Order this document by
MC68HC705V12/D
Rev. 3.0

HC05

68HC705V12

Advance Information

This document contains information on a new product.
Specifications and information herein are subject to change without notice.

MOTOROLA

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

List of Sections

Table of Contents

Section 1. General Description

Section 2. Memory Map

Section 3. Central Processor Unit (CPU)

Section 4. Interrupts

Section 5. Resets

Section 6. Low-Power Modes

Section 7. Parallel Input/Output (I/O)

Section 8. Core Timer

Section 9. 16-Bit Timer

Section 10. Serial Peripheral Interface (SPI)

Section 11. Pulse Width Modulators (PWMs)

Section 12. EPROM and EEPROM

Section 13. Analog-to-Digital (A/D) Converter

Section 14. Byte Data Link Controller – Digital (BDLC–D)

Section 15. Gauge Drivers

Section 16. Instruction Set

Section 17. Electrical Specifications

Section 18. Mechanical Specifications

Section 19. Ordering Information

List of Figures

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The Motorola MC68HC705V12 microcontroller is a custom M68HC05-based MCU featuring a byte data link controller (BDLC) module and on-chip power regulation for the on-chip gauge drivers. The device is available packaged in a 68-pin plastic leaded chip carrier (PLCC).

A functional block diagram of the MC68HC705V12 is shown in **Figure 1-1**.

1.3 Features

Features of the MC68HC705V12 include:

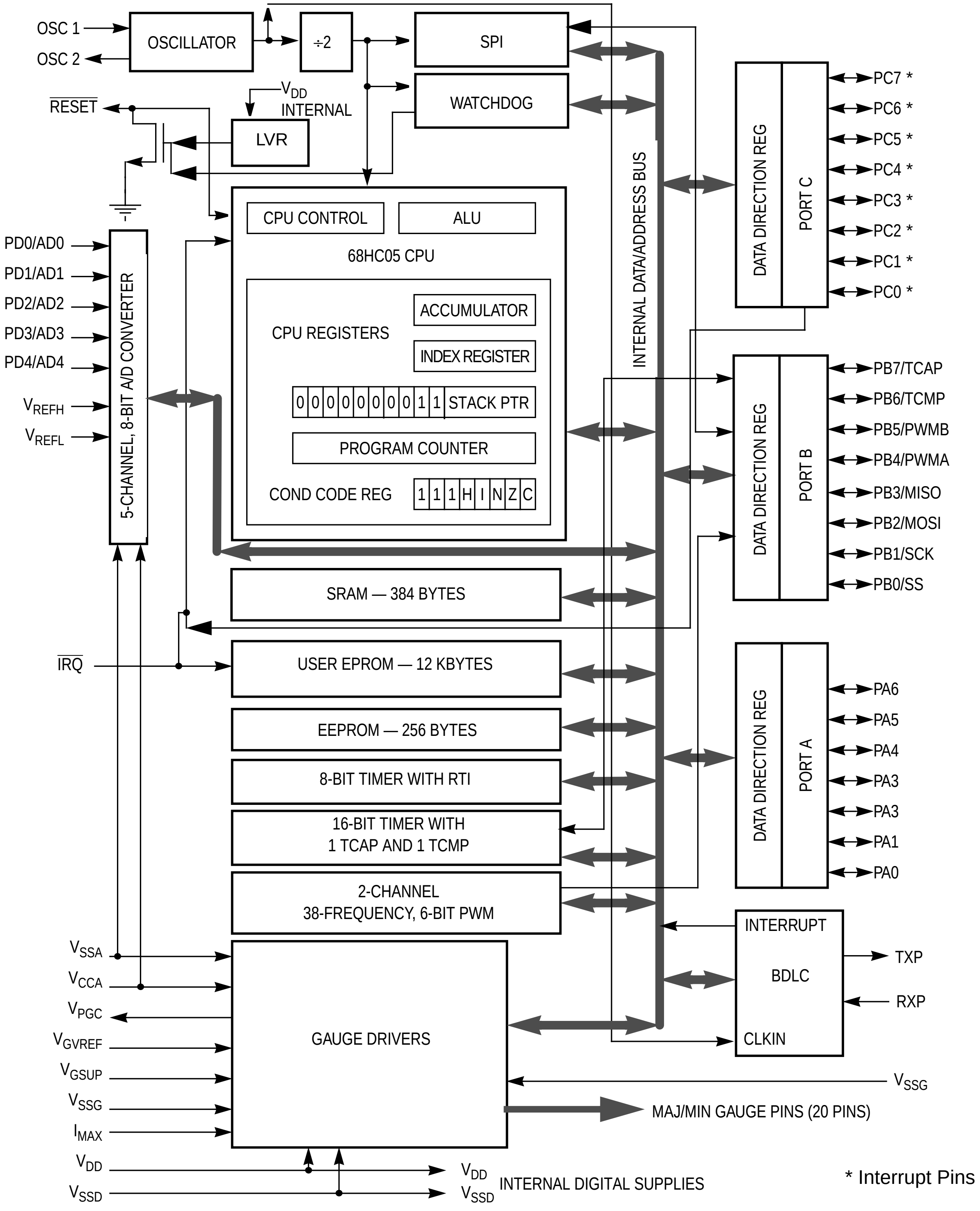
- M68HC05 core with on-chip oscillator for crystal/ceramic resonator
- 12 Kbytes of user erasable programmable read-only memory (EPROM) and 384 bytes of user random-access memory (RAM)
- 256 bytes of byte, block, or bulk electrically erasable programmable read-only memory (EEPROM)
- Byte data link controller (BDLC) module
- 5-channel, 8-bit analog-to-digital (A/D) converter
- Serial peripheral interface (SPI)

- 8-bit timer with real-time interrupt (RTI)
- 16-bit timer with one input capture and one output compare
- Two 38-frequency, 6-bit pulse width modulators (PWMs)
- Mask option register (MOR) selectable computer operating properly (COP) watchdog system
- 23 general-purpose input/output (I/O) pins:
  - Eight I/O pins with interrupt wakeup capability
  - Eight I/O pins multiplexed with timer, PWMs, and SPI pins
  - Seven general-purpose I/O pins
- Five input-only pins multiplexed with analog to digital (A/D)
- On-chip H-bridge driver circuitry to drive six gauges:
  - Four minor gauges
  - Two major gauges
- MOR selectable low-voltage reset (LVR)
- Power-saving stop mode and wait mode instructions (MOR selectable STOP instruction disable)

1.4 MCU Structure

The overall block diagram of the MC68HC705V12 is shown in **Figure 1-1**.

NOTE: *A line over a signal name indicates an active low signal. For example, RESET is active high and $\overline{\text{RESET}}$ is active low. Any reference to voltage, current, or frequency specified in the following sections will refer to the nominal values. The exact values and their tolerance or limits are specified in **Section 17. Electrical Specifications**.*



* Interrupt Pins

Figure 1-1. MC68HC705V12 Block Diagram

1.5 Programmable Mask Options

These mask options are programmable via the MOR (see **12.5 EPROM Programming**):

- Sensitivity on $\overline{\text{IRQ}}$ interrupt, edge- and level-sensitive or edge-sensitive only
- Selectable COP watchdog system enable/disable
- Selectable low-voltage reset (LVR) to hold the central processor unit (CPU) in reset
- Selectable STOP instruction disable

The mask options are provided through individual bits within the mask option register which is located and programmed as part of the EPROM array.

1.6 Operating Modes

The MCU has two modes of operation intended for users:

- User mode
- Bootloader mode

These modes are briefly described in the following subsections.

1.6.1 User Mode

This mode is the intended mode of operation for executing user firmware. All user mode functions are as described in this document.

1.6.2 Bootloader Mode

This mode is used for programming the on-chip erasable programmable read-only memory (EPROM). See **Section 12. EPROM and EEPROM** for more details on EPROM programming.

1.7 Functional Pin Descriptions

The pinout for the MC68HC705V12 is shown in **Figure 1-2** and is followed by a functional description of each pin.

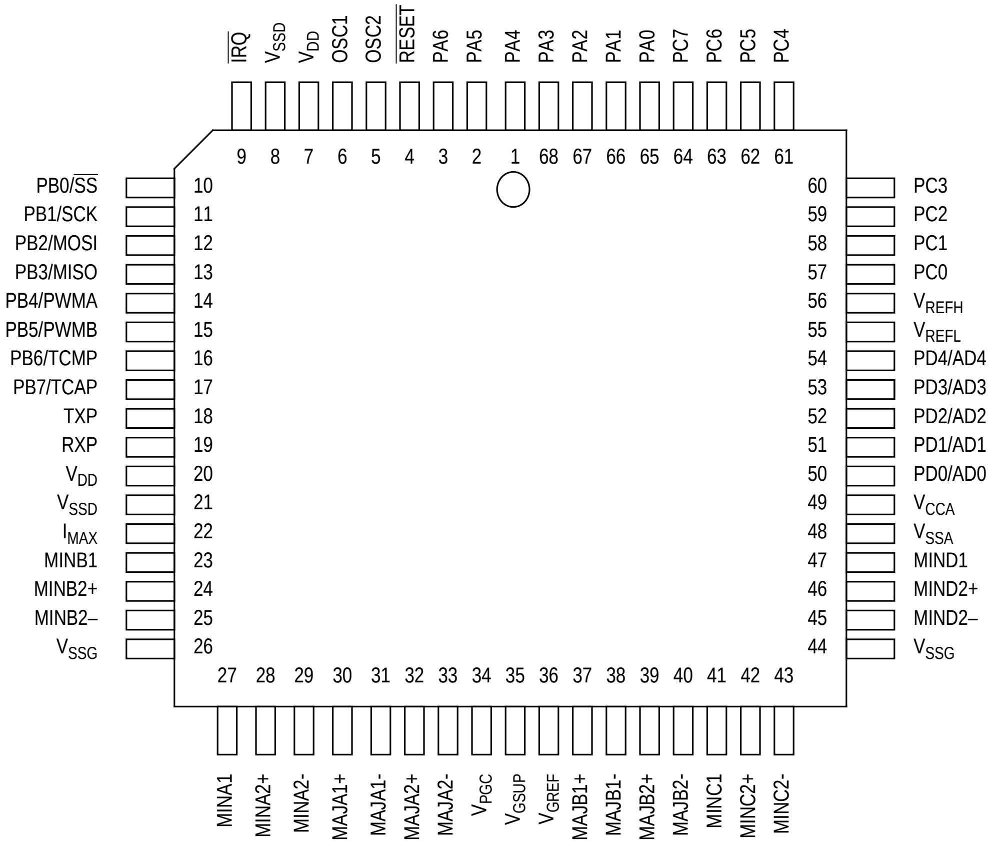


Figure 1-2. Pin Assignments (68-Pin PLCC Package)

1.7.1 V_{DD} and V_{SSD}

These pins provide power to all the microcontroller’s digital circuits. The short rise and fall times of the MCU supply current transients place very high short-duration current demands on the internal power supply. To prevent noise problems, special care should be taken to provide good power supply bypassing at the MCU by using bypass capacitors with good high-frequency characteristics that are positioned as close to the

MCU supply pins as possible. Two sets of V_{DD} and V_{SS} pins are required to maintain on-chip supply noise within acceptable limits. Each supply pin pair will require its own decoupling capacitor. These are high-current pins.

1.7.2 V_{SSA}

V_{SSA} is a separate ground pad which provides a ground return for the analog-to-digital (A/D) subsystem and the digital-to-analog (D/A) gauge subsystem. To prevent digital noise contamination, this pin should be connected directly to a low-impedance ground reference point.

1.7.3 V_{CCA}

V_{CCA} is a separate supply pin providing power to the analog subsystems of the A/D converter and gauge drivers. This pin must be connected to the V_{DD} pin externally. To prevent contamination from the digital supply, this pin should be adequately decoupled to a low-impedance ground reference.

1.7.4 V_{REFH} and V_{REFL}

V_{REFH} is the positive (high) reference voltage for the A/D subsystem. V_{REFL} is the negative (low) reference voltage for the A/D subsystem. V_{REFH} and V_{REFL} should be isolated from the digital supplies to prevent any loss of accuracy from the A/D converter.

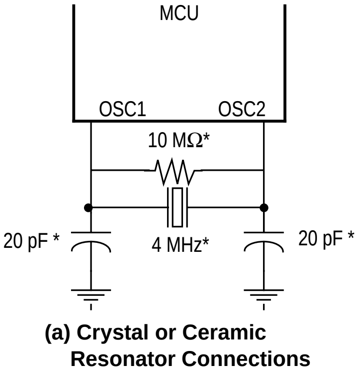
1.7.5 OSC1 and OSC2

The OSC1 and OSC2 pins are the connections for the on-chip oscillator. OSC1 is the input to the oscillator inverter. The output (OSC2) will always reflect OSC1 inverted except when the device is in stop mode which forces OSC2 high.

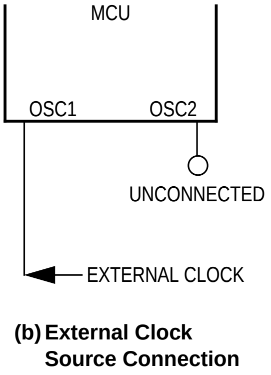
The OSC1 and OCS2 pins can accept these sets of components:

1. A crystal as shown in **Figure 1-3**(a)
2. A ceramic resonator as shown in **Figure 1-3**(a)
3. An external clock signal as shown in **Figure 1-3**(b)

The frequency, f_{OSC}, of the oscillator or external clock source is divided by two to produce the internal operating frequency, f_{OP}.



(a) Crystal or Ceramic Resonator Connections



(b) External Clock Source Connection

*Values shown are typical. For further information, consult the crystal oscillator vendor.

Figure 1-3. Oscillator Connections

1.7.5.1 Crystal Oscillator

The circuit in **Figure 1-3**(a) shows a typical oscillator circuit for an AT-cut, parallel resonant crystal.

NOTE: *The crystal manufacturer's recommendations should be followed, as the crystal parameters determine the external component values required to provide maximum stability and reliable startup.*

The load capacitance values used in the oscillator circuit design should include all stray capacitances. The crystal and components should be mounted as close as possible to the pins for startup stabilization and to minimize output distortion and radiated emissions.

1.7.5.2 Ceramic Resonator Oscillator

In cost-sensitive applications, a ceramic resonator can be used in place of the crystal. The circuit in **Figure 1-3**(a) can be used for a ceramic resonator.

NOTE: *The resonator manufacturer's recommendations should be followed, as the resonator parameters determine the external component values required for maximum stability and reliable starting.*

The load capacitance values used in the oscillator circuit design should include all stray capacitances. The ceramic resonator and components should be mounted as close as possible to the pins for startup stabilization and to minimize output distortion and radiated emissions.

1.7.5.3 External Clock

An external clock from another CMOS-compatible device can be connected to the OSC1 input. The OSC2 pin should be left unconnected, as shown in **Figure 1-3**(b).

1.7.6 $\overline{\text{RESET}}$

This pin can be used as an input to reset the MCU to a known startup state by pulling it to the low state. The $\overline{\text{RESET}}$ pin contains an internal Schmitt trigger to improve its noise immunity as an input. The $\overline{\text{RESET}}$ pin has an internal pulldown device that pulls the $\overline{\text{RESET}}$ pin low when there is a COP watchdog reset, power-on reset (POR), illegal address reset, a disabled STOP instruction reset, or an internal low-voltage reset. Refer to **Section 5. Resets**.

1.7.7 $\overline{\text{IRQ}}/V_{PP}$

This input pin drives the asynchronous maskable interrupt request (IRQ) function of the CPU. The IRQ interrupt function has a programmable mask option to select either negative edge-sensitive triggering or both negative edge-sensitive and low level-sensitive triggering. The IRQ input requires an external resistor to V_{DD} for wire-OR operation, if desired. If the $\overline{\text{IRQ}}$ pin is not used, it must be tied to the V_{DD} supply. The $\overline{\text{IRQ}}$ pin contains an internal Schmitt trigger as part of its input to improve noise immunity. Each of the PC0–PC7 I/O pins may be connected as an OR function with the IRQ interrupt function. This capability allows keyboard scan applications where the transitions on the I/O pins will behave the same as the $\overline{\text{IRQ}}$ pin. The edge or level sensitivity selected by a mask option for the $\overline{\text{IRQ}}$ pin does not apply to the port C I/O pin interrupt. The I/O pin interrupt is always negative edge-sensitive. See **Section 4. Interrupts** for more details on the interrupts.

This pin is also used to provide the programming voltage for the EPROM array. See **Section 12. EPROM and EEPROM** for more details on EPROM programming.

1.7.8 PA0–PA6

These seven I/O lines comprise port A. The state of any pin is software programmable, and all port A lines are configured as inputs during power-on or reset. See **Section 7. Parallel Input/Output (I/O)** for more details on the I/O ports.

1.7.9 PB0–PB3 (SPI Pins), PB4/PWMA, PB5/PWMB, PB6/TCMP, and PB7/TCAP

These eight I/O lines comprise port B. The state of any pin is software programmable, and all port B lines are configured as inputs during power-on or reset. See **Section 7. Parallel Input/Output (I/O)** for more details on the I/O ports.

PB0–PB3 are shared SPI functions. See **Section 10. Serial Peripheral Interface (SPI)** for more details concerning the operation of the SPI and configuration of these pins.

PB6 and PB7 are also shared with timer functions. The TCAP pin controls the input capture feature for the on-chip 16-bit timer. The TCMP pin provides an output for the output compare feature of the on-chip 16-bit timer. See **Section 9. 16-Bit Timer** for more details on the operation of the timer subsystem.

PB4 and PB5 are shared with the PWM output pins (PWMA and PWMB). See **Section 11. Pulse Width Modulators (PWMs)** for more details on the operation of the PWMs.

1.7.10 PC0–PC7

These eight I/O lines comprise port C. The state of any pin is software programmable and all port C lines are configured as inputs during power-on or reset. All eight pins are connected via an internal gate to the IRQ interrupt function. When the IRQ interrupt function is enabled, all the port C pins will act as negative edge-sensitive IRQ sources. See **Section 7. Parallel Input/Output (I/O)** for more details on the I/O ports.

1.7.11 PD0–PD4/AD0–AD4

When the A/D converter is disabled, PD0–PD4 are general-purpose input pins. The A/D converter is disabled upon exiting from reset. When the A/D converter is enabled, one of these pins is the analog input to the A/D converter. The A/D control register contains control bits to direct which of the analog inputs are to be converted at any one time. A digital read of this pin when the A/D converter is enabled results in a read of logical 0 from the selected analog pin. A digital read of the remaining pins gives their correct (digital) values. See **Section 13. Analog-to-Digital (A/D) Converter** for more details on the operation of the A/D subsystem.

1.7.12 TXP and RXP

These pins provide the I/O interface for the byte data link controller (BDLC) subsystem. See **Section 14. Byte Data Link Controller – Digital (BDLC–D)** for more details on the operation of the BDLC.

1.7.13 I_{MAX}

This pin is used to define the maximum coil current in the gauges by connecting a resistor from this pin (R_{MAX}) to ground as shown in **15.7 Coil Sequencer and Control**.

1.7.14 V_{PGC}

This pin is the gauge power control pin for the external pass device. Refer to **15.7 Coil Sequencer and Control**.

1.7.15 V_{GSUP}

This pin is the regulated gauge voltage input. Refer to **15.7 Coil Sequencer and Control**.

1.7.16 V_{SSG}

Two pins are provided for a separate gauge driver ground, V_{SSG}. Used as the current return only for the coil driver circuitry, it is a high-current pin.

1.7.17 V_{GVREF}

This pin is the feedback pin for the gauge power regulator. External resistors as shown in **Figure 15-14. Sample Gauge Connections to the MC68HC705V12** are used to set the gauge input voltage at pin V_{GSUP}.

1.7.18 MAJA(B)1+, MAJA(B)1–, MAJA(B)2+, and MAJA(B)2–

These pins are the full H-bridge coil driver pins. The A or B refer to pins associated with major gauge A or gauge B, and pin 1+/– or pin 2+/– refer to coil 1 or coil 2 of that major gauge and the direction of current flow. Refer to **15.3 Gauge System Overview** for more details on the operation of these pins.

