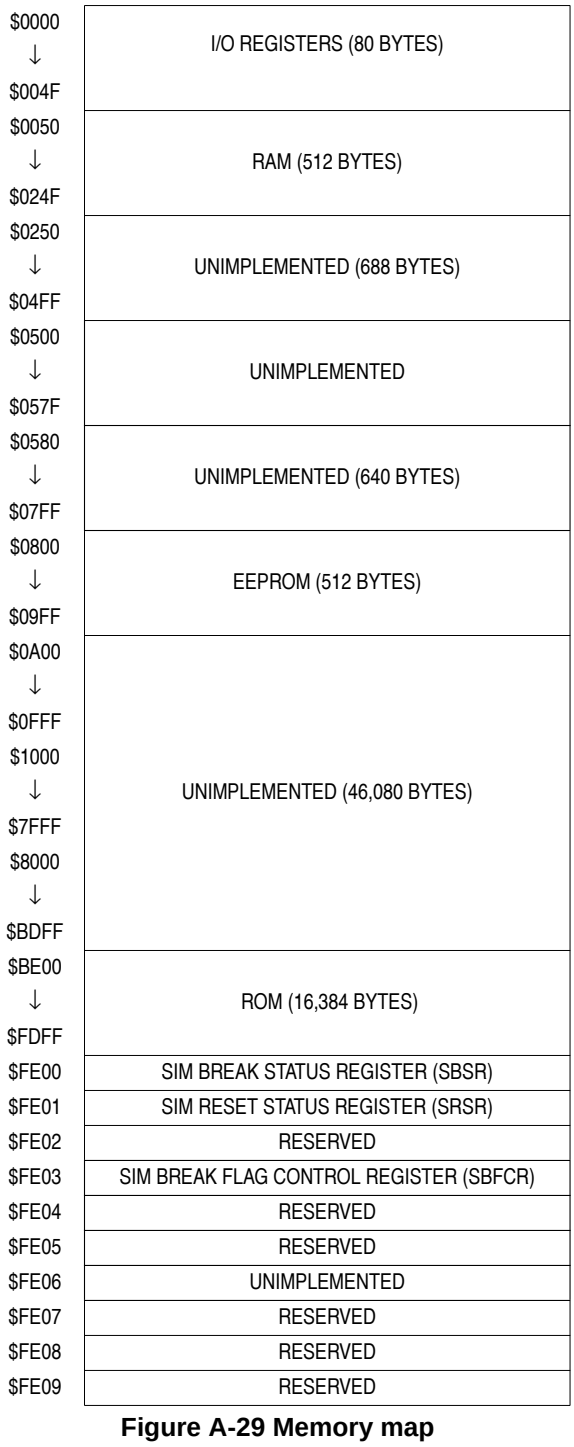


Figure A-29 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (40 BYTES)

Figure A-29 Memory map

A.7 HC08AB24

The HC08AB24 is a device similar to the HC08AZ24, but without the msCAN08 controller. The entire HC08AZ32 Technical Summary applies to the HC08AB24, with the exceptions outlined in this appendix.

A.7.1 Features

- No msCAN08 controller
- 24,576 bytes of on-chip user ROM
- 768 bytes of RAM
- Available in 64-pin QFP (Quad Flat Pack) package

A.7.2 Block diagram

The HC08AB24 block diagram is shown in Figure A-30.

A.7.3 Pin assignments

The pin assignments for the HC08AB24 are shown in Figure A-31.