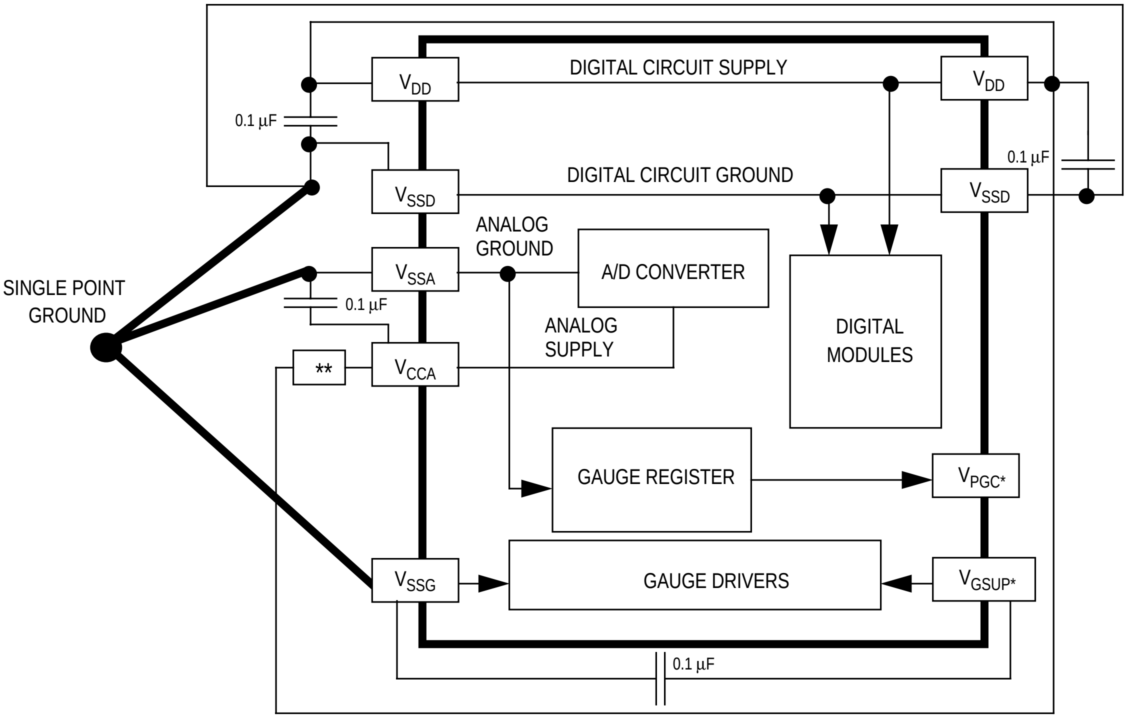
1.7.19 MINA(B,C,D)1, MINA(B,C,D)2+, and MINA(B,C,D)2–

MINA(B,C,D)2+ and MINA(B,C,D)2– are the full H-bridge driver pins used with or for the minor gauges. These pins allow the coil current to be reversed for movement of gauge pointer from 0 to 180 degrees.

MINA(B,C,D)1 is the low-side driver pin used with the minor gauges. The current flow through the coil is restricted to one direction. Refer to **15.3 Gauge System Overview** for more details on the operation of these pins.

1.8 Power Supply Pin Connections

Refer to **Figure 1-4** for a supply decoupling diagram.



* Refer to **Section 15. Gauge Drivers** for decoupling recommendations.
**Optional supply isolation circuit

Figure 1-4. Supply Decoupling Diagram

1.9 Decoupling Recommendations

To provide effective decoupling and to reduce radiated RF (radio frequency) emissions, small decoupling capacitors must be located as close to the supply pins as possible. The self-inductance of these capacitors and the parasitic inductance and capacitance of the interconnecting traces determine the self-resonant frequency of the decoupling network. A frequency that is too low will reduce decoupling effectiveness and could increase radiated RF emissions from the system. A low value capacitor (470 pF to 0.01 μF) placed in parallel with the other capacitors will improve the bandwidth and effectiveness of the network.

1.9.1 V_{DD} to V_{SSD} — MCU Internal Digital Power Decoupling

Decouple with a 0.1 μF ceramic or polystyrene cap. If the self-resonance frequency of the decoupling circuit (assume 4 nH per bond wire) is too low, add a 0.01 μF or smaller cap in parallel to increase the bandwidth of the decoupling network. Place the smaller cap closest to the V_{DD} and V_{SSD} pins.

1.9.2 V_{CCA} to V_{SSA} — Analog Subsystem Power Supply Pins

These pins are internally isolated from the digital V_{DD} and V_{SS} supplies. The V_{SSA} pin provides a ground return for the A/D subsystem and portions of the gauge subsystem. The analog supply pins should be appropriately filtered to prevent any external noise affecting the analog subsystems. The V_{SSA} pin should be brought together with the digital ground at a single point which has a low (HF) impedance to ground to prevent common mode noise problems. If this is not practical, then the V_{SSA} PCB traces should be routed in such a manner that digital ground return current is impeded from passing through the analog input ground reference as shown in **Figure 1-5**.

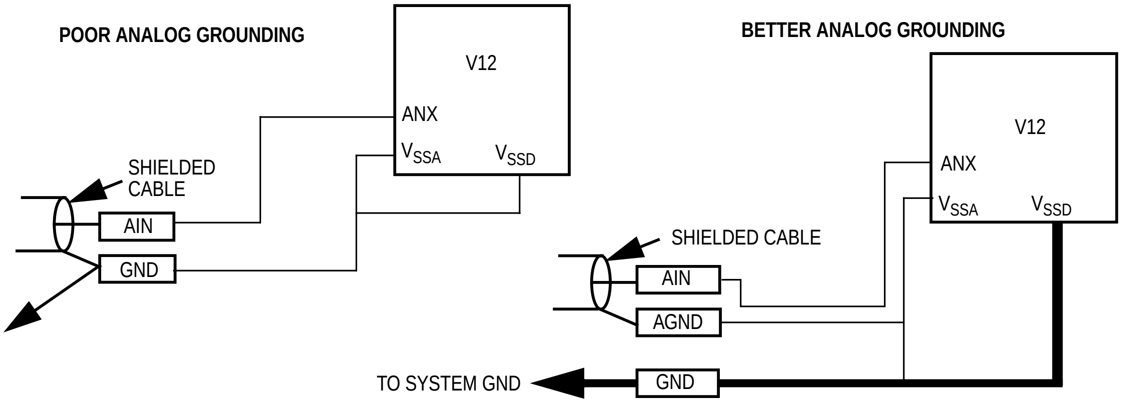


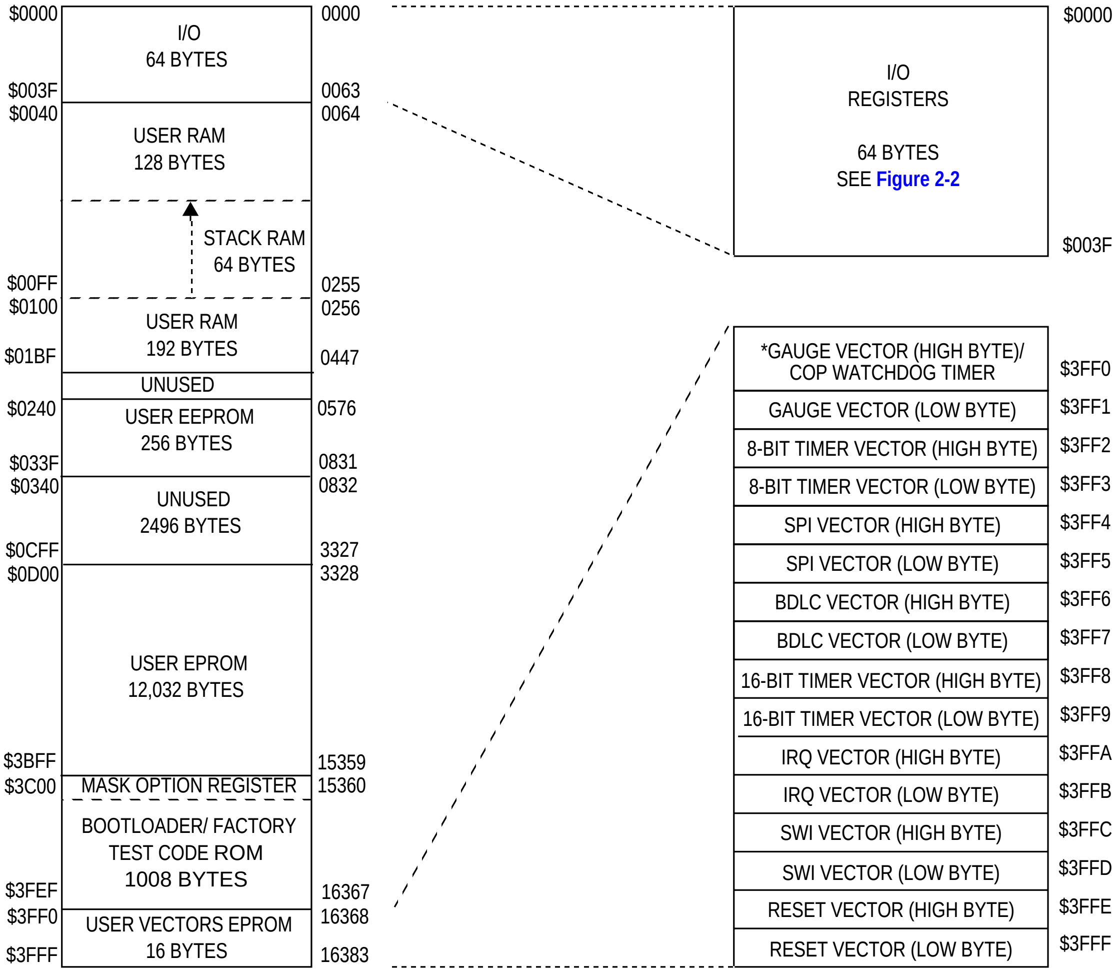
Figure 1-5. Single-Sided PCB Example

Section 2. Memory Map

2.1 Contents

2.2 Introduction

When the MC68HC705V12 is in the single-chip mode, the input/output (I/O) and peripherals, user random-access memory (RAM), electrically erasable programmable read-only memory (EEPROM), and user erasable programmable read-only memory (EPROM) are all active as shown in **Figure 2-1**.



*Reading $3FF0 returns the gauge vector EPROM byte.
Writing a 0 to $3FF0, bit 0, resets the COP.

Figure 2-1. MC68HC705V12 Single-Chip Mode Memory Map

2.3 I/O and Control Registers

The I/O and control registers reside in locations $0000–$003F. The overall organization of these registers is shown in **Figure 2-2**. The bit assignments for each register are shown in **Figure 2-3**. Reading from unimplemented bits will return unknown states, and writing to unimplemented bits will be ignored.

Port A Data Register	$0000	Gauge Enable Register — GER	$0020
Port B Data Register	$0001	Scan Status & Control Reg — SSCR	$0021
Port C Data Register	$0002	Magnitude Register — MAJA1	$0022
Port D Data Register	$0003	Magnitude Register — MAJA2	$0023
Port A Data Direction Register	$0004	Magnitude Register — MAJB1	$0024
Port B Data Direction Register	$0005	Magnitude Register — MAJB2	$0025
Port C Data Direction Register	$0006	Magnitude Register — MINA1	$0026
Unused	$0007	Magnitude Register — MINA2	$0027
8-Bit Timer Status and Control	$0008	Magnitude Register — MINB1	$0028
8-Bit Timer Counter Register	$0009	Magnitude Register — MINB2	$0029
SPI Control Register	$000A	Magnitude Register — MINC1	$002A
SPI Status Register	$000B	Magnitude Register — MINC2	$002B
SPI Data Register	$000C	Magnitude Register — MIND1	$002C
EPROM Program Register	$000D	Magnitude Register — MIND2	$002D
Unimplemented	$000E	Current Direction Register — DMAJA	$002E
Unimplemented	$000F	Current Direction Register — DMAJB	$002F
Unimplemented	$0010	Current Direction Register — DMINA	$0030
Unimplemented	$0011	Current Direction Register — DMINB	$0031
16-Bit Timer Control Register	$0012	Current Direction Register — DMINC	$0032
16-Bit Timer Status Register	$0013	Current Direction Register — DMIND	$0033
Input Capture Register (High)	$0014	Reserved	$0034
Input Capture Register (Low)	$0015	Miscellaneous Register	$0035
Output Compare Register (High)	$0016	PWMA Data Register	$0036
Output Compare Register (Low)	$0017	PWMA Control Register	$0037
16-Bit Timer Count Register (High)	$0018	PWMB Data Register	$0038
16-Bit Timer Count Register (Low)	$0019	PWMB Control Register	$0039
Alternate Count Register (High)	$001A	BDLC Control Register 1	$003A
Alternate Count Register (Low)	$001B	BDLC Control Register 2	$003B
EEPROM Program Register	$001C	BDLC State Vector Register	$003C
A/D Data Register	$001D	BDLC Data Register	$003D
A/D Status and Control Register	$001E	BDLC Analog Roundtrip Delay Register	$003E
IRQ Status and Control Register	$001F	Reserved	$003F

Figure 2-2. MC68HC705V12 I/O Registers Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PORTA) See page 82.	Read:	0	PA6	PA5	PA4	PA3	PA2	PA1	PA0
		Write:	▨							
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PORTB) See page 83.	Read:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PORTC) See page 84.	Read:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PORTD) See page 86.	Read:	0	0	0	PD4	PD3	PD2	PD1	PD0
		Write:	▨	▨	▨					
		Reset:	Unaffected by reset							
$0004	Port A Data Direction (DDRA) See page 82.	Read:	0	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0
$0005	Port B Data Direction (DDRB) See page 83.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006	Port C Data Direction (DDRC) See page 84.	Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$0008	Core Timer Status and Control Register (CTSCR) See page 89.	Read:	CTOF	RTIF	TOFE	RTIE	0	0	RT1	RT0
		Write:	▨	▨			TOFC	RTFC		
		Reset:	0	0	0	0	0	0	1	1

▨ = Unimplemented R = Reserved U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 1 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0009	Core Timer Counter Register (CTCR) See page 92.	Read:	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$000A	SPI Control Register (SPCR) See page 107.	Read:	SPIE	SPE	0	MSTR	CPOL	CPHA	SPR1	SPR0
		Write:			▨					
		Reset:	0	0	0	0	0	1	U	U
$000B	SPI Status Register (SPSR) See page 109.	Read:	SPIF	WCOL	0	MODF	0	0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$000C	SPI Data Register (SPDR) See page 110.	Read:	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
		Write:								
		Reset:	Unaffected by reset							
$000D	EPROM Programming Register (EPROG) See page 124.	Read:	MORON	0	0	0	0	ELAT	0	EPGM
		Write:		R	R	R	R		▨	
		Reset:	0	0	0	0	0	0	0	0
$000E	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$000F	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$0010	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$0011	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$0012	16-Bit Timer Control Register (TMRCR) See page 98.	Read:	ICIE	OCIE	TOIE	0	0	TON	IEDG	OLVL
		Write:				▨	▨			
		Reset:	0	0	0	0	0	0	0	0
$0013	16-Bit Timer Status Register (TMRSR) See page 99.	Read:	ICF	OCF	TOF	0	0	0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 2 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0014	Input Capture MSB Register (TCAPH) See page 97.	Read:	IC15	IC14	IC13	IC12	IC11	IC10	IC9	IC8
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	Unaffected by reset							
$0015	Input Capture LSB Register (TCAPL) See page 97.	Read:	IC7	IC6	IC5	IC4	IC3	IC2	IC1	IC0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	Unaffected by reset							
$0016	Output Compare MSB Register (TCMPH) See page 96.	Read:	OC15	OC14	OC13	OC12	OC11	OC10	OC9	OC8
		Write:								
		Reset:	Unaffected by reset							
$0017	Output Compare LSB Register (TCMPL) See page 96.	Read:	OC7	OC6	OC5	OC4	OC3	OC2	OC1	OC0
		Write:								
		Reset:	Unaffected by reset							
$0018	Timer Counter MSB Register (TCNTH) See page 94.	Read:	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8
		Write:								
		Reset:	Unaffected by reset							
$0019	Timer Counter LSB Register (TCNTL) See page 94.	Read:	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
		Write:								
		Reset:	Unaffected by reset							
$001A	Alternate Counter MSB Register (ALTCNTH) See page 94.	Read:	AC15	AC14	AC13	AC12	AC11	AC10	AC9	AC8
		Write:								
		Reset:	Unaffected by reset							
$001B	Alternate Counter LSB Register (ALTCNTL) See page 94.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$001C	EEPROM Programming Register (EEPROG) See page 127.	Read:	0	CPEN	0	ER1	ER0	EELAT	EERC	EEPGM
		Write:	▨		▨					
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   R = Reserved   U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 3 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001D	A/D Data Register (ADDR) See page 135.	Read:	D7	D6	D5	D4	D3	D2	D1	D0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	Unaffected by reset							
$001E	A/D Status and Control Register (ADSCR) See page 134.	Read:	COCO	ADRC	ADON	CH4	CH3	CH2	CH1	CH0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0
$001F	IRQ Status and Control Register (ISCR) See page 62.	Read:	IRQE	0	IPCE	0	IRQF	0	IPCF	0
		Write:		▨		▨	▨	▨	▨	IRQA
		Reset:	1	0	0	0	0	0	0	0
$0020	Gauge Enable Register (GER) See page 192.	Read:	MJAON	MJBON	MIAON	MIBON	MICON	MIDON	CMPS	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0021	Scan Status and Control Register (SSCR) See page 202.	Read:	SYNIE	SYNF	0	R	GCS1	GCS0	SCNS	AUTOS
		Write:		▨	SYNR					
		Reset:	0	0	0	0	0	0	0	0
$0022	MAJA1 Magnitude Register (MAJA1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	MAJA2 Magnitude Register (MAJA2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	MAJB1 Magnitude Register (MAJB1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0025	MAJB2 Magnitude Register (MAJB2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   R = Reserved   U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 4 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0026	MINA1 Magnitude Register (MINA1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0027	MINA2 Magnitude Register (MINA2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0028	MINB1 Magnitude Register (MINB1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0029	MINB2 Magnitude Register (MINB2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002A	MINC1 Magnitude Register (MINC1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002B	MINC2 Magnitude Register (MINC2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002C	MIND1 Magnitude Register (MIND1) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002D	MIND2 Magnitude Register (MIND2) See page 193.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002E	MAJA Current Direction Register (DMAJA) See page 196.	Read:	R	R	R	R	R	0	DMJA1	DMJA2
		Write:								
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented　　R = Reserved　　U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 5 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002F	MAJB Current Direction Register (DMAJB) See page 196.	Read:	0	0	0	0	0	0	DMJB1	DMJB2
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0030	MINA Current Direction Register (DMINA) See page 197.	Read:	0	0	0	0	0	0	0	DMIA
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0031	MINB Current Direction Register (MINB) See page 197.	Read:	0	0	0	0	0	0	0	DMIB
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0032	MINC Current Direction Register (DMINC) See page 198.	Read:	0	0	0	0	0	0	0	DMIC
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0033	MIND Current Direction Register (DMID) See page 198.	Read:	0	0	0	0	0	0	0	DMID
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0034	Reserved		R	R	R	R	R	R	R	R
$0035	Miscellaneous Register (MISC) See page 48.	Read:	0	OCE	0	0	0	0	0	0
		Write:	▨		▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0036	PWMA Data Register (PWMAD) See page 119.	Read:	POLA	0	D5	D4	D3	D2	D1	D0
		Write:		▨						
		Reset:	0	0	U	U	U	U	U	U
$0037	PWMA Control Register (PWMAC) See page 117.	Read:	PSA1A	PSA0A	0	0	PSB3A	PSB2A	PSB1A	PSB0A
		Write:			▨	▨				
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   R = Reserved   U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 6 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
	PWMB Data Register	Read:	POLB	0	D5	D4	D3	D2	D1	D0
$0038	(PWMBD)	Write:		▨						
	See page 119.	Reset:	0	0	U	U	U	U	U	U
	PWMB Control Register	Read:	PSA1B	PSA0B	0	0	PSB3B	PSB2B	PSB1B	PSB0B
$0039	(PWMBC)	Write:			▨	▨				
	See page 118.	Reset:	0	0	0	0	0	0	0	0
	BDLC Control 1 Register	Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
$003A	(BCR1)	Write:					R	R		
	See page 169.	Reset:	1	1	1	0	0	0	0	0
	BDLC Control 2 Register	Read:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
$003B	(BCR2)	Write:								
	See page 171.	Reset:	1	1	0	0	0	0	0	0
	BDLC State Vector Register	Read:	0	0	I3	I2	I1	I0	0	0
$003C	(BSVR)	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	See page 179.	Reset:	0	0	0	0	0	0	0	0
	BDLC Data Register	Read:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
$003D	(BDR)	Write:								
	See page 181.	Reset:	Indeterminate after reset							
	BDLC Analog and Roundtrip	Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
$003E	Delay Register (BARD)	Write:			▨	▨				
	See page 167.	Reset:	1	1	0	0	0	1	1	1
$003F	Reserved		R	R	R	R	R	R	R	R

▨ = Unimplemented   R = Reserved   U = Unaffected

Figure 2-3. I/O and Control Registers (Sheet 7 of 7)

2.4 RAM

The total RAM consists of 384 bytes (including the stack). The stack begins at address $00FF and proceeds down to $00C0 (64 bytes). Using the stack area for data storage or temporary work locations requires care to prevent it from being overwritten due to stacking from an interrupt or subroutine call.

NOTE: *The stack is located in the middle of the RAM address space. Data written to addresses within the stack address range can be overwritten during stack activity.*

2.5 Boot ROM

The boot ROM space in the MC68HC705V12 consists of 1008 bytes including EPROM bootloader code, EEPROM test code, burn-in code, and 16 bytes of bootloader vectors. The mask option register (MOR) is located in this space.

2.6 EPROM

There are 12,032 bytes of user EPROM and 16 bytes of EPROM for user vectors and the computer operating properly (COP) update location. Refer to **Section 12. EPROM and EEPROM** for programming details.

2.7 EEPROM

This device contains 256 bytes of EEPROM. Programming the EEPROM is performed by the user on a single-byte basis by manipulating the programming register located at address $001C. Refer to **Section 12. EPROM and EEPROM** for programming details.

2.8 Miscellaneous Register

The miscellaneous register (MISC) is located at $0035.

Address: $0035

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

▨ = Unimplemented

Figure 2-4. Miscellaneous Register (MISC)

OCE — Output Compare Enable Bit

This bit controls the function of the PB6 pin.

0 = PB6 functions as a normal I/O pin.

1 = PB6 becomes the TCMP output pin for the 16-bit timer.

See **Section 9. 16-Bit Timer** for a description of the TCMP function.

Section 3. Central Processor Unit (CPU)

3.1 Contents

3.2 Introduction

This section describes the central processor unit (CPU) registers.

3.3 CPU Registers

Figure 3-1 shows the five CPU registers. CPU registers are not part of the memory map.

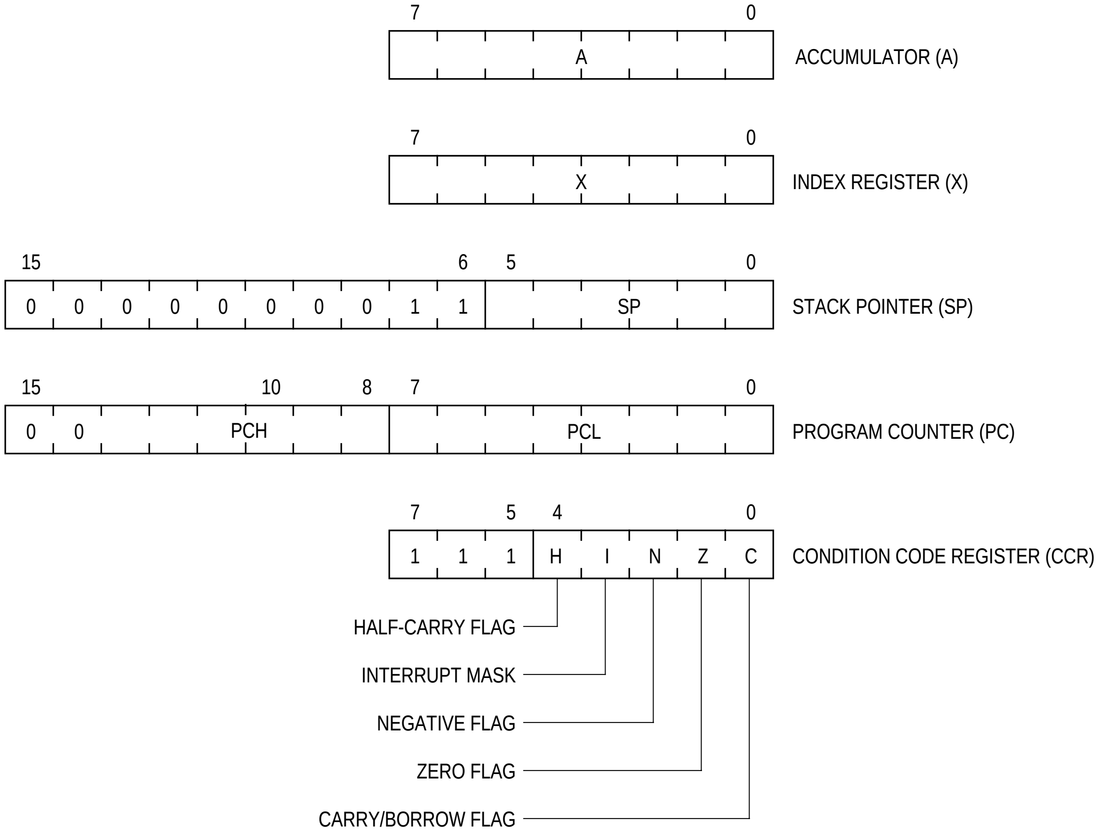


Figure 3-1. Programming Model

3.3.1 Accumulator

The accumulator (A) is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and results of arithmetic and non-arithmetic operations.

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

Figure 3-2. Accumulator (A)

3.3.2 Index Register

In the indexed addressing modes, the CPU uses the byte in the index register (X) to determine the conditional address of the operand.

The 8-bit index register can also serve as a temporary data storage location.

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

Figure 3-3. Index Register (X)

3.3.3 Stack Pointer

The stack pointer (SP) is a 16-bit register that contains the address of the next location on the stack. During a reset or after the reset stack pointer (RSP) instruction, the stack pointer is preset to $00FF. The address in the stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

The 10 most significant bits of the stack pointer are permanently fixed at 000000011, so the stack pointer produces addresses from $00C0 to $00FF. If subroutines and interrupts use more than 64 stack locations, the stack pointer wraps around to address $00FF and begins writing over the previously stored data. A subroutine uses two stack locations. An interrupt uses five locations.

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

Figure 3-4. Stack Pointer (SP)

3.3.4 Program Counter

The program counter (PC) is a 16-bit register that contains the address of the next instruction or operand to be fetched. The two most significant bits of the program counter are ignored internally and appear as 00.

Normally, the address in 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.

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0											5			
Reset	0	0	Loaded with vectors from $3FF3 and $3FFF													

Figure 3-5. Program Counter (PC)

3.3.5 Condition Code Register

The condition code register (CCR) is an 8-bit register whose three most significant bits are permanently fixed at 111. The condition code register contains the interrupt mask and four flags that indicate the results of the instruction just executed. The following paragraphs describe the functions of the condition code register.

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

▨ = Unimplemented U = Unaffected

Figure 3-6. Condition Code Register (CCR)

Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between bits 3 and 4 of the accumulator during an ADD or ADC operation. The half-carry flag is required for binary coded decimal (BCD) arithmetic operations.

Interrupt Mask

Setting the interrupt mask disables interrupts. If an interrupt request occurs while the interrupt mask is logic 0, the CPU saves the CPU registers on the stack, sets the interrupt mask, and then fetches the interrupt vector. If an interrupt request occurs while the interrupt mask is set, the interrupt request is latched. Normally, the CPU processes the latched interrupt as soon as the interrupt mask is cleared again.

A return-from-interrupt (RTI) instruction pulls the CPU registers from the stack, restoring the interrupt mask to its cleared state. After any reset, the interrupt mask is set and can be cleared only by a software instruction.

Negative Flag

The CPU sets the negative flag when an arithmetic operation, logical operation, or data manipulation produces a negative result.

Zero Flag

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

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 logical operations and data manipulation instructions also clear or set the carry/borrow flag.

3.4 Arithmetic/Logic Unit

The arithmetic/logic unit (ALU) performs the arithmetic and logical operations defined by the instruction set.

The binary arithmetic circuits decode instructions and set up the ALU for the selected operation. Most binary arithmetic is based on the addition algorithm, carrying out subtraction as negative addition. Multiplication is not performed as a discrete operation but as a chain of addition and shift operations within the ALU. The multiply instruction (MUL) requires 11 internal clock cycles to complete this chain of operations.

Section 4. Interrupts

4.1 Contents

4.2 Introduction

The MCU can be interrupted eight different ways:

1. Non-maskable software interrupt instruction (SWI)
2. External asynchronous interrupt (IRQ)
3. External interrupt via IRQ on PC0–PC7 (IRQ)
4. Internal 16-bit timer interrupt (TIMER)
5. Internal BDLC interrupt (BDLC)
6. Internal serial peripheral interface interrupt (SPI)
7. Internal 8-bit timer interrupt (CTIMER)
8. Internal gauge interrupt (GAUGE)

4.3 CPU Interrupt Processing

Interrupts cause the processor to save register contents on the stack and to set the interrupt mask (I bit) to prevent additional interrupts. Unlike reset, hardware interrupts do not cause the current instruction execution to be halted, but are considered pending until the current instruction is complete.

If interrupts are not masked (I bit in the condition code register (CCR) is clear) and the corresponding interrupt enable bit is set, then the processor will proceed with interrupt processing. Otherwise, the next instruction is fetched and executed. If an interrupt occurs, the processor completes the current instruction, then stacks the current CPU register states, sets the I bit to inhibit further interrupts, and finally checks the pending hardware interrupts. If more than one interrupt is pending after the stacking operation, the interrupt with the highest vector location shown in **Table 4-1** will be serviced first. The SWI is executed the same as any other instruction, regardless of the I-bit state.

When an interrupt is to be processed, the central processor unit (CPU) fetches the address of the appropriate interrupt software service routine from the vector table at locations $3FF0–$3FFF as defined in **Table 4-1**.

Table 4-1. Vector Address for Interrupts and Reset

Register	Flag Name	Interrupts	CPU Interrupt	Vector Address
N/A	N/A	Reset	RESET	$3FFE–$3FFF
N/A	N/A	Software	SWI	$3FFC–$3FFD
ISCR	IRQF/IPCF	External (IRQ and port C)	IRQ	$3FFA–$3FFB
TSR	TOF	Timer overflow	TIMER	$3FF8–$3FF9
TSR	OCF	Output compare	TIMER	$3FF8–$3FF9
TSR	ICF	Input capture	TIMER	$3FF8–$3FF9
BSVR	I3:I0	BDLC	BDLC	$3FF6–$3FF7
SPSR	SPIF	SPI	SPI	$3FF4–$3FF5
CTSCR	CTOF	Core timer overflow	CTIMER	$3FF2–$3FF3
CTSCR	RTIF	Real time	CTIMER	$3FF2–$3FF3
SSCR	SYNF	Gauge synchronize	GAUGE	$3FF0–$3FF1

Because the M68HC05 CPU does not support interruptible instructions, the maximum latency to the first instruction of the interrupt service routine must include the longest instruction execution time plus stacking overhead.

$$\text{Latency} = (\text{Longest instruction execution time} + 10) \text{ x } t_{\text{CYC}} \text{ seconds}$$

A return-from-interrupt (RTI) instruction is used to signify when the interrupt software service routine is completed. The RTI instruction causes the register contents to be recovered from the stack and normal processing to resume at the next instruction that was to be executed when the interrupt took place. **Figure 4-1** shows the sequence of events that occur during interrupt processing.

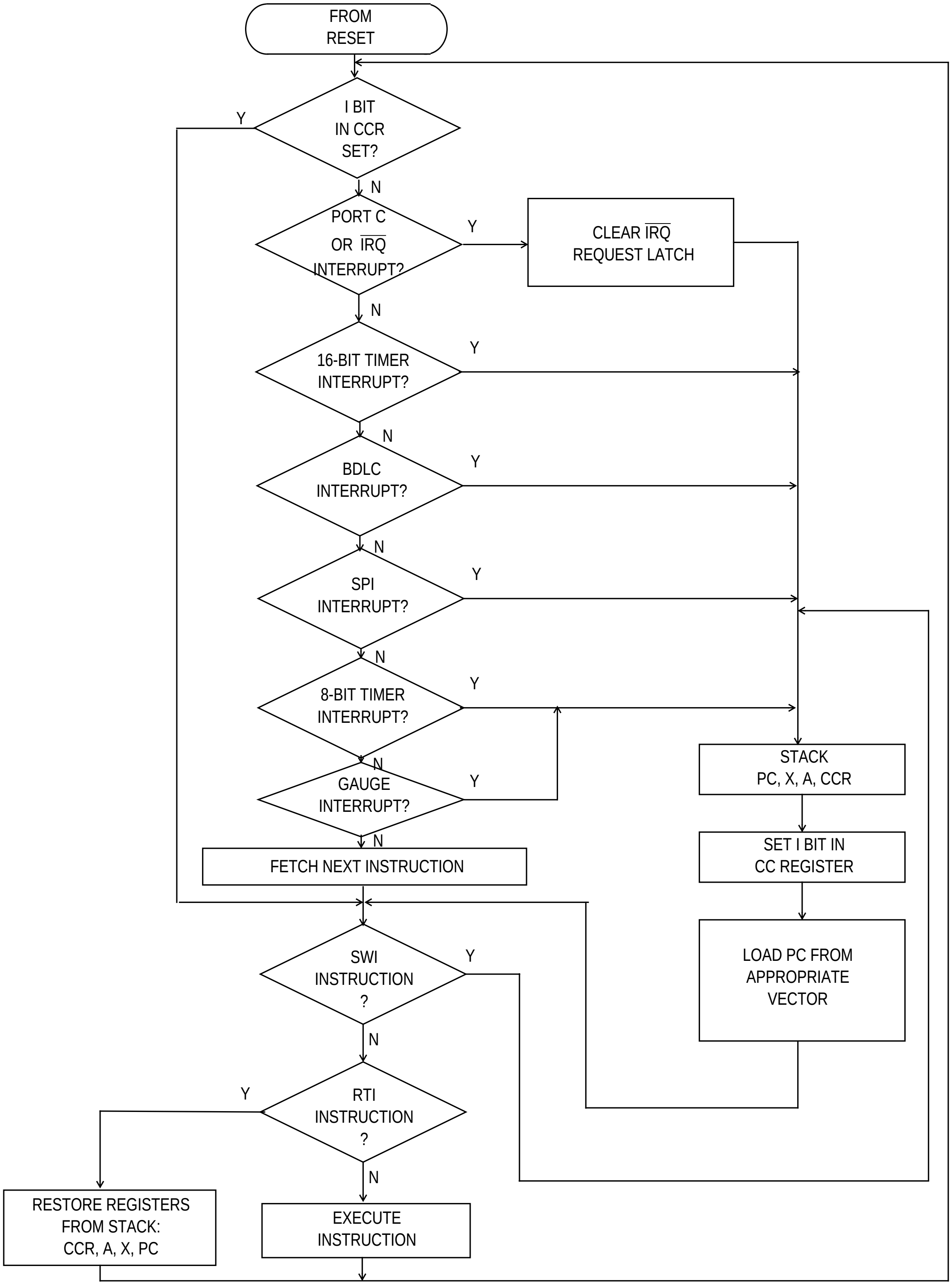


Figure 4-1. Interrupt Processing Flowchart

4.4 Reset Interrupt Sequence

The reset function is not in the strictest sense an interrupt; however, it is acted upon in a similar manner as shown in **Figure 4-1**. A low-level input on the $\overline{\text{RESET}}$ pin or internally generated RST signal causes the program to vector to its starting address which is specified by the contents of memory locations $3FFE and $3FFF. The I bit in the condition code register is also set. The MCU is configured to a known state during this type of reset as described in **Section 5. Resets**.

4.5 Software Interrupt (SWI)

The SWI is an executable instruction and a non-maskable interrupt since it is executed regardless of the state of the I bit in the CCR. If the I bit is zero (interrupts enabled), the SWI instruction executes after interrupts which were pending before the SWI was fetched or before interrupts generated after the SWI was fetched. The interrupt service routine address is specified by the contents of memory locations $3FFC and $3FFD.

4.6 Hardware Interrupts

All hardware interrupts except reset are maskable by the I bit in the CCR. If the I bit is set, all hardware interrupts (internal and external) are disabled. Clearing the I bit enables the hardware interrupts. Two types of hardware interrupts are explained in the following subsections.

4.7 External Interrupt (IRQ)

The $\overline{\text{IRQ}}$ pin provides an asynchronous interrupt to the CPU. A block diagram of the IRQ function is shown in **Figure 4-2**.

NOTE: *The BIH and BIL instructions will apply only to the level on the $\overline{\text{IRQ}}$ pin itself and not to the output of the logic OR function with the port C IRQ interrupts. The state of the individual port C pins can be checked by reading the appropriate port C pins as inputs.*

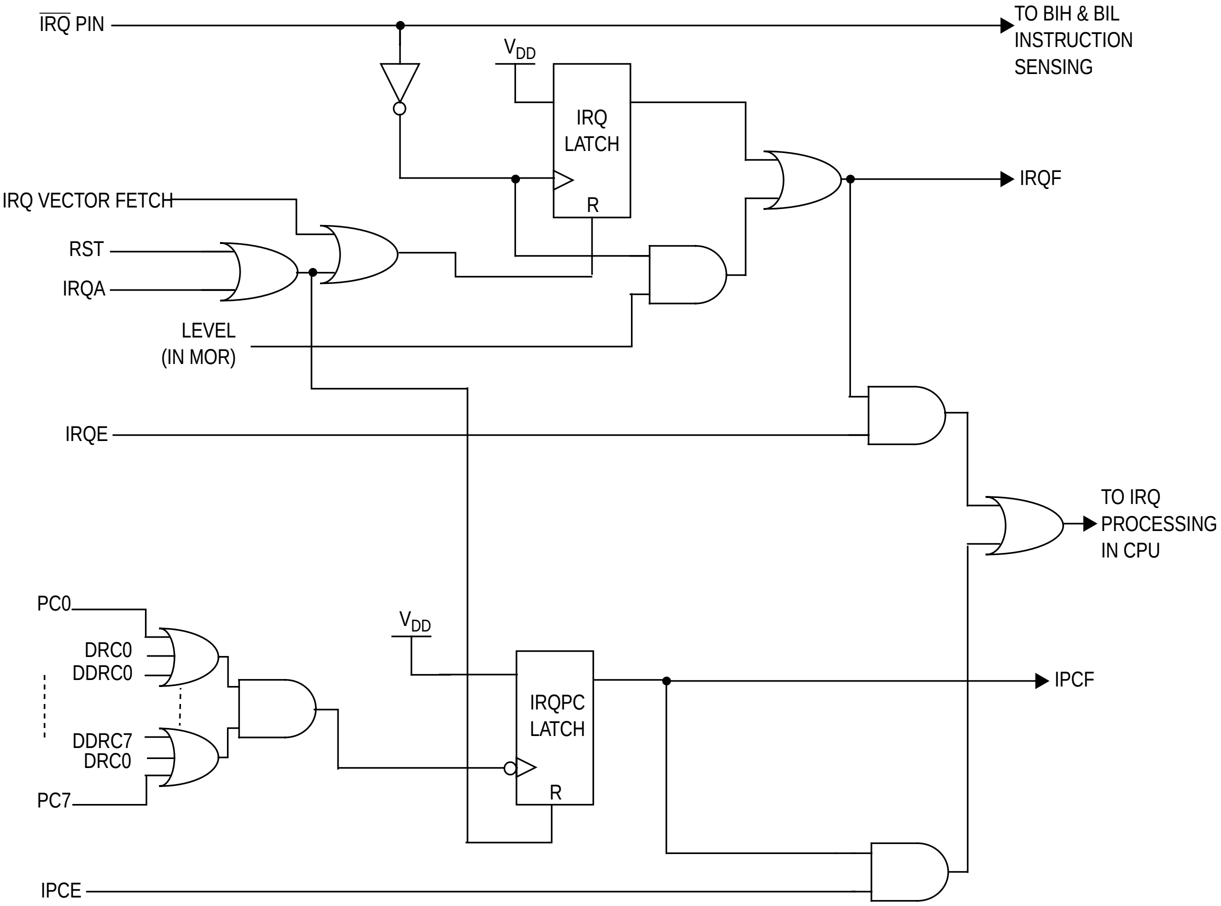


Figure 4-2. **IRQ Function Block Diagram**

The $\overline{\text{IRQ}}$ pin is one source of an external interrupt. All port C pins (PC0–PC7) act as other external interrupt sources. These sources have their own interrupt latch but are combined with IRQ into a single external interrupt request.

The port C interrupt sources are negative (falling) edge-sensitive only. Note that all port C pins are ANDed together to form the negative edge signal which sets the corresponding flag bits. A high-to-low transition on any port C pin configured as an interrupt input will, therefore, set the respective flag bit. If a port C pin is to be used as an interrupt input, the corresponding data direction and data bits must both be cleared. If either the pin is configured as an output or the data bit is set, a falling edge on the pin will not generate an interrupt. The $\overline{\text{IRQ}}$ pin interrupt source may be selected to be either edge sensitive or edge and level sensitive through a mask option or an MOR bit. If the edge-sensitive interrupt option is selected for the $\overline{\text{IRQ}}$ pin, only the IRQ latch output can activate an IRQF flag which creates an interrupt request to the CPU to generate the external interrupt sequence.

When edge sensitivity is selected for the IRQ interrupt, it is sensitive to these cases:

1. Falling edge on the $\overline{\text{IRQ}}$ pin
2. Falling edge on any port C pin with IRQ enabled

If the LEVEL select bit in the MOR is set, the active low state of the $\overline{\text{IRQ}}$ pin can also activate an IRQF flag which creates an IRQ request to the CPU to generate the IRQ interrupt sequence.

When edge and level sensitivity are selected for the IRQ interrupt, it is sensitive to these cases:

1. Low level on the $\overline{\text{IRQ}}$ pin
2. Falling edge on the $\overline{\text{IRQ}}$ pin
3. Falling edge on any port C pin with IRQ enabled

The IRQE enable bit controls whether an active IRQF flag ($\overline{\text{IRQ}}$ pin interrupt) can generate an IRQ interrupt sequence. The IPCE enable bit controls whether an active IPCF flag (port C interrupt) can generate an IRQ interrupt sequence. The IRQ interrupt is serviced by the interrupt service routine located at the address specified by the contents of $3FFA and $3FFB.

The IRQF latch is cleared automatically by entering the interrupt service routine to maintain compatibility with existing M6805 interrupt servicing protocol. To allow the user to identify the source of the interrupt, the port interrupt flag (IPCF) is not cleared automatically. This flag must be cleared within the interrupt handler prior to exit to prevent repeated re-entry. This is achieved by writing a logic 1 to the IRQA (IRQ acknowledge) bit, which will clear all pending IRQ interrupts, including a pending $\overline{\text{IRQ}}$ pin interrupt.

The interrupt request flag (IPCF) is read only and cannot be cleared by writing to it. The acknowledge flag always reads as a logic 0. Together, these features permit the safe use of read-modify-write instructions (for instance, BSET and BCLR) on the ISCR.

NOTE: *Although read-modify-write instruction use is allowable on the ISCR, shift operations should be avoided due to the possibility of inadvertently setting the IRQA.*

4.7.1 IRQ Status and Control Register

The IRQ interrupt function is controlled by the IRQ status and control register (ISCR) located at $001F. All unused bits in the ISCR will read as logic 0s. The IRQF bit is cleared and IRQE bit is set by reset.