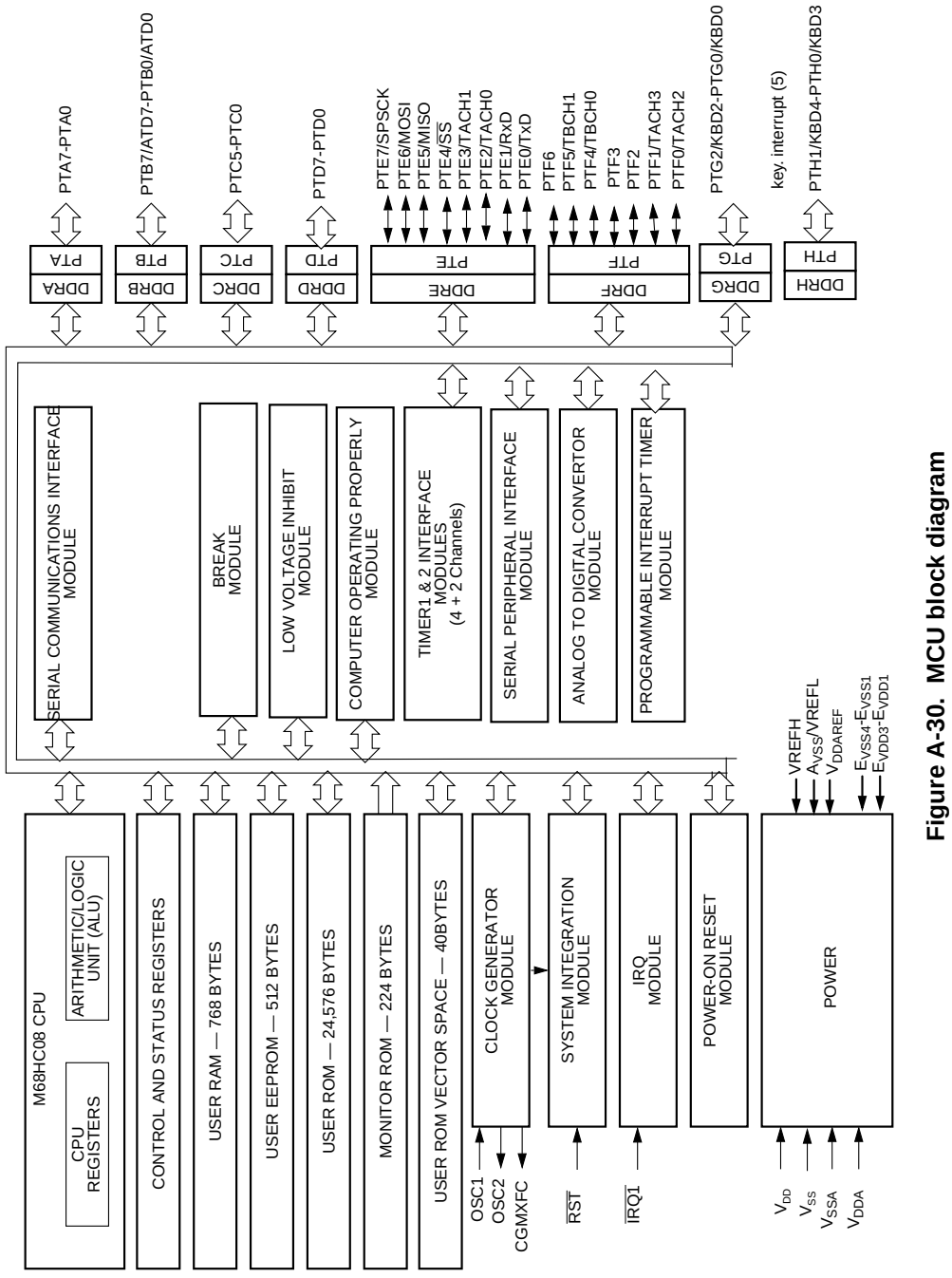


Figure A-30. MCU block diagram

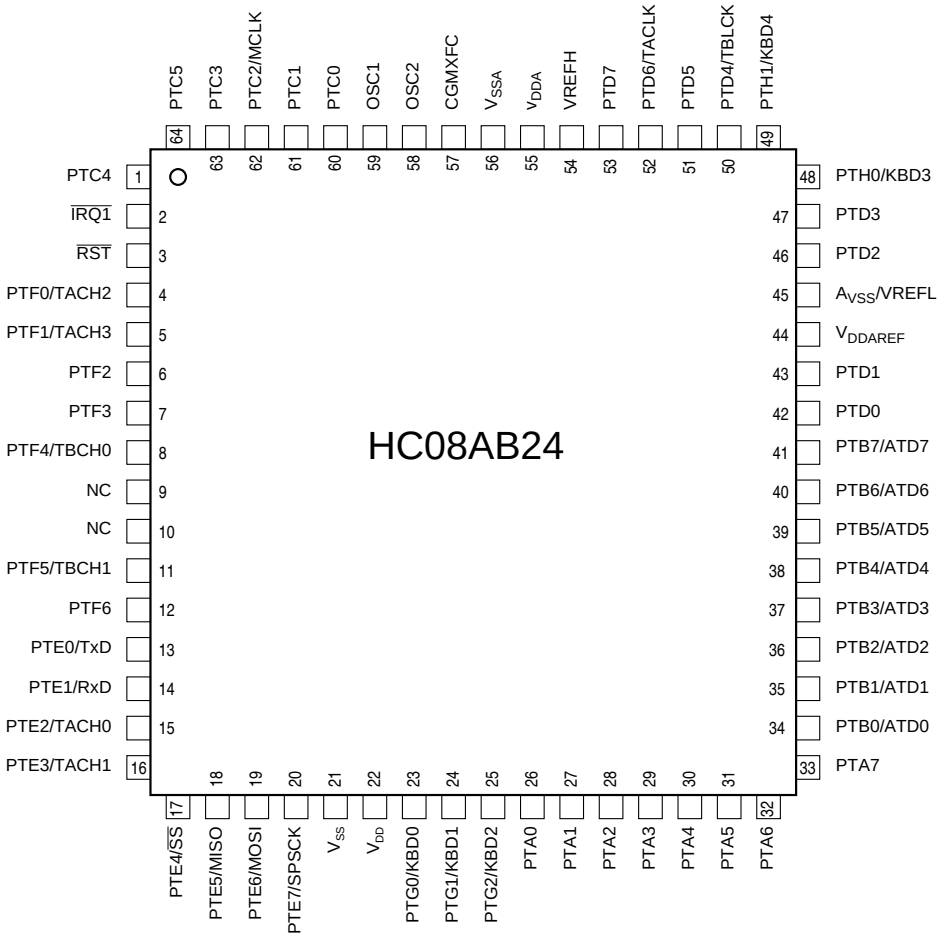


Figure A-31 QFP pin assignments

A.7.4 Memory section

The HC08AB24 operates in Non-Expanded mode, where 24,576 bytes of on-chip user ROM is available.

A.7.4.1 Memory map

The memory map for the HC08AB24 is shown in Figure A-32.

A.7.4.2 RAM

The HC08AB24 has 768 bytes of user RAM, from locations $0050 to $034F.

A.7.4.3 ROM

The HC08AB24 has 24,576 bytes of user ROM, from locations $9E00 to $FDFF. The monitor ROM and vectors are located from $FE20 - $FEFF.

Forty user Vectors, $FFD0–$FFD9 and $FFE2–$FFFF, are dedicated to user-defined reset and interrupt vectors.

A.7.5 I/O section

The locations from $0500–$057F are unimplemented.