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IRQE	0	IPCE	0	IRQF	0	IPCF	0
Write:		▨		▨	▨	▨	▨	IRQA
Reset:	1	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 4-3. IRQ Status and Control Register (ISCR)

IRQE — IRQ Interrupt Enable Bit

The IRQE bit controls whether the IRQF flag bit can or cannot initiate an IRQ interrupt sequence. If the IRQE enable bit is set, the IRQF flag bit can generate an interrupt sequence. If the IRQE enable bit is cleared, the IRQF flag bit cannot generate an interrupt sequence. Reset sets the IRQE enable bit, thereby enabling IRQ interrupts once the I bit is cleared. Execution of the STOP or WAIT instructions causes the IRQE bit to be set to allow the external IRQ to exit these modes. In addition, reset also sets the I bit, which masks all interrupt sources.

IPCE — Port C IRQ Interrupt Enable Bit

The IPCE bit controls whether the IPCF flag bit can or cannot initiate an IRQ interrupt sequence. If the IPCE enable bit is set, the IPCF flag bit will generate an interrupt sequence. If the IPCE enable bit is cleared, the IPCF flag bit will not generate an interrupt sequence. Reset clears the IPCE enable bit, thereby disabling port C IRQ interrupts. In addition, reset also sets the I bit, which masks all interrupt sources. Execution of the STOP or WAIT instructions does not affect the IPCE bit.

NOTE: *The IPCE mask bit must be set prior to entering stop or wait modes if port IRQ interrupts are to be enabled.*

IRQF — IRQ Interrupt Request Bit

The IRQF flag bit indicates that an IRQ request is pending. Writing to the IRQF flag bit will have no effect on it. The IRQF flag bit is cleared when the IRQ vector is fetched prior to the service routine being entered. The IRQF flag bit also can be cleared by writing a logic one to the IRQA acknowledge bit to clear the IRQ latch. In this way, any additional IRQF flag bit that is set while in the service routine can be ignored by clearing the IRQF flag bit before exiting the service routine. If the additional IRQF flag bit is not cleared in the IRQ service routine and the IRQE enable bit remains set, the CPU will re-enter the IRQ interrupt sequence continuously until either the IRQF flag bit or the IRQE enable bit is clear. This flag can be set only when the IRQE enable is set. The IRQ latch is cleared by reset.

IPCF — Port C IRQ Interrupt Request Bit

The IPCF flag bit indicates that a port C IRQ request is pending. Writing to the IPCF flag bit will have no effect on it. The IPCF flag bit must be cleared by writing a logic 1 to the IRQA acknowledge bit. If the IPCF bit is not cleared via IRQA, the CPU will re-enter the IRQ interrupt sequence continuously until either the IPCF flag bit or the IPCE enable bit is clear. This bit is operational regardless of the state of the IPCE bit. The IPCF bit is cleared by reset.

IRQA — IRQ Interrupt Acknowledge Bit

The IRQA acknowledge bit clears an IRQ interrupt by clearing the IRQF and IPCF bits. This is achieved by writing a logic 1 to the IRQA acknowledge bit. Writing a logic 0 to the IRQA acknowledge bit will have no effect on the any of the IRQ bits. If either the IRQF or IPCF bit is not cleared within the IRQ service routine, then the CPU will re-enter the IRQ interrupt sequence continuously until the IRQ flag bits are all cleared. The IRQA is useful for cancelling unwanted or spurious interrupts which may have occurred while servicing the initial IRQ interrupt.

NOTE: *The IRQ flag is cleared automatically during the IRQ vector fetch. The IRQPC latch is not cleared automatically (to permit interrupt source differentiation as long as the Interrupt source is present) and must be cleared from within the IRQ service routine.*

4.7.2 External Interrupt Timing

If the interrupt mask bit (I bit) of the CCR is set, all maskable interrupts (internal and external) are disabled. Clearing the I bit enables interrupts. The interrupt request is latched immediately following the falling edge of the IRQ source. It is then synchronized internally and serviced as specified by the contents of $3FFA and $3FFB. The IRQ timing diagram is shown in **Figure 4-4**.

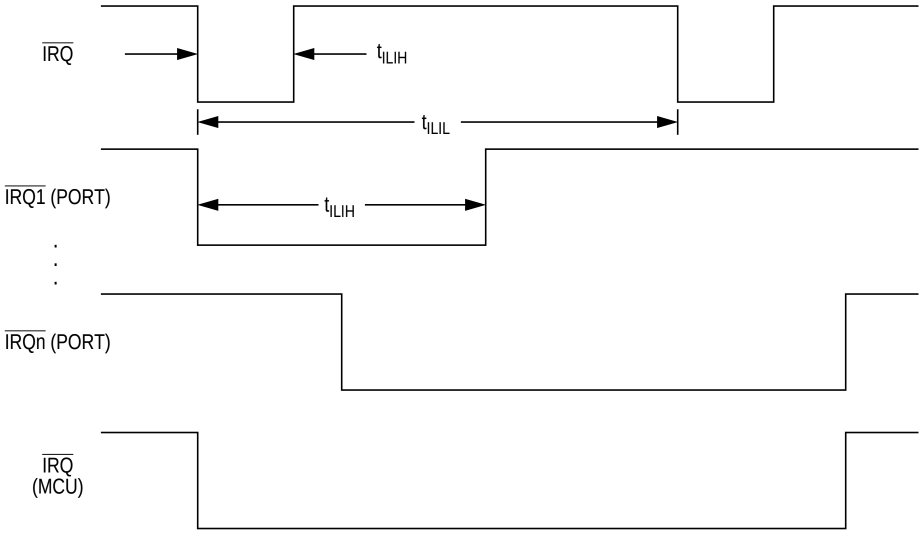


Figure 4-4. External Interrupts Timing Diagram

Either a level-sensitive and edge-sensitive trigger or an edge-sensitive-only trigger is available as a mask option for the $\overline{\text{IRQ}}$ pin only.

4.8 16-Bit Timer Interrupt

Three different timer interrupt flags cause a 16-bit timer interrupt whenever they are set and enabled. The interrupt flags are in the timer status register (TSR), and the enable bits are in the timer control register (TCR). Any of these interrupts will vector to the same interrupt service routine, located at the address specified by the contents of memory location $3FF8 and $3FF9.

4.9 BDLC Interrupt

The interrupt service routine is located at the address specified by the contents of memory location $3FF6 and $3FF7.

4.10 SPI Interrupt

Two different SPI interrupt flags cause an SPI interrupt whenever they are set and enabled. The interrupt flags are in the SPI status register (SPSR), and the enable bits are in the SPI control register (SPCR). Either of these interrupts will vector to the same interrupt service routine, located at the address specified by the contents of memory location $3FF4 and $3FF5.

4.11 8-Bit Timer Interrupt

This timer can create two types of interrupts.

- A timer overflow interrupt will occur whenever the 8-bit timer rolls over from $FF to $00 and the enable bit TOFE is set.
- A real-time interrupt will occur whenever the programmed time elapses and the enable bit RTIE is set.

The real-time interrupt will vector to the interrupt service routine located at the address specified by the contents of memory location $3FF2 and $3FF3.

4.12 Gauge Synchronize Interrupt

This interrupt service routine is located at the address specified by the contents of memory location $3FF0 and $3FF1. See **15.6.2 Current Magnitude Registers** for further details.

4.13 Stop Mode and Wait Mode

All modules which are capable of generating interrupts in stop mode or wait mode will be allowed to do so if the module is configured properly. The I bit is cleared automatically when stop or wait mode is entered. Interrupts detected on port C are recognized in stop or wait mode if port C interrupts are enabled.

Section 5. Resets

5.1 Contents

5.2 Introduction

The MCU can be reset from six sources:

- One external input
- Five internal restart conditions

The $\overline{\text{RESET}}$ pin is an input with a Schmitt trigger as shown in **Figure 5-1**. All the internal peripheral modules will be reset by the internal reset signal (RST). Refer to **Figure 5-2** for reset timing details.

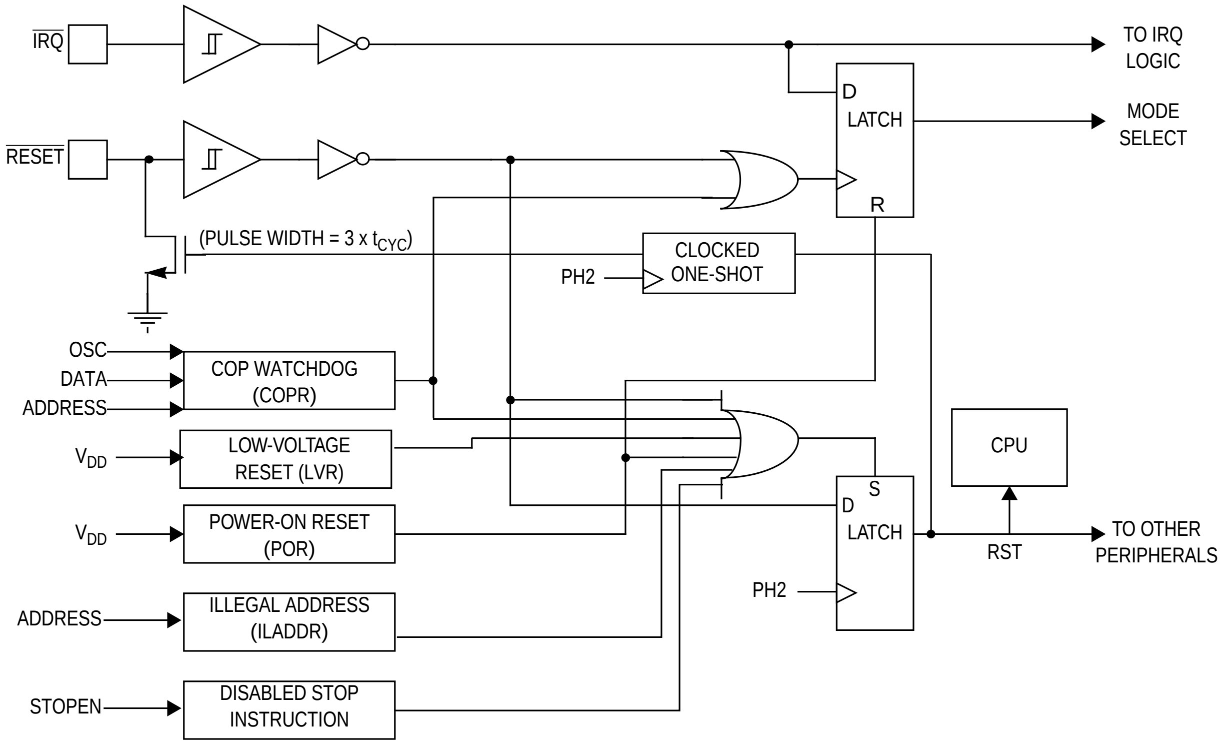


Figure 5-1. Reset Block Diagram

5.3 External Reset ($\overline{RESET}$)

The $\overline{RESET}$ pin is the only external source of a reset. This pin is connected to a Schmitt trigger input gate to provide an upper and lower threshold voltage separated by a minimum amount of hysteresis. This external reset occurs whenever the $\overline{RESET}$ pin is pulled below the lower threshold and remains in reset until the $\overline{RESET}$ pin rises above the upper threshold. This active low input will generate the RST signal and reset the CPU and peripherals.

NOTE: *Activation of the RST signal is generally referred to as reset of the device, unless otherwise specified.*

The $\overline{RESET}$ pin can also act as an open drain output. It will be pulled to a low state by an internal pulldown that is activated by any reset source. This reset pulldown device will be asserted only for three to four cycles of the internal clock, f_{OP}, or as long as an internal reset source is asserted. When the external $\overline{\text{RESET}}$ pin is asserted, the pulldown device will be turned on for only the three to four internal clock cycles.

5.4 Internal Resets

The five internally generated resets are:

- Initial power-on reset (POR) function
- Computer operating properly reset (COPR)
- Illegal address detector
- Low-voltage reset (LVR)
- Disabled STOP instruction

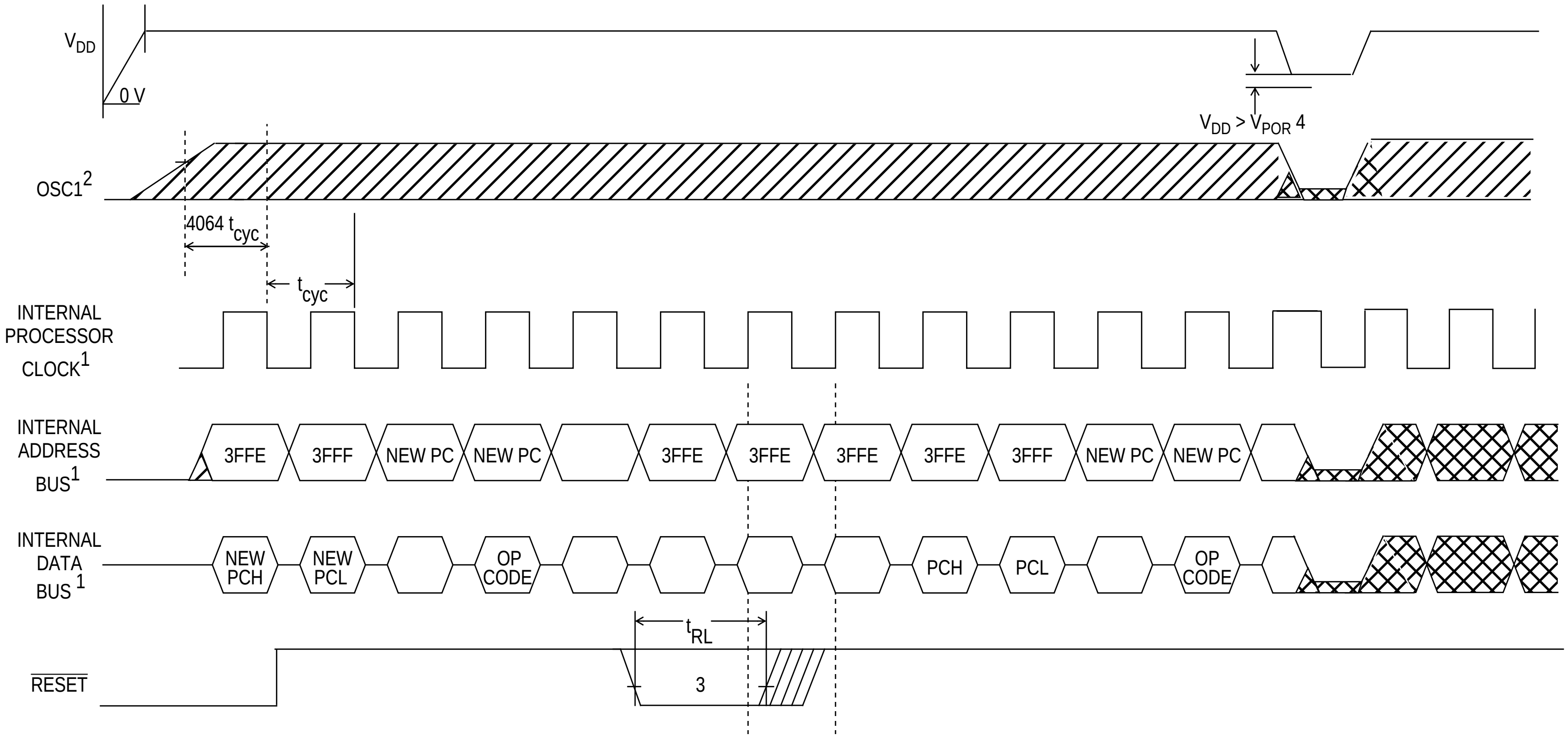
All internal resets will also assert (pull to logic 0) the external $\overline{\text{RESET}}$ pin for the duration of the reset or three to four internal clock cycles, whichever is longer.

5.4.1 Power-On Reset (POR)

The internal POR is generated on power-up to allow the clock oscillator to stabilize. The POR is strictly for power turn-on conditions and is not able to detect a drop in the power supply voltage (brown-out). There is an oscillator stabilization delay of 4064 internal processor bus clock cycles (PH2) after the oscillator becomes active.

The POR will generate the RST signal which will reset the CPU. If any other reset function is active at the end of this 4064-cycle delay, the RST signal will remain in the reset condition until the other reset condition(s) end.

POR will activate the $\overline{\text{RESET}}$ pin pulldown device connected to the pin. V_{DD} must drop below V_{POR} for the internal POR circuit to detect the next rise of V_{DD}.



Notes:

1. Internal timing signal and bus information are not available externally.
2. OSC1 line is not meant to represent frequency. It is only used to represent time.
3. The next rising edge of the internal processor clock following the rising edge of $\overline{\text{RESET}}$ initiates the reset sequence.
4. V_{DD} must fall to a level lower than V_{POR} to be recognized as a power-on reset.

Figure 5-2. Reset and POR Timing Diagram

5.4.2 Computer Operating Properly Reset (COPR)

The MCU contains a watchdog timer that automatically times out if not reset (cleared) within a specific time by a program reset sequence. If the COP watchdog timer is allowed to time out, an internal reset is generated to reset the MCU. Regardless of an internal or external reset, the MCU comes out of a COP reset according to the pin conditions that determine mode selection.

The COP reset function is enabled or disabled by the MOR[COPE] bit and is verified during production testing.

The COP watchdog reset will activate the internal pulldown device connected to the $\overline{\text{RESET}}$ pin.

5.4.2.1 Resetting the COP

Preventing a COP reset is done by writing a 0 to the COPR bit. This action will reset the counter and begin the timeout period again. The COPR bit is bit 0 of address $3FF0. A read of address $3FF0 will return user data programmed at that location.

5.4.2.2 COP during Wait Mode

The COP will continue to operate normally during wait mode. The system should be configured to pull the device out of wait mode periodically and reset the COP by writing to the COPR bit to prevent a COP reset.

5.4.2.3 COP during Stop Mode

When the STOP enable mask option is selected, stop mode disables the oscillator circuit and thereby turns the clock off for the entire device. The COP counter will be reset when stop mode is entered. If a reset is used to exit stop mode, the COP counter will be held in reset during the 4064 cycles of startup delay. If any operable interrupt is used to exit stop mode, the COP counter will not be reset during the 4064-cycle startup delay and will have that many cycles already counted when control is returned to the program.

5.4.2.4 COP Watchdog Timer Considerations

The COP watchdog timer is active in user mode if enabled by the MOR[COPEN] bit. If the COP watchdog timer is selected, any execution of the STOP instruction (either intentional or inadvertent due to the CPU being disturbed) will cause the oscillator to halt and prevent the COP watchdog timer from timing out. Therefore, it is recommended that the STOP instruction should be disabled if the COP watchdog timer is enabled.

If the COP watchdog timer is selected, the COP will reset the MCU when it times out. Therefore, it is recommended that the COP watchdog should be disabled for a system that must have intentional uses of the wait mode for periods longer than the COP timeout period.

The recommended interactions and considerations for the COP watchdog timer, STOP instruction, and WAIT instruction are summarized in **Table 5-1**.

Table 5-1. COP Watchdog Timer Recommendations

IF These Conditions Exist:		**THEN the COP Watchdog Timer Should:**
STOP Instruction	**WAIT Time**	
Converted to reset	WAIT time less than COP timeout	Enable or disable COP by the MOR
Converted to reset	WAIT time MORE than COP tmeout	Disable COP by the MOR
Acts as STOP	Any length WAIT time	Disable COP by the MOR

5.4.2.5 COP Register

The COP register is shared with the most significant bit (MSB) of an unimplemented user interrupt vector as shown in **Figure 5-3**. Reading this location will return whatever user data has been programmed at this location. Writing a 0 to the COPR bit in this location will clear the COP watchdog timer.

Address: $3FF0

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

▨ = Unimplemented

Figure 5-3. COP Watchdog Timer Location

5.4.3 Illegal Address Reset

An illegal address reset is generated when the CPU attempts to fetch an instruction from either unimplemented address space ($01C0 to $023F and $0340 to $0CFF) or I/O address space ($0000 to $003F).

The illegal address reset will activate the internal pulldown device connected to the $\overline{\text{RESET}}$ pin.

5.4.4 Disabled STOP Instruction Reset

When the mask option is selected to disable the STOP instruction, execution of a STOP instruction results in an internal reset. This activates the internal pulldown device connected to the $\overline{\text{RESET}}$ pin.

5.4.5 Low-Voltage Reset (LVR)

The internal LVR is generated when V_{DD} falls below the LVR threshold, $V_{LVRI,}$ and will be released following a POR delay starting when V_{DD} rises above V_{LVRR}. The LVR threshold is tested to be above the minimum operating voltage of the microcontroller and is intended to assure that the CPU will be held in reset when the V_{DD} supply voltage is below reasonable operating limits. A mask option is provided to disable the LVR when the device is expected to normally operate at low voltages. Note that the V_{DD} rise and fall slew rates (S_{VDDR} and S_{VDDF}) must be within the specification for proper LVR operation. If the specification is not met, the circuit will operate properly following a delay of V_{DD}/slew rate.

The LVR will generate the RST signal which will reset the CPU and other peripherals. The low-voltage reset will activate the internal pulldown device connected to the $\overline{\text{RESET}}$ pin.

If any other reset function is active at the end of the LVR reset signal, the RST signal will remain in the reset condition until the other reset condition(s) end.

5.4.6 LVR Operation in Stop and Wait Modes

If enabled, the LVR supply voltage sense option is active during stop and wait modes. Any reset source can bring the MCU out of stop or wait mode.

Section 6. Low-Power Modes

6.1 Contents

6.2 Introduction

The MC68HC705V12 is capable of running in one of several low-power operational modes. The WAIT and STOP instructions provide two modes that reduce the power required for the MCU by stopping various internal clocks and/or the on-chip oscillator. The STOP and WAIT instructions are not normally used if the computer operating properly (COP) watchdog timer is enabled. A programmable mask option is provided to convert the STOP instruction to an internal reset. The flow of the stop and wait modes is shown in **Figure 6-2**.

6.3 STOP Instruction

The STOP instruction can result in one of two operations depending on the state of the MOR[STOPE] bit:

- If the STOP option is enabled, the STOP instruction operates like the STOP in normal MC68HC05 Family members and places the device in the low-power stop mode.
- If the STOP option is disabled, the STOP instruction will cause a chip reset when executed.

6.4 Stop Mode

Execution of the STOP instruction with the MOR[STOPE] bit set places the MCU in its lowest power-consumption mode. In stop mode, the internal oscillator is turned off, halting all internal processing, including the COP watchdog timer.

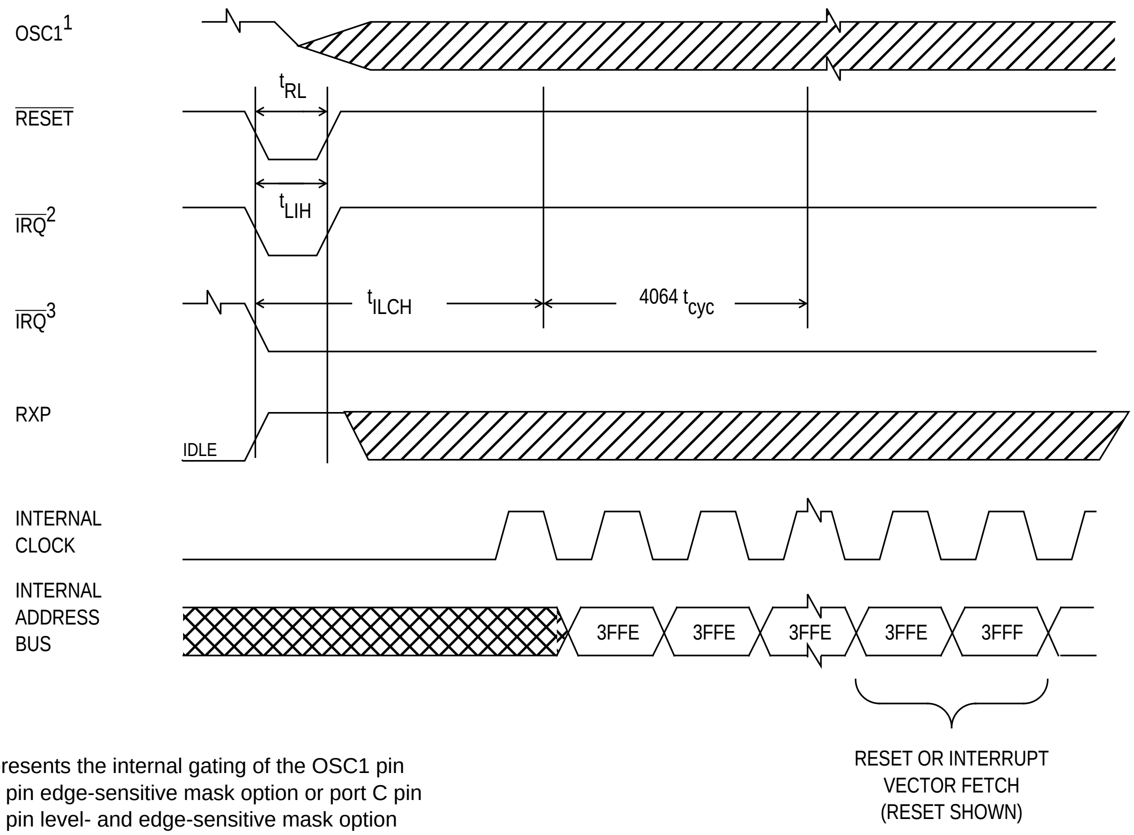
During stop mode, the TCR bits are altered to remove any pending timer interrupt request and to disable any further timer interrupts. The timer prescaler is cleared. The I bit in the condition code register (CCR) is cleared and the IRQE mask is set in the ICSR to enable external interrupts. All other registers and memory remain unaltered. All input/output lines remain unchanged.

The MCU can be brought out of stop mode only by:

- An $\overline{\text{IRQ}}$ pin external interrupt
- An externally generated reset
- A falling edge on any port C pin (if enabled)
- A rising edge on the BDLC RXP pin

When exiting the stop mode, the internal oscillator will resume after a 4064 internal processor clock cycle oscillator stabilization delay as shown in **Figure 6-1**.

NOTE: *Entering stop mode will cause the oscillator to stop and, therefore, disable the COP watchdog timer. If the COP watchdog timer is to be used, stop mode should be disabled by programming MOR[STOPE] to a 0.*



Notes:

1. Represents the internal gating of the OSC1 pin
2. $\overline{IRQ}$ pin edge-sensitive mask option or port C pin
3. $\overline{IRQ}$ pin level- and edge-sensitive mask option

Figure 6-1. Stop Recovery Timing Diagram

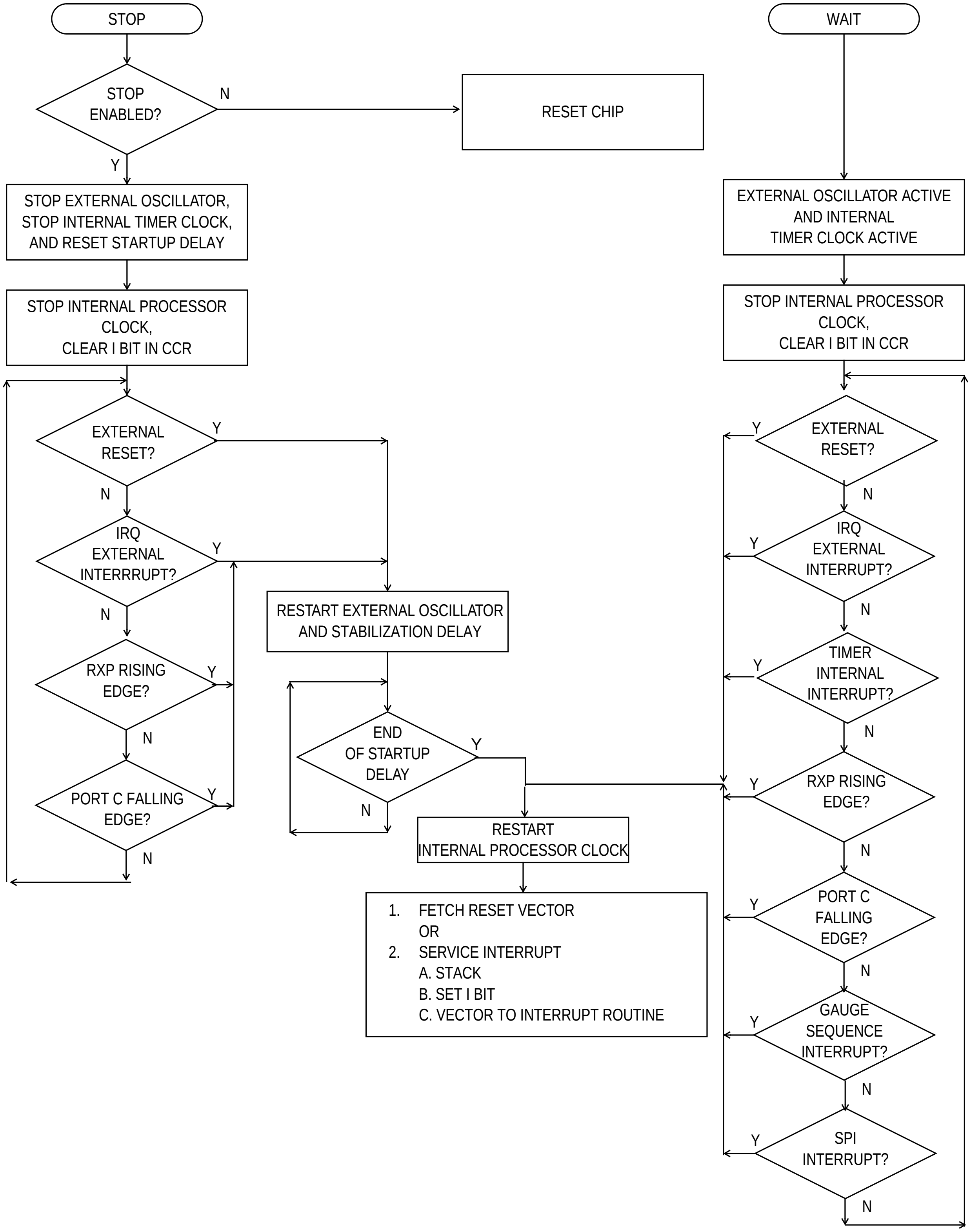

STOP
STOP ENABLED?
N
RESET CHIP
Y
STOP EXTERNAL OSCILLATOR, STOP INTERNAL TIMER CLOCK, AND RESET STARTUP DELAY
STOP INTERNAL PROCESSOR CLOCK, CLEAR I BIT IN CCR
EXTERNAL RESET?
Y
N
IRQ EXTERNAL INTERRRUPT?
Y
N
RXP RISING EDGE?
Y
N
PORT C FALLING EDGE?
Y
N
RESTART EXTERNAL OSCILLATOR AND STABILIZATION DELAY
END OF STARTUP DELAY
Y
N
RESTART INTERNAL PROCESSOR CLOCK
1. FETCH RESET VECTOR
OR
2. SERVICE INTERRUPT
A. STACK
B. SET I BIT
C. VECTOR TO INTERRUPT ROUTINE
WAIT
EXTERNAL OSCILLATOR ACTIVE AND INTERNAL TIMER CLOCK ACTIVE
STOP INTERNAL PROCESSOR CLOCK, CLEAR I BIT IN CCR
EXTERNAL RESET?
Y
N
IRQ EXTERNAL INTERRUPT?
Y
N
TIMER INTERNAL INTERRUPT?
Y
N
RXP RISING EDGE?
Y
N
PORT C FALLING EDGE?
Y
N
GAUGE SEQUENCE INTERRUPT?
Y
N
SPI INTERRUPT?
Y
N

Figure 6-2. Stop/Wait Flowcharts

6.5 WAIT Instruction

The WAIT instruction places the MCU in a low-power mode, which consumes more power than stop mode. In wait mode, the internal processor clock is halted, suspending all processor and internal bus activity. Internal timer clocks remain active, permitting interrupts to be generated from the timer or a reset to be generated from the COP watchdog timer. Execution of the WAIT instruction automatically clears the I bit in the condition code register. All other registers, memory, and input/output lines remain in their previous states.

If timer interrupts are enabled, a timer interrupt will cause the processor to exit wait mode and resume normal operation. The timer may be used to generate a periodic exit from wait mode.

The MCU can be brought out of wait mode by:

- A TIMER interrupt from either timer
- A serial peripheral interface (SPI) interrupt
- An $\overline{\text{IRQ}}$ pin external interrupt
- An externally generated reset
- A falling edge on any port C pin, if enabled
- A rising edge on the BDLC RXP pin
- A gauge sequence interrupt

6.6 Data-Retention Mode

Contents of the random-access memory (RAM) and central processor unit (CPU) registers are retained at supply voltages as low as 2.0 Vdc. This is called the data-retention mode where the data is held, but the device is not guaranteed to operate. The $\overline{\text{RESET}}$ pin must be held low during data-retention mode.

NOTE: *More power is consumed in data-retention mode than in stop mode because internal clocks remain running.*

Section 7. Parallel Input/Output (I/O)

7.1 Contents

7.2 Introduction

In single-chip mode, 23 bidirectional input/output (I/O) lines are arranged as two 8-bit I/O ports (ports B and C), and one 7-bit I/O port (port A). There is one 5-bit input port (port D). The individual bits in the I/O ports are programmable as either inputs or outputs under software control by the data direction registers (DDRs). The port C pins also have the additional property of acting as IRQ interrupt input sources.

7.3 Port A

Port A is a 7-bit bidirectional port which functions as shown in **Figure 7-1**. Each pin is controlled by the corresponding bit in a data direction register and a data register. The port A data register (PORTA) is located at address $0000. The port A data direction register (DDRA) is located at address $0004. Reset clears DDRA. The port A data register is unaffected by reset.

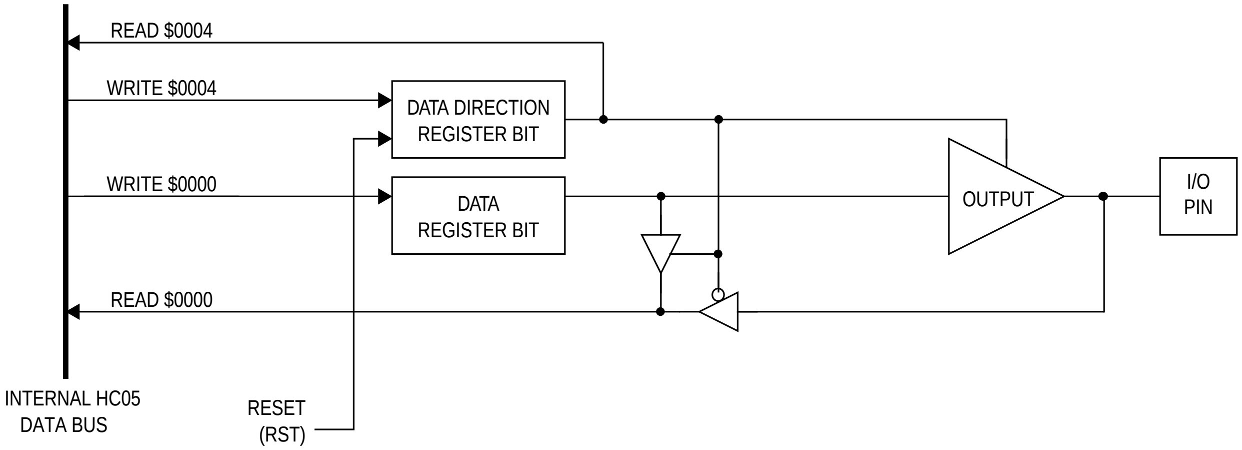


Figure 7-1. Port A I/O Circuitry

7.3.1 Port A Data Register

Each port A I/O pin has a corresponding bit in the port A data register. When a port A pin is programmed as an output, the state of the corresponding data register bit determines the state of the output pin. When a port A pin is programmed as an input, any read of the port A data register will return the logic state of the corresponding I/O pin. The port A data register is unaffected by reset.

7.3.2 Port A Data Direction Register

Each port A I/O pin may be programmed as an input by clearing the corresponding bit in the DDRA or programmed as an output by setting the corresponding bit in the DDRA. The DDRA can be accessed at address $0004 and is cleared by reset.

7.4 Port B

Port B is an 8-bit bidirectional port. Each port B pin is controlled by the corresponding bits in a data direction register and a data register as shown in **Figure 7-2**. PB5 and PB4 are shared with the PWMs as shown in **Section 11. Pulse Width Modulators (PWMs)** and PB7 and PB6 are shared with 16-bit timer functions. See **Section 9. 16-Bit Timer** for timer description. PB0–PB3 are shared with the SPI as shown in **Section 10. Serial Peripheral Interface (SPI)**. The port B data register (PORTB) is located at address $0001. The port B data direction register (DDRB) is located at address $0005. Reset clears the DDRB register. The port B data register is unaffected by reset.

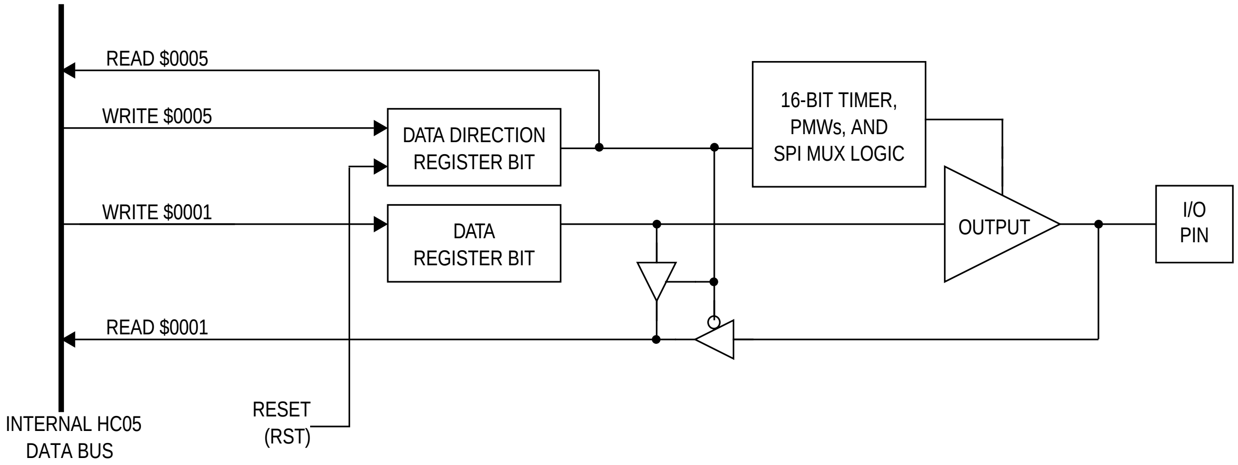


Figure 7-2. Port B I/O Circuitry

7.4.1 Port B Data Register

Each port B I/O pin has a corresponding bit in the port B data register. When a port B pin is programmed as an output, the state of the corresponding data register bit determines the state of the output pin. When a port B pin is programmed as an input, any read of the port B data register will return the logic state of the corresponding I/O pin. The port B data register is unaffected by reset.

7.4.2 Port B Data Direction Register

Each port B I/O pin may be programmed as an input by clearing the corresponding bit in the DDRB or programmed as an output by setting the corresponding bit in the DDRB. The DDRB can be accessed at address $0005. The DDRB is cleared by reset.

7.5 Port C

Port C is an 8-bit bidirectional port shared with the IRQ interrupt subsystem as shown in **Figure 7-3**. Each pin is controlled by the corresponding bits in a data direction register and a data register. The port C data register (PORTC) is located at address $0002. The port C data direction register (DDRC) is located at address $0006. Reset clears DDRC. The port C data register is unaffected by reset.

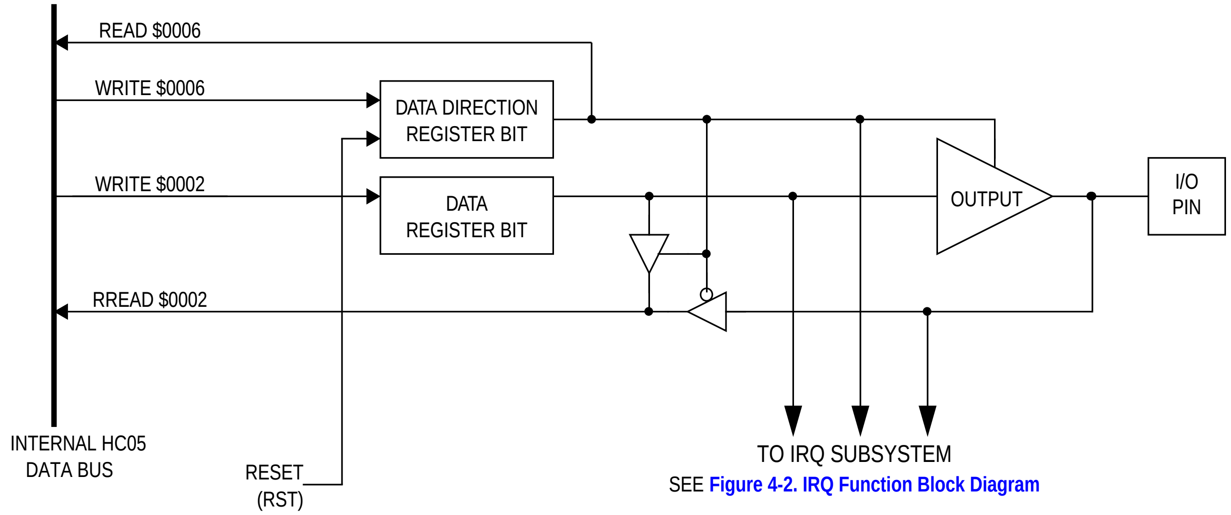


Figure 7-3. Port C I/O Circuitry

7.5.1 Port C Data Register

Each port C I/O pin has a corresponding bit in the port C data register (PORTC). When a port C pin is programmed as an output, the state of the corresponding data register bit determines the state of the output pin. When a port C pin is programmed as an input, any read of the port C data register will return the logic state of the corresponding I/O pin. The port C data register is unaffected by reset.

7.5.2 Port C Data Direction Register

Each port C I/O pin may be programmed as an input by clearing the corresponding bit in the port C data direction register (DDRC) or programmed as an output by setting the corresponding bit in the DDRC. The DDRC can be accessed at address $0006 and is cleared by reset.

7.5.3 Port C I/O Pin Interrupts

The inputs of all eight bits of port C are ANDed into the IRQ input of the CPU. See **Figure 4-2. IRQ Function Block Diagram**. This port has its own interrupt request latch to enable the user to differentiate between the IRQ sources. The port IRQ inputs are falling edge sensitive only. Any port C pin can be disabled as an interrupt input by setting the corresponding DDR bit or data register bit. To enable port pin interrupts, the corresponding DDR and data register bits must both be cleared. Any port C pin that is configured as an output will not cause a port interrupt when the pin transitions from a 1 to a 0.

NOTE: *The BIH and BIL instructions will apply only to the level on the IRQ pin itself and not to the internal IRQ input to the CPU. Therefore, BIH and BIL cannot be used to obtain the result of the logical combination of the eight pins of port C.*

CAUTION: *Exercise caution when writing to the port C data register and data direction register due to their interaction with the IRQ subsystem as depicted in* ***Figure 4-2. IRQ Function Block Diagram****. Special care should be exercised in using read/modify/write instructions on these registers.*

7.6 Port D

Port D is a 5-bit input-only port which shares all of its pins with the A/D converter (AD0–AD4) as shown in **Figure 7-4**. The port D data register (PORTD) is located at address $0003. When the A/D converter is active, one of these five input ports may be selected by the A/D multiplexer for conversion. A logical read of a selected input port will always return 0.