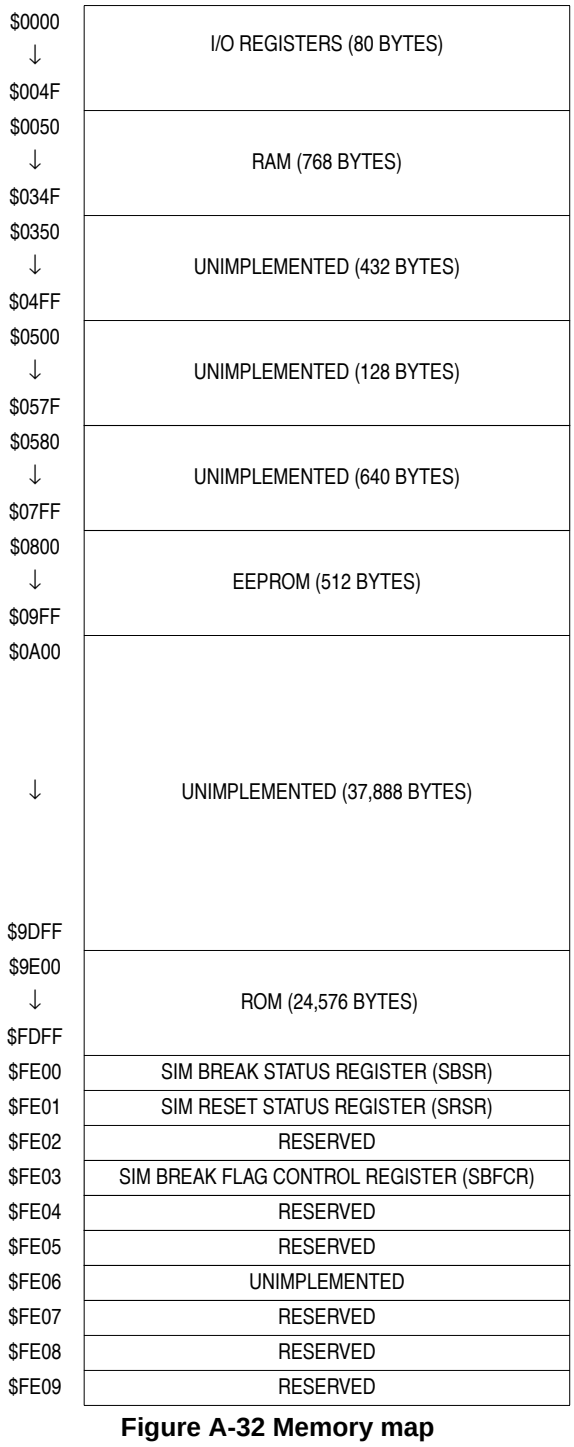


Figure A-32 Memory map

A

$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFCF	ROM (16 BYTES)
$FFD0 ↓ $FFFF	VECTORS (40 BYTES)

Figure A-32 Memory map

INDEX

In this index numeric entries are placed first; page references in *italics* indicate that the reference is to a figure.

D

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

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

Mfax™: RMFAX0@email.sps.mot.com, http://sps.motorola.com/mfax/ – TOUCHTONE 1-602-244-6609 Motorola Fax Back System

INTERNET: http://motorola.com/sps/

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

Customer Focus Center: 1-800-521-6274

Mfax is a trademark of Motorola, Inc.

HC08AZ32TS/D

MOTOROLA