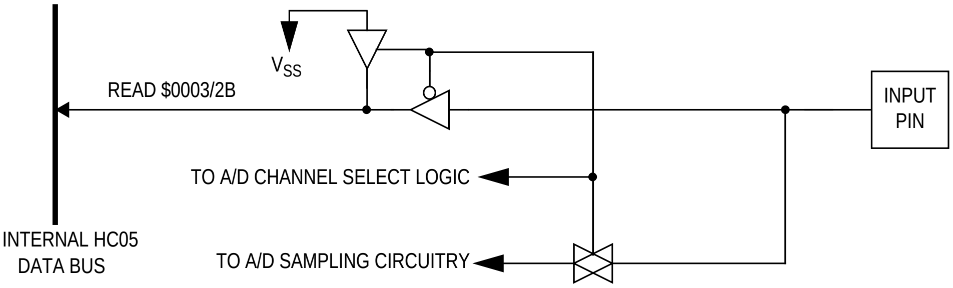


Figure 7-4. Port D Circuitry

Section 8. Core Timer

8.1 Contents

8.2 Introduction

The core timer for this device is a 12-stage multi-functional ripple counter. Features include:

- Timer overflow
- Power-on reset (POR)
- Real-time interrupt (RTI)
- Computer operating properly (COP) watchdog timer

As seen in in **Figure 8-1**, the internal peripheral clock is divided by four then drives an 8-bit ripple counter. The value of this 8-bit ripple counter can be read by the CPU at any time by accessing the core timer counter register (CTCR) at address $09. A timer overflow function is implemented on the last stage of this counter, giving a possible interrupt rate of the internal peripheral clock(E)/1024. This point is then followed by two more stages, with the resulting clock (E/2048) driving the RTI circuit. The RTI circuit consists of three divider stages with a 1-of-4 selector. The output of the RTI circuit is further divided by eight to drive the mask optional COP watchdog timer circuit. The RTI rate selector bits and the RTI and CTOF enable bits and flags are located in the timer control and status register at location $08.

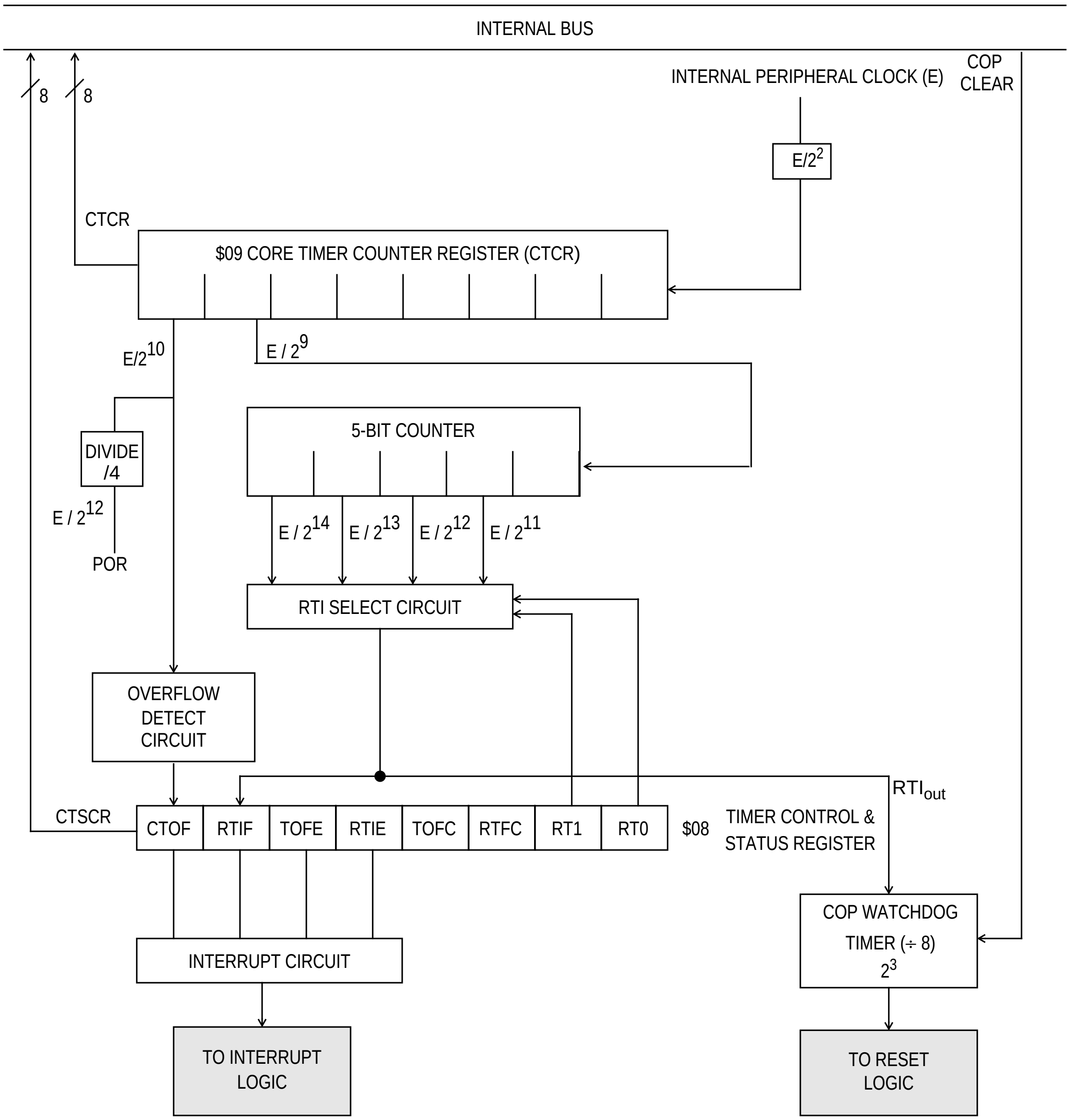


Figure 8-1. Core Timer Block Diagram

8.3 Core Timer Status and Control Register

The core timer status and control register (CTSCR) contains:

- Timer interrupt flag
- Timer interrupt enable bits
- Real-time interrupt rate select bits

Figure 8-2 shows the value of each bit in the CTSCR when coming out of reset.

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CTOF	RTIF	TOFE	RTIE	0	0	RT1	RT0
Write:	▨	▨			TOFC	RTFC		
Reset:	0	0	0	0	0	0	1	1

▨ = Unimplemented

Figure 8-2. Core Timer Status and Control Register (CTSCR)

CTOF — Core Timer Overflow Bit

CTOF is a read-only status bit set when the 8-bit ripple counter rolls over from $FF to $00. Clearing the CTOF is done by writing a 1 to TOFC. Writing to this bit has no effect. Reset clears CTOF.

RTIF — Real Time Interrupt Flag

The real-time interrupt circuit consists of a 3-stage divider and a 1-of-4 selector. The clock frequency that drives the RTI circuit is E/2**11 (or E/2048) with three additional divider stages giving a maximum interrupt period of 7.8 milliseconds at a bus rate of 2.1 MHz. RTIF is a clearable, read-only status bit and is set when the output of the chosen (1-of-4 selection) stage goes active. Clearing the RTIF is done by writing a 1 to RTFC. Writing has no effect on this bit. Reset clears RTIF.

TOFE — Timer Overflow Enable Bit

When this bit is set, a CPU interrupt request is generated when the CTOF bit is set. Reset clears this bit.

RTIE — Real-Time Interrupt Enable Bit

When this bit is set, a CPU interrupt request is generated when the RTIF bit is set. Reset clears this bit.

TOFC — Timer Overflow Flag Clear Bit

When a 1 is written to this bit, CTOF is cleared. Writing a 0 has no effect on the CTOF bit. This bit always reads as 0.

RTFC — Real-Time Interrupt Flag Clear Bit

When a 1 is written to this bit, RTIF is cleared. Writing a 0 has no effect on the RTIF bit. This bit always reads as 0.

RT1–RT0 — Real-Time Interrupt Rate Select Bit

These two bits select one of four taps from the real-time interrupt (RTI) circuit. See **Table 8-1** which shows the available interrupt rates with a 2.1- and 1.05-MHz bus clock. Reset sets bits RT1 and RT0, which selects the lowest periodic rate, and gives the maximum time in which to alter these bits if necessary. Take care when altering RT0 and RT1 if the timeout period is imminent or uncertain. If the selected tap is modified during a cycle in which the counter is switching, an RTIF could be missed or an additional one could be generated. To avoid problems, the COP should be cleared before changing RTI taps.

Table 8-1. RTI and COP Rates at 2.1 MHz

RTI Rate			**RT1–RT0**	**Minimum COP Rates**		
2.1 MHz	**1.05 MHz**				**2.1 MHz**	**1.05 MHz**
0.97 ms	1.95 ms	$2^{11}/E$	00	$(2^{14}-2^{11})/E$	6.83 ms	13.65 ms
1.95 ms	3.90 ms	$2^{12}/E$	01	$(2^{15}-2^{12})/E$	13.65 ms	27.31 ms
3.90 ms	7.80 ms	$2^{13}/E$	10	$(2^{16}-2^{13})/E$	27.31 ms	54.61 ms
7.80 ms	15.60 ms	$2^{14}/E$	11	$(2^{17}-2^{14})/E$	54.61 ms	109.23 ms

8.4 Computer Operating Properly (COP) Reset

The computer operating properly (COP) watchdog timer function is implemented on this device by using the output of the RTI circuit and further dividing it by eight. The minimum COP reset rates are listed in **Figure 8-1**. If the COP circuit times out, an internal reset is generated and the normal reset vector is fetched. Preventing a COP timeout, or clearing the COP, is accomplished by writing a 0 to bit 0 of address $3FF0. When the COP is cleared, only the final divide-by-eight stage (output of the RTI) is cleared. The COP time out period will vary depending on when the COP is fed with respect to the RTI output clock.

If the COP watchdog timer is allowed to time out, an internal reset is generated to reset the MCU. In addition the $\overline{\text{RESET}}$ pin will be pulled low for a minimum of three E clock cycles for emulation purposes. During a chip reset (regardless of the source), the entire core timer counter chain is cleared.

The COP will remain enabled after execution of the WAIT instruction and all associated operations apply. If the STOP instruction is disabled, execution of STOP instruction will cause an internal reset.

This COP’s objective is to make it impossible for this part to become “stuck” or “locked-up” and to be sure the COP is able to “rescue” the part from any situation where it might entrap itself in an abnormal or unintended behavior. This function is a mask option.

8.5 Core Timer Counter Register

The core timer counter register (CTCR) is a read-only register which contains the current value of the 8-bit ripple counter at the beginning of the timer chain. This counter is clocked by the CPU clock (E/4) and can be used for various functions including a software input capture. Extended time periods can be attained using the TOF function to increment a temporary RAM storage location, thereby simulating a 16-bit (or more) counter.

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 8-3. Core Timer Counter Register (CTCR)

The power-on cycle clears the entire counter chain and begins clocking the counter. After 4064 cycles, the power-on reset circuit is released which again clears the counter chain and allows the device to come out of reset. At this point, if $\overline{\text{RESET}}$ is not asserted, the timer will start counting up from 0 and normal device operation will begin. When $\overline{\text{RESET}}$ is asserted any time during operation (other than POR), the counter chain will be cleared.

8.6 Core Timer during Wait Mode

The CPU clock halts during wait mode, but the timer remains active. If interrupts are enabled, a timer interrupt will cause the processor to exit wait mode. The COP watchdog timer, derived from the core timer, remains active in wait mode, if enabled via the MOR.

Section 9. 16-Bit Timer

9.1 Contents

9.2 Introduction

The timer consists of a 16-bit, free-running counter driven by a fixed divide-by-four prescaler. This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from several microseconds to many seconds. See **Figure 9-1**.

Because the timer has a 16-bit architecture, each specific functional segment (capability) is represented by two registers. These registers contain the high and low bytes of that functional segment. Access of the high byte inhibits that specific timer function until the low byte is also accessed.

NOTE: *The I bit in the condition code register (CCR) should be set while manipulating both the high and low byte registers of a specific timer function to ensure that an interrupt does not occur.*

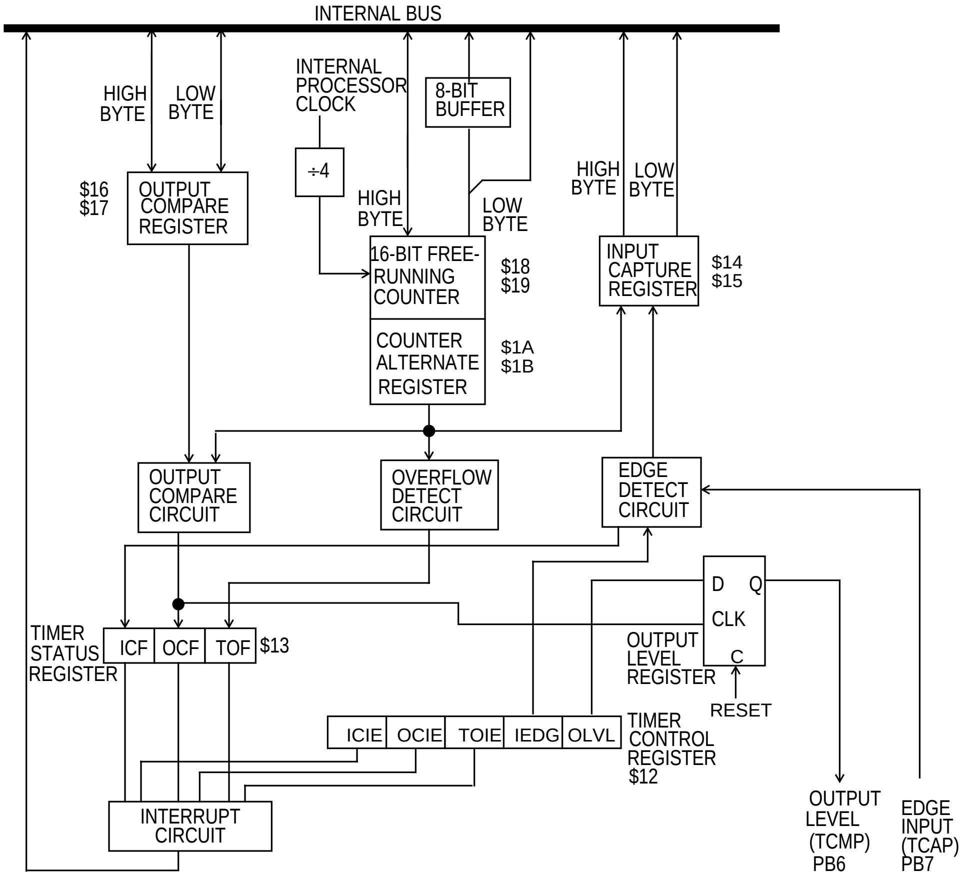


Figure 9-1. 16-Bit Timer Block Diagram

9.3 Timer Counter Registers $18–$19 and $1A–$1B

The key element in the programmable timer is a 16-bit, free-running counter or counter register preceded by a prescaler that divides the internal processor clock by four. The prescaler gives the timer a resolution of 2.0 microseconds if the internal bus clock is 2.0 MHz. The counter is incremented during the low portion of the internal bus clock. Software can read the counter at any time without affecting its value.

The double-byte, free-running counter can be read from either of two locations, $18–$19 (counter register) or $1A–$1B (counter alternate register). A read from only the least significant byte (LSB) of the free-running counter ($19, $1B) receives the count value at the time of the read. If a read of the free-running counter or counter alternate register first addresses the most significant byte (MSB) ($18, $1A), the LSB ($19, $1B) is transferred to a buffer. This buffer value remains fixed after the first MSB read, even if the user reads the MSB several times. This buffer is accessed when reading the free-running counter or counter alternate register LSB ($19 or $1B) and, thus, completes a read sequence of the total counter value. In reading either the free-running counter or counter alternate register, if the MSB is read, the LSB also must be read to complete the sequence.

The counter alternate register differs from the counter register in one respect: A read of the counter register MSB can clear the timer overflow flag (TOF). Therefore, the counter alternate register can be read at any time without the possibility of missing timer overflow interrupts due to clearing of the TOF.

The free-running counter is configured to $FFFC during reset and is a read-only register only when the timer is enabled. During a power-on reset, the counter also is preset to $FFFC and begins running only after the TON bit in the timer control register is set. Because the free-running counter is 16 bits preceded by a fixed divided-by-four prescaler, the value in the free-running counter repeats every 262,144 internal bus clock cycles. When the counter rolls over from $FFFF to $0000, the TOF bit is set. When counter roll-over occurs, an interrupt also can be enabled by setting its interrupt enable bit (TOIE).

NOTE: *To ensure that an interrupt does not occur, the I bit in the CCR should be set while manipulating both the high and low byte registers of a specific timer function.*

9.4 Output Compare Register $16–$17

The 16-bit output compare register is made up of two 8-bit registers at locations $16 (MSB) and $17 (LSB). The output compare register is used for several purposes, such as indicating when a period of time has elapsed. All bits are readable and writable and are not altered by the timer hardware or reset. If the compare function is not needed, the two bytes of the output compare register can be used as storage locations.

The output compare register contents are continually compared with the contents of the free-running counter. If a match is found, the corresponding output compare flag (OCF) bit is set and the corresponding output level (OLVL) bit is clocked to an output level register. The output compare register values and the output level bit should be changed after each successful comparison to establish a new elapsed timeout. An interrupt can also accompany a successful output compare provided the corresponding interrupt enable bit (OCIE) is set. After a processor write cycle to the output compare register containing the MSB ($16), the output compare function is inhibited until the LSB ($17) is also written. The user must write both bytes (locations) if the MSB is written first. A write made only to the LSB ($17) will not inhibit the compare function. The free-running counter is updated every four internal bus clock cycles. The minimum time required to update the output compare register is a function of the program rather than the internal hardware.

The processor can write to either byte of the output compare register without affecting the other byte. The output level (OLVL) bit is clocked to the output level register regardless of whether the output compare flag (OCF) is set or clear.

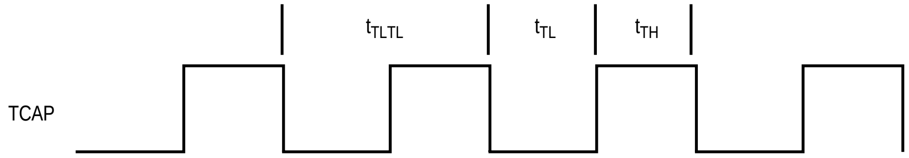
9.5 Input Capture Register $14–$15

Two 8-bit registers, which make up the 16-bit input capture register, are read-only and are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The level transition which triggers the counter transfer is defined by the corresponding input edge bit (IEDG). Reset does not affect the contents of the input capture register.

The result obtained by an input capture will be one more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization. Resolution is one count of the free-running counter, which is four internal bus clock cycles.

The free-running counter contents are transferred to the input capture register on each proper signal transition regardless of whether the input capture flag (ICF) is set or clear. The input capture register always contains the free-running counter value that corresponds to the most recent input capture.

After a read of the input capture register MSB ($14), the counter transfer is inhibited until the LSB ($15) is also read. This characteristic causes the time used in the input capture software routine and its interaction with the main program to determine the minimum pulse period. A read of the input capture register LSB ($15) does not inhibit the free-running counter transfer since they occur on opposite edges of the internal bus clock.



See control timing specifications for TCAP timing requirements.

Figure 9-2. TCAP Timing

NOTE: *The input capture pin (TCAP) and the output compare pin (TCMP) are shared with PB7 and PB6 respectively. The timer’s TCAP input always is connected to PB7. PB6 is the timer’s TCMP pin if the OCE bit in the miscellaneous control register is set.*

9.6 16-Bit Timer Control Register

The 16-bit timer control register (TMRCR) is a read/write register containing six control bits. Three bits control interrupts associated with the timer status register flags ICF, OCF, and TOF.

Address: $0012

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ICIE	OCIE	TOIE	0	0	TON	IEDG	OLVL
Write:				▨	▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 9-3. 16- Bit Timer Control Register (TMRCR)

ICIE — Input Capture Interrupt Enable Bit
1 = Interrupt enabled
0 = Interrupt disabled

OCIE — Output Compare Interrupt Enable Bit
1 = Interrupt enabled
0 = Interrupt disabled

TOIE — Timer Overflow Interrupt Enable Bit
1 = Interrupt enabled
0 = Interrupt disabled

TON — Timer On Bit

When disabled, the timer is initialized to the reset condition.
1 = Timer enabled
0 = Timer disabled

IEDG — Input Edge Bit

Value of input edge determines which level transition on TCAP pin will trigger free-running counter transfer to the input capture register. Reset clears this bit.
1 = Positive edge
0 = Negative edge

OLVL — Output Level Bit

Value of output level is clocked into output level register by the next successful output compare and will appear on the TCMP pin.
1 = High output
0 = Low output

9.7 16-Bit Timer Status Register

The 16-bit timer status register (TMRSR) is a read-only register containing three status flag bits.

Address: $0013

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ICF	OCF	TOF	0	0	0	0	0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 9-4. Timer Status Register (TMRSR)

ICF – Input Capture Flag
1 = Flag set when selected polarity edge is sensed by input capture edge detector
0 = Flag cleared when TMRSR and input capture low register ($15) are accessed

OCF – Output Compare Flag
1 = Flag set when output compare register contents match the free-running counter contents
0 = Flag cleared when TMRSR and output compare low register ($17) are accessed

TOF – Timer Overflow Flag
1 = Flag set when free-running counter transition from $FFFF to $0000 occurs
0 = Flag cleared when TMRSR and counter low register ($19) are accessed

Accessing the timer status register satisfies the first condition required to clear status bits. The remaining step is to access the register corresponding to the status bit.

A problem can occur when using the timer overflow function and reading the free-running counter at random times to measure an elapsed time. Without incorporating the proper precautions into software, the timer overflow flag could unintentionally be cleared if both of these occur:

1. The timer status register is read or written when TOF is set
2. The MSB of the free-running counter is read but not for the purpose of servicing the flag.

The counter alternate register at address $1A and $1B contains the same value as the free-running counter (at address $18 and $19); therefore, this alternate register can be read at any time without affecting the timer overflow flag in the timer status register.

9.8 16-Bit Timer during Wait Mode

The CPU clock halts during wait mode, but the timer remains active if turned on prior to entering wait mode. If interrupts are enabled, a timer interrupt will cause the processor to exit wait mode.

9.9 16-Bit Timer during Stop Mode

In stop mode, the timer stops counting and holds the last count value if stop mode is exited by an interrupt. If reset is used, the counter is forced to $FFFC. During STOP, if the timer is on and at least one valid input capture edge occurs at the TCAP pin, the input capture detect circuit is armed. This does not set any timer flags or wake up the MCU, but when the MCU does wake up, there is an active input capture flag and data from the first valid edge that occurred during stop mode. If RESET is used to exit stop mode, then no input capture flag or data remains, even if a valid input capture edge occurred.

Section 10. Serial Peripheral Interface (SPI)

10.1 Contents

10.2 Introduction

The serial peripheral interface (SPI) allows several MC68HC05 microcontrollers (MCUs) or an MC68HC05 MCU plus peripheral devices to be interconnected within a single printed circuit board. In an SPI, separate wires are required for data and clock. In the SPI format, the clock is not included in the data stream and must be furnished as a separate signal.

10.3 Features

Features of the MC68HC705V12 include:

- Full duplex, 3-wire synchronous transfers
- Master or slave operation
- Internal MCU clock divided by two (maximum) master bit frequency
- Internal MCU clock (maximum) slave bit frequency
- Four programmable master bit rates
- Programmable clock polarity and phase
- End of transmission interrupt flag
- Write collision flag protection
- Master-master mode fault protection capability

10.4 SPI Signal Description

This subsection describes these signal functions for both master and slave modes:

- Master out/slave in (MOSI)
- Master in/slave out (MISO)
- Serial clock (SCK)
- Slave select ($\overline{\text{SS}}$)

To function properly, the SPI forces the direction on some of the pins to output.

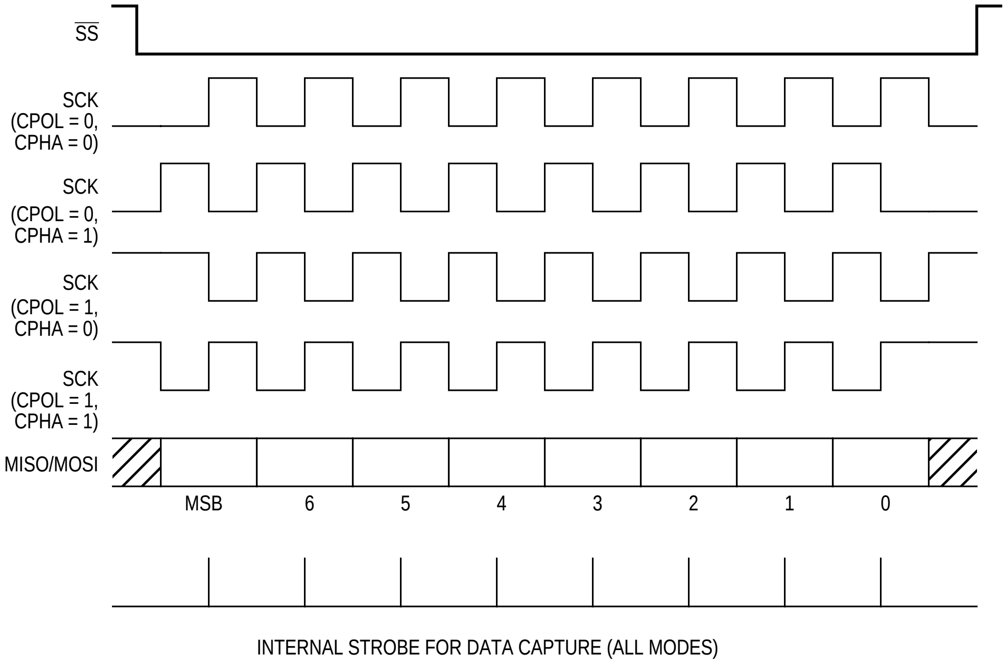


Figure 10-1. Data Clock Timing Diagram

10.4.1 Slave Select ($\overline{SS}$/PB0)

The slave select ($\overline{SS}$) pin is used to select the MCU as a slave device. It has to be low prior to data transactions and must stay low for the duration of the transaction. The $\overline{SS}$ pin on the master must be set high. If it goes low, a mode fault error flag (MODF) is set in the SPSR.

When CPHA = 0, the shift clock is the OR of $\overline{SS}$ with SCK. In this clock phase mode, $\overline{SS}$ must go high between successive characters in an SPI message. When CPHA = 1, $\overline{SS}$ may be left low for several SPI characters. In cases where there is only one SPI slave MCU, its $\overline{SS}$ pin could be set low as long as CPHA = 1 clock modes are used.

NOTE: *If the SPI is in master mode, this pin can be used as a general-purpose output pin. If configured as an input pin while in master mode, it must be set high.*

10.4.2 Serial Clock (SCK/PB1)

The master clock is used to synchronize data movement both in and out of the device through its MOSI and MISO lines. The master and slave devices are capable of exchanging a byte of information during a sequence of eight clock cycles. Since SCK is generated by the master device, this line becomes an input on a slave device.

As shown in **Figure 10-1**, four possible timing relationships may be chosen by using control bits CPOL and CPHA in the serial peripheral control register (SPCR). Both master and slave devices must operate with the same timing. The master device always places data on the MOSI line a half cycle before the clock edge (SCK) for the slave device to latch the data.

Two bits (SPR0 and SPR1) in the SPCR of the master device select the clock rate. In a slave device, SPR0 and SPR1 have no effect on the operation of the SPI.

10.4.3 Master In/Slave Out (MISO/PB2)

The MISO line is configured as an input in a master device and as an output in a slave device. It is one of the two lines that transfer serial data in one direction, with the most significant bit sent first. The MISO line of a slave device is placed in the high-impedance state if the slave is not selected.

10.4.4 Master Out/Slave In (MOSI/PB3)

The MOSI line is configured as an output in a master device and as an input in a slave device. It is one of the two lines that transfer serial data in one direction with the most significant bit sent first.

10.5 SPI Functional Description

Figure 10-2 shows a block diagram of the serial peripheral interface circuitry. When a master device transmits data to a slave via the MOSI line, the slave device responds by sending data to the master device via the master's MISO line. This implies full duplex transmission with both data out and data in synchronized with the same clock signal. Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receive-full status bits. A single status bit (SPIF) is used to signify that the I/O operation has been completed.

The SPI is double buffered on read, but not on write. If a write is performed during data transfer, the transfer occurs uninterrupted, and the write will be unsuccessful. This condition will cause the write collision (WCOL) status bit in the SPSR to be set. After a data byte is shifted, the SPIF flag of the SPSR is set.

In the master mode, the SCK pin is an output. It idles high or low, depending on the CPOL bit in the SPCR, until data is written to the shift register, at which point eight clocks are generated to shift the eight bits of data and then SCK goes idle again.

In a slave mode, the slave select start logic receives a logic low from the $\overline{\text{SS}}$ pin and a clock at the SCK pin. Thus, the slave is synchronized with the master. Data from the master is received serially at the MOSI line and loads the 8-bit shift register. After the 8-bit shift register is loaded, its data is parallel transferred to the read buffer. During a write cycle, data is written into the shift register, then the slave waits for a clock train from the master to shift the data out on the slave's MISO line.

Figure 10-3 illustrates the MOSI, MISO, SCK, and $\overline{\text{SS}}$ master-slave interconnections.

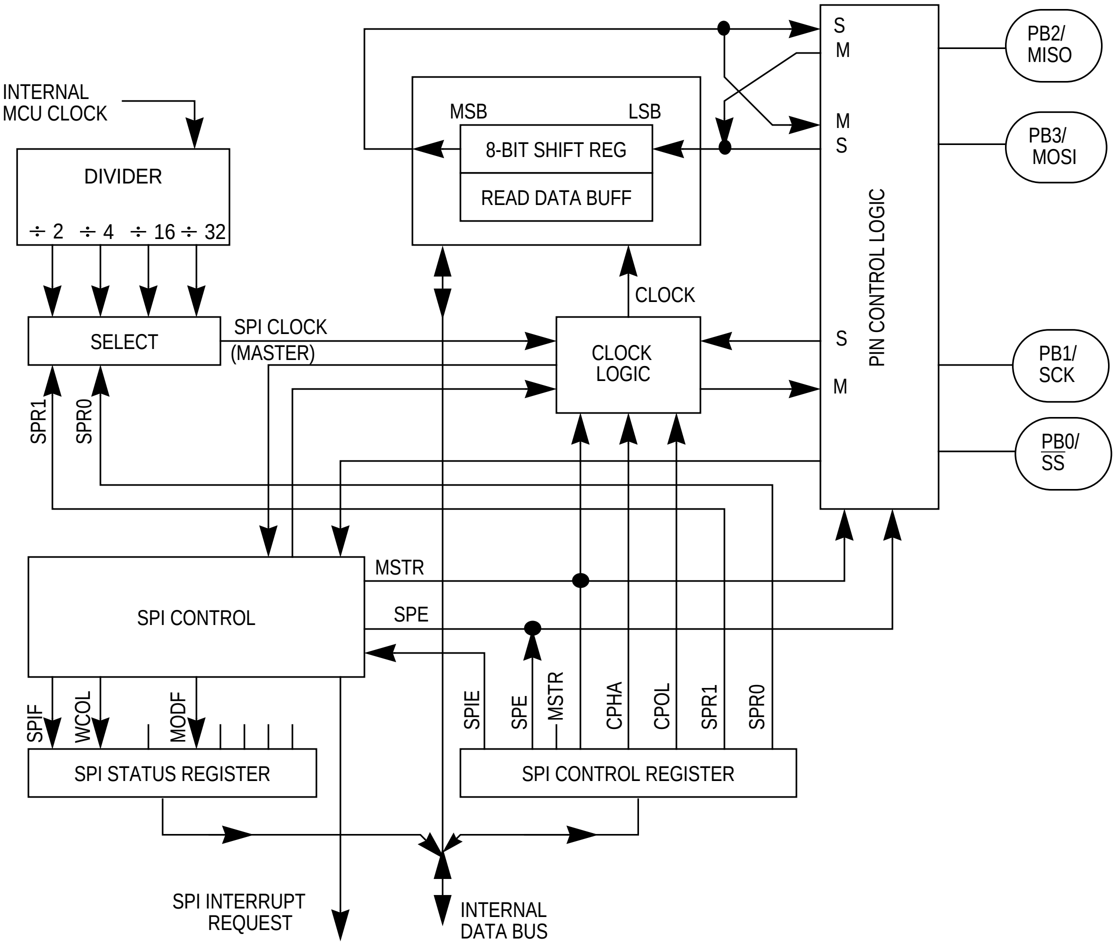


Figure 10-2. Serial Peripheral Interface Block Diagram

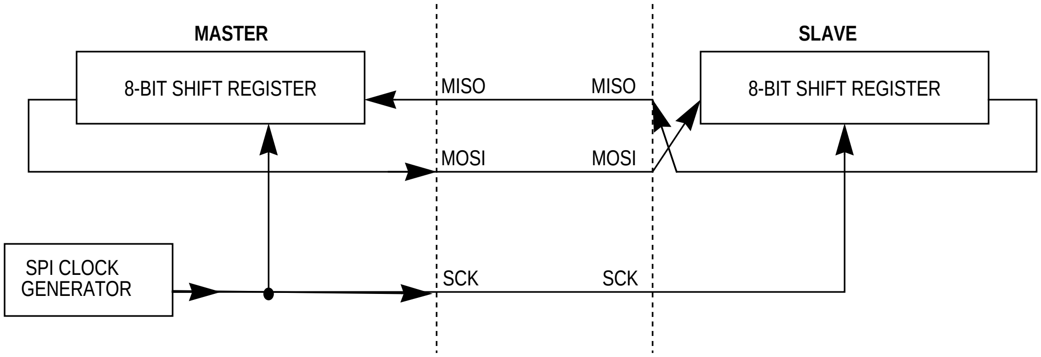


Figure 10-3. Serial Peripheral Interface Master-Slave Interconnection

10.6 SPI Registers

The three registers in the SPI, described here, provide control, status, and data storage functions. These registers are:

- Serial peripheral control register (SPCR)
- Serial peripheral status register (SPSR)
- Serial peripheral data I/O register (SPDR)

10.6.1 Serial Peripheral Control Register

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPIE	SPE	0	MSTR	CPOL	CPHA	SPR1	SPR0
Write:			▨					
Reset:	0	0	0	0	0	1	U	U

▨ = Unimplemented U = Unaffected

Figure 10-4. SPI Control Register (SPCR)

SPIE — Serial Peripheral Interrupt Enable Bit
1 = SPI interrupt if SPIF = 1
0 = SPIF interrupts disabled

SPE — Serial Peripheral System Enable Bit
1 = SPI system on; port B becomes SPI pins
0 = SPI system off

MSTR — Master Mode Select Bit
1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

When the clock polarity bit is cleared and data is not being transferred, a steady state low value is produced at the SCK pin of the master device. Conversely, if this bit is set, the SCK pin will idle high. This bit is also used in conjunction with the clock phase control bit to produce the desired clock-data relationship between master and slave. See **Figure 10-1**.

CPHA — Clock Phase Bit

The clock phase bit, in conjunction with the CPOL bit, controls the clock-data relationship between master and slave. The CPOL bit can be thought of as simply inserting an inverter in series with the SCK line. The CPHA bit selects one of two fundamentally different clocking protocols. When CPHA = 0, the shift clock is the OR of SCK with $\overline{\text{SS}}$. As soon as $\overline{\text{SS}}$ goes low, the transaction begins and the first edge on SCK invokes the first data sample. When CPHA = 1, $\overline{\text{SS}}$ may be thought of as a simple output enable control. See **Figure 10-1**.

SPR1 and SPR0 — SPI Clock Rate Select Bits

These two bits select one of four baud rates (see **Table 10-1**) to be used as SCK if the device is a master; however, they have no effect in slave mode.

Table 10-1. Serial Peripheral Rate Selection

SPR1	SPR0	Internal MCU Clock Divided by
0	0	2
0	1	4
1	0	16
1	1	32

10.6.2 Serial Peripheral Status Register

Address: $000B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPIF	WCOL	0	MODF	0	0	0	0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 10-5. SPI Status Register (SPSR)

SPIF — SPI Transfer Complete Flag

The serial peripheral data transfer flag bit is set upon completion of data transfer between the processor and external device. If SPIF goes high, and if SPIE is set, a serial peripheral interrupt is generated. Clearing the SPIF bit is accomplished by reading the SPSR (with SPIF set) followed by an access of the SPDR. Unless SPSR is read (with SPIF set) first, attempts to write to SPDR are inhibited.

WCOL — Write Collision Bit

The write collision bit is set when an attempt is made to write to the serial peripheral data register while data transfer is taking place. If CPHA is 0, a transfer is said to begin when $\overline{\text{SS}}$ goes low and the transfer ends when $\overline{\text{SS}}$ goes high after eight clock cycles on SCK. When CPHA is 1, a transfer is said to begin the first time SCK becomes active while $\overline{\text{SS}}$ is low and the transfer ends when the SPIF flag gets set. Clearing the WCOL bit is accomplished by reading the SPSR (with WCOL set) followed by an access to SPDR.

MODF — Mode Fault Bit

The mode fault flag indicates that there may have been a multi-master conflict for system control and allows a proper exit from system operation to a reset or default system state. The MODF bit is normally clear and is set only when the master device has its $\overline{\text{SS}}$ pin set low.

Setting the MODF bit affects the internal serial peripheral interface system in these ways:

1. An SPI interrupt is generated if SPIE = 1.
2. The SPE bit is cleared, disabling the SPI.
3. The MSTR bit is cleared, thus forcing the device into the slave mode.

Clearing the MODF bit is accomplished by reading the SPSR (with MODF set), followed by a write to the SPCR. Control bits SPE and MSTR may be restored by user software to their original state after the MODF bit has been cleared. It is also necessary to restore the port B DDR bits after a mode fault.

10.6.3 Serial Peripheral Data Register

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
Reset:	Unaffected by reset							

Figure 10-6. SPI Data Register (SPDR)

The serial peripheral data I/O register is used to transmit and receive data on the serial bus. Only a write to this register will initiate transmission/reception of another byte, and this will only occur in the master device. At the completion of transmitting a byte of data, the SPIF status bit is set in both the master and slave devices.

When the user reads the serial peripheral data I/O register, a buffer is actually being read. The first SPIF must be cleared by the time a second transfer of the data from the shift register to the read buffer is initiated or an overrun condition will exist. In cases of overrun, the byte which causes the overrun is lost.

A write to the serial peripheral data I/O register is not buffered and places data directly into the shift register for transmission.

10.7 SPI in Stop Mode

When the MCU enters stop mode, the baud rate generator driving the SPI shuts down. This essentially stops all master mode SPI operation; thus, the master SPI is unable to transmit or receive any data. If the STOP instruction is executed during an SPI transfer, that transfer is halted until the MCU exits stop mode (provided it is an exit resulting from a viable interrupt source). If the stop mode is exited by a reset, then the appropriate control/status bits are cleared and the SPI is disabled. If the device is in slave mode when the STOP instruction is executed, the slave SPI will still operate. It can still accept data and clock information in addition to transmitting its own data back to a master device.

At the end of a possible transmission with a slave SPI in stop mode, no flags are set until a viable interrupt results in an MCU wake up. Be cautious when operating the SPI (as a slave) during stop mode because none of the protection circuitry (write collision, mode fault, etc.) is active.

Also note that when the MCU enters stop mode, all enabled output drivers (MISO, MOSI, and SCLK ports) remain active and any sourcing currents from these outputs will be part of the total supply current required by the device.

10.8 SPI in Wait Mode

The SPI subsystem remains active in wait mode. Therefore, it is consuming power. Before reducing power, the SPI should be shut off prior to entering wait mode. A non-reset exit from wait mode will result in the state of the SPI being unchanged. A reset exit will return the SPI to its reset state, which is disabled.

Section 11. Pulse Width Modulators (PWMs)

11.1 Contents

11.2 Introduction

The pulse width modulator (PWM) system has two 6-bit PWMs (PWMA and PWMB). Preceding the 6-bit (÷64) counters are two programmable prescalers.The PWM frequency is selected by choosing the desired divide option from the programmable prescalers. Note that the PWM clock input is f_{OP}. The PWM frequency will be f_{OP}/(PSA*(PSB-1)*64) where PSA and PSB are the values selected by the A and B prescaler and 64 comes from the 6-bit modulus counter. See **Table 11-1** for precise values. The f_{OP} is the internal bus frequency fixed to half of the external oscillator frequency.

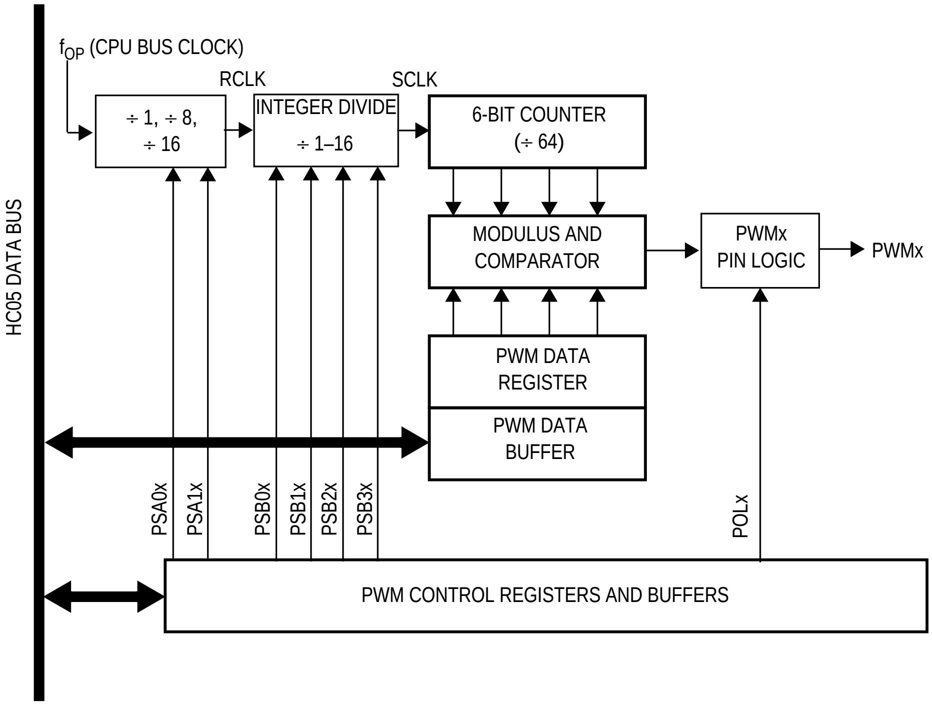


Figure 11-1. PWM Block Diagram

11.3 PWM Functional Description

The PWM is capable of generating signals from 0 percent to 100 percent duty cycle. A $00 in the PWM data register yields a low output (0 percent), but a $3F yields a duty of 63/64. To achieve the 100 percent duty (high output), the polarity control bit is set to 0 while the data register has $00 in it.

When not in use, the PWM system can be shut off to save power by clearing the clock rate select bits PSA0x and PSA1x in PWM control registers.

Writes to the PWM data registers are buffered and can, therefore, be performed at any time without affecting the output signal. When the PWM subsystem is enabled, a write to the PWM control register will become effective immediately.

When the PWM subsystem is enabled, a write to the PWM data register will not become effective until the end of the current PWM period has occurred, at which time the new data value is loaded into the PWM data register.

However, should a write to the registers be performed when the PWM subsystem is disabled, the data is transferred immediately. All registers are updated after the PWM data register is written to and the end of a PWM cycle occurs.

The PWM output can have an active high or an active low pulse under software control using the POL (polarity) bit as shown in **Figure 11-2** and **Figure 11-3**.

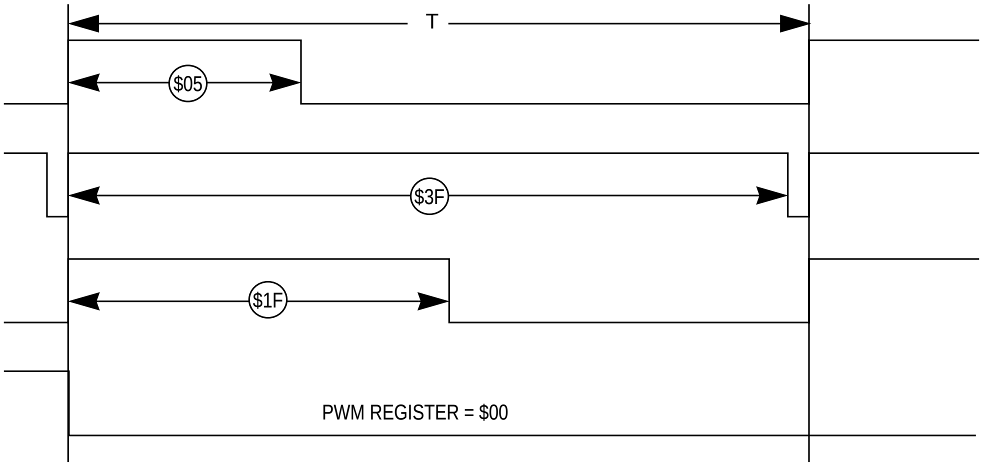


Figure 11-2. PWM Waveform Examples (POL = 1)

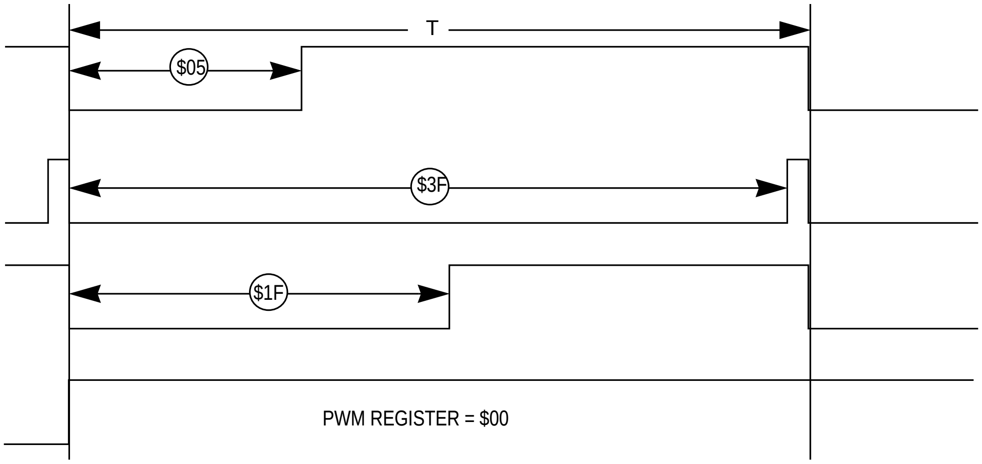


Figure 11-3. PWM Waveform Examples (POL = 0)

11.4 PWM Registers

Associated with each PWM system, there is a PWM data register and a control register. These registers can be written to and read at any time. Data written to the data register is held in a buffer and transferred to the PWM data register at the end of a PWM cycle. Reads of this register will always result in the read of the PWM data register and not the buffer.

Upon reset the user should write to the data register prior to enabling the PWM system (for example, prior to setting the PSAx and PSBx bits for PWM input clock rate). This will avoid an erroneous duty cycle from being driven. During user mode, the user should write to the PWM data register after writing the PWM control register.

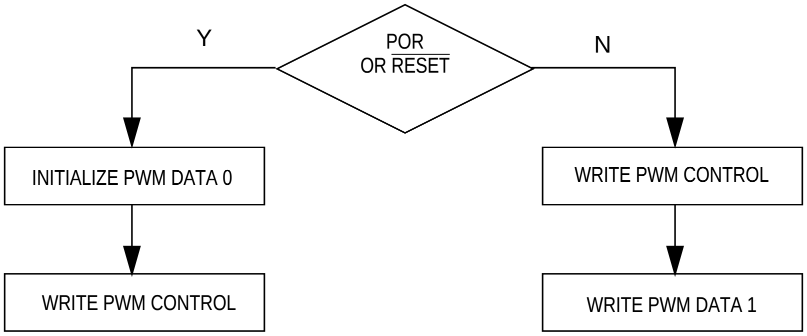


Figure 11-4. PWM Write Sequences

11.4.1 PWMA Control Register

Address: $0037

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PSA1A	PSA0A	0	0	PSB3A	PSB2A	PSB1A	PSB0A
Write:			▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 11-5. PWMA Control Register (PWMAC)

PSA1A, PSA0A, PSB3A–PSB0A — PWMA Clock Rate Bits

These bits select the input clock rate and determine the period as shown in **Table 11-1**. Note that some output frequencies can be obtained with more than one combination of PSA and PSB values. For instance, a PWMA output of f_{OP}/512 can be obtained with either PSA–PSA0 = 10 and PSB3–PSB0 = $0 or PSA1–PSA0 = 01 and PSB3–PSB0 = $07. The frequency division provided by the PSB values will be one more that the value written to the register. For example, a $0 written to the PSB bits provides a ÷1 and a $1 provides a ÷2, etc.

This scheme allows for 38 unique frequency selections.

NOTE: *Any non-zero value of PSA1A–PSA0A forces PB4 to the PWMA output state. If PSA1A:PSA0A = 00, PB4 is determined by the port B data and data direction registers as described in* ***Section 7. Parallel Input/Output (I/O)****.*

Table 11-1. PWMA Clock Rates

PSA1A–PSA0A	PSB3A–PSB0A	RCLKA	SCLKA	PWMA OUT
00	xxxx	Off	Off	Off
01	0000–1111	$f_{OP}/1$	$f_{OP}/1$ – $f_{OP}/16$	$f_{OP}/64$ – $f_{OP}/1024$
10	0000–1111	$f_{OP}/8$	$f_{OP}/8$ – $f_{OP}/128$	$f_{OP}/512$ – $f_{OP}/8192$
11	0000–1111	$f_{OP}/16$	$f_{OP}/16$ – $f_{OP}/256$	$f_{OP}/1024$ – $f_{OP}/16384$

11.4.2 PWMB Control Register

Address: $0039

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PSA1B	PSA0B	0	0	PSB3B	PSB2B	PSB1B	PSB0B
Write:			▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 11-6. PWMB Control Register (PWMBC)

PSA1B, PSA0B, and PSB3B–PSB0B — PWM Clock Rate Bits

These bits select the input clock rate for PWMB and determine the period as shown in **Table 11-2**. These bits function exactly the same as the corresponding bits in the PWMA control register except they affect the PWMB output pin.

Table 11-2. PWMB Clock Rates

PSA1B–PSA0B	PSB3B–PSB0B	RCLKB	SCLKB	PWMB OUT
00	xxxx	Off	Off	Off
01	0000–1111	$f_{OP}/1$	$f_{OP}/1$ – $f_{OP}/16$	$f_{OP}/64$ – $f_{OP}/1024$
10	0000–1111	$f_{OP}/8$	$f_{OP}/8$ – $f_{OP}/128$	$f_{OP}/512$ – $f_{OP}/8192$
11	0000–1111	$f_{OP}/16$	$f_{OP}/16$ – $f_{OP}/256$	$f_{OP}/1024$ – $f_{OP}/16384$

NOTE: *Any non-zero value of PSA1B–PSA0B forces PB5 to the PWMB output state. If PSA1B–PSA0B = 00, PB5 is determined by the port B data and data direction registers as described in **Section 7. Parallel Input/Output (I/O)**.*

11.4.3 PWMA Data Register

The PWMA system has one 6-bit data register which holds the duty cycle information. The data bits in this register are unaffected by reset. A value of $00 in this register corresponds to a steady state output level (0 percent duty cycle) on the PWMA pin. The logic level of the output will depend on the value of the POLA bit in the PWMA control register.

Address: $0036

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POLA	0	D5	D4	D3	D2	D1	D0
Write:		▨						
Reset:	0	0	U	U	U	U	U	U

▨ = Unimplemented U = Unaffected

Figure 11-7. PWMA Data Register (PWMAD)

POLA — PWMA Polarity Bits
1 = PWMA pulse is active high.
0 = PWMA pulse is active low.

11.4.4 PWMB Data Register

The PWMB system has one 6-bit data register which holds the duty cycle information. These bits work the same way as the data bits in the PWMA data register except they affect the PWMB output pin. The data bits in this register are unaffected by reset.

Address: $0038

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POLB	0	D5	D4	D3	D2	D1	D0
Write:		▨						
Reset:	0	0	U	U	U	U	U	U

▨ = Unimplemented U = Unaffected

Figure 11-8. PWMB Data Register (PWMBD)

POLB — PWMB Polarity Bit
1 = PWMB pulse is active high.
0 = PWMB pulse is active low.

11.5 PWMs during Wait Mode

The PWM continues normal operation during wait mode. To decrease power consumption during wait mode, it is recommended that the rate select bits in the PWM control registers be cleared if the PWM is not being used.

11.6 PWMs during Stop Mode

In stop mode, the oscillator is stopped causing the PWM to cease functioning. Any signal in process is aborted in whatever phase the signal happens to be in.

11.7 PWMs during Reset

Upon reset the PSA0X and PSA1X bits in PWMX control registers are cleared. This disables the PWM system and sets the PWM outputs low. The user should write to the data registers prior to enabling the PWM system (for example, prior to setting PSA1X or PSA0X). This will avoid an erroneous duty cycle from being driven.

Section 12. EPROM and EEPROM

12.1 Contents

12.2 Introduction

The MC68HC705V12 contains:

- Erasable programmable read-only memory (EPROM)
- Electrically erasable programmable read-only memory (EEPROM)

This section describes the programming mechanisms for each type of memory.

12.3 EPROM Bootloader

Bootloader programming mode is entered upon the rising edge of $\overline{\text{RESET}}$ if the $\overline{\text{IRQ}}/V_{PP}$ pin is at V_{TST} and the PA6 pin is at logic 1. The bootloader code resides in the ROM from $3C00 to $3FEF. This program handles copying of user code from an external EPROM into the on-chip EPROM.

The user code must be a one-to-one correspondence with the internal EPROM addresses (including the mask option register (MOR)).

12.4 Bootloader Functions

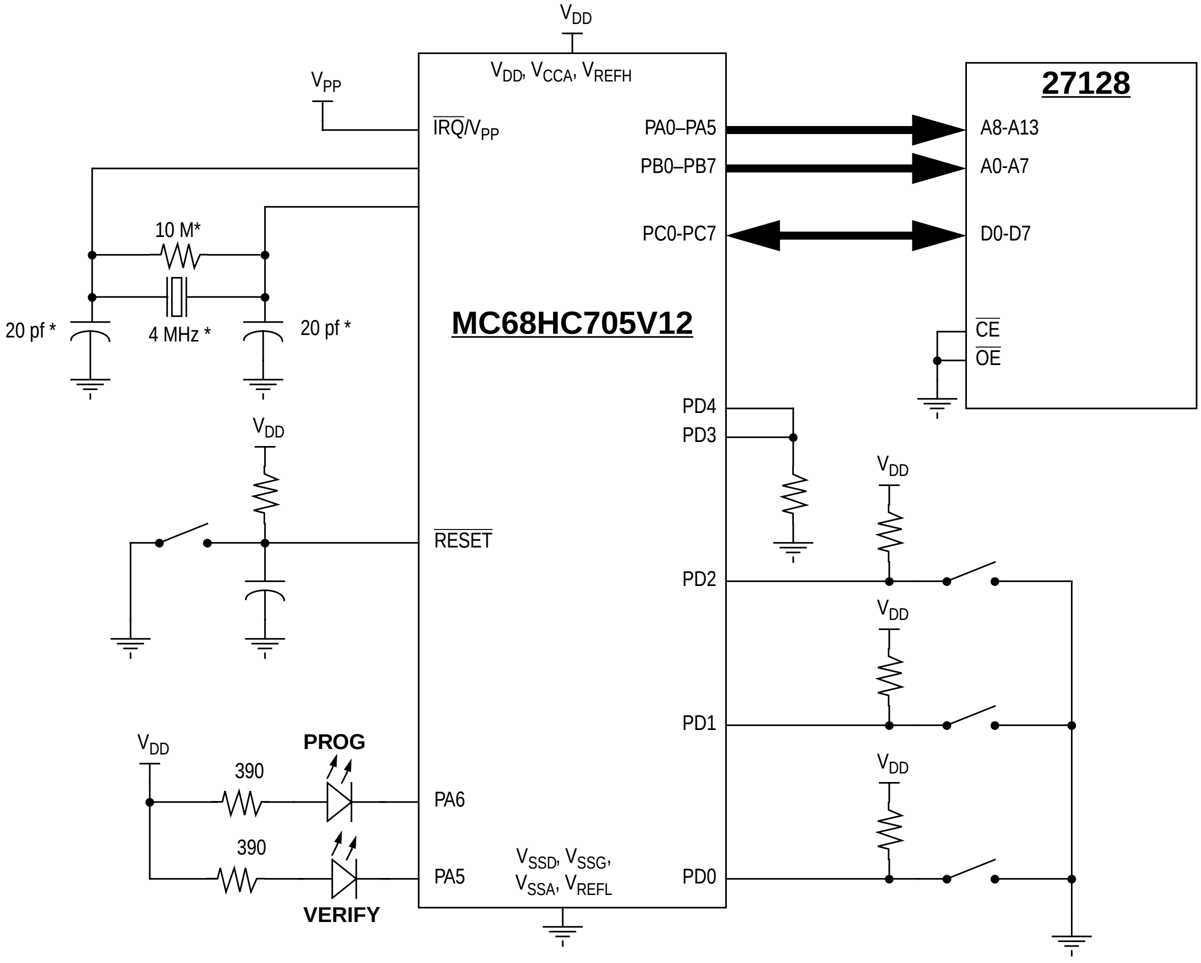
Three pins are used to select various bootloader functions. These pins are PD2, PD1, and PD0. PD3 and PD4 are tied to logic 0. Two other pins, PA6 and PA5, are used to drive the PROG LED and the VERF LED, respectively. The programming modes are shown in **Table 12-1**.

Table 12-1. Bootloader Functions

PD2	PD1	PD0	Mode
0	0	0	Program/verify EPROM
0	0	1	Verify only
0	1	0	Factory use
0	1	1	Jump to top of EEPROM
1	0	0	Jump to top of RAM
1	0	1	Jump to top of EPROM

12.5 EPROM Programming

The EPROM array is programmed through manipulation of the programming register located at $000D. The schematic for the EPROM programmer using the bootstrap firmware is shown in **Figure 12-1**. In addition to the main EPROM array, the mask option register also must be programmed appropriately by the programming software.



Note: All resistors are 10 k unless specified otherwise.

Figure 12-1. Bootstrap EPROM Programmer Schematic

12.6 EPROM Programming Register

This register is used to program the EPROM array. To program a byte of EPROM, set ELAT, write data to the desired address, then set EPGM for t_{EPGM}.

Address: $000D

	BIt 7	6	5	4	3	2	1	Bit 0
Read:	MORON	0	0	0	0	ELAT	0	EPGM
Write:		R	R	R	R		▨	
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 12-2. EPROM Programming Register (EPROG)

MORON — Mask Option Register On Bit

This bit enables and disables the decoding of the MOR.

1 = The MOR contents are placed into the memory map at location $3C00.

0 = The first byte of boot ROM will be read from location $3C00.

The contents of the MOR register can be read/written only if this bit is set and is available in any operating mode. This bit must be set when the MOR byte is being programmed.

ELAT — EPROM Latch Control Bit

This bit latches the address and data bus when a write to the EPROM array is performed.

1 = EPROM address and data bus configured for programming

0 = EPROM address and data bus configured for normal reads

NOTE: *The EPROM array cannot be read while this bit is set.*

EPGM — EPROM Program Control Bit

This bit controls the programming voltage to the EPROM array. EPGM cannot be set if ELAT is not already set. EPGM is cleared automatically when ELAT = 0.

1 = Programming power switched on to the EPROM array

0 = Programming power switched off to the EPROM array

NOTE: *ELAT and EPGM cannot both be set on the same write.*

The sequence for programming the EPROM is:

1. Set the ELAT bit. If programming the MOR byte, also set the MORON bit.
2. Write the data to be programmed to the EPROM (or MOR byte) location to be programmed.
3. Set the EPGM bit.
4. Wait a time, t_{EPGM}.
5. Clear the ELAT, MORON (if programming the MOR byte), and EPGM bits.
6. Repeat for each byte.

12.7 Mask Option Register

The mask option register (MOR) is used to select all mask options available on the MC68HC705V12.   When in the erased state, the EPROM cells will read as a logic zero which will, therefore, represent the value transferred from the MOR at reset if it is left unprogrammed. The unimplemented bits of this register are read as 0.

Address: $3C00

	BIt 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	LVRE	0	STOPE	LEVEL	COPE
Write:	▨	▨	▨		▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 12-3. Mask Option Register (MOR)

NOTE: *Options are disabled while the MOR is programmed (MORON = ELAT = EPGM = 1 in EPROM programming register).*

LVRE — Low-Voltage Reset Enable Bit
1 = LVR enabled
0 = LVR disable

STOPE — STOP Instruction Enable Bit
1 = Stop mode enabled
0 = Stop mode disabled; if STOP instruction is executed, a chip reset will result.

LEVEL — Interrupt Request Pin Sensitivity Bit
1 = $\overline{\text{IRQ}}$/V$_{PP}$ pin is both negative edge and level sensitive.
0 = $\overline{\text{IRQ}}$/V$_{PP}$ pin is negative edge sensitive only.

COPE — COP Timer Enable Bit
0 = COP timer enabled
1 = COP timer disabled

12.8 EEPROM Programming Register

The contents and use of the programming register are discussed here.

Address: $001C

	BIt 7	6	5	4	3	2	1	Bit 0
Read:	0	CPEN	0	ER1	ER0	EELAT	EERC	EEPGM
Write:	▨		▨					
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 12-4. EEPROM Programming Register (EEPROG)

NOTE: *Any reset including low-voltage reset (LVR) will abort any write in progress when it is asserted. Data written to the addressed byte will, therefore, be indeterminate.*

CPEN — Charge Pump Enable Bit

When set, CPEN enables the charge pump which produces the internal programming voltage. This bit should be set with the EELAT bit. The programming voltage will not be available until EEPGM is set. The charge pump should be disabled when not in use. CPEN is readable and writable and is cleared by reset.

ER1–ER0 — Erase Select Bits

ER1 and ER0 form a 2-bit field which is used to select one of three erase modes: byte, block, or bulk. **Table 12-2** shows the modes selected for each bit configuration. These bits are readable and writable and are cleared by reset.

Table 12-2. Erase Mode Select

ER1	ER0	Mode
0	0	No erase
0	1	Byte erase
1	0	Block erase
1	1	Bulk erase

In byte erase mode, only the selected byte is erased. In block mode, a 64-byte block of EEPROM is erased. The EEPROM memory space is divided into four 64-byte blocks (\$0240–\$027F, \$0280–\$02BF, \$02C0–\$02FF, and \$0300–\$033F), and doing a block erase to any address within a block will erase the entire block. In bulk erase mode, the entire 256-byte EEPROM section is erased.

EELAT — EEPROM Programming Latch Bit

When set, EELAT configures the EEPROM address and data bus for programming. When EELAT is set, writes to the EEPROM array cause the data bus and the address bus to be latched. This bit is readable and writable, but reads from the array are inhibited if the EELAT bit is set and a write to the EEPROM space has taken place. When clear, address and data buses are configured for normal operation. Reset clears this bit.

EERC — EEPROM RC Oscillator Control Bit

When this bit is set, the EEPROM section uses the internal RC oscillator instead of the CPU clock. After setting the EERC bit, delay a time, t_{RCON}, to allow the RC oscillator to stabilize. This bit is readable and writable and should be set by the user when the internal bus frequency falls below 1.5 MHz. Reset clears this bit.

EEPGM — EEPROM Programming Power Enable Bit

EEPGM must be written to enable (or disable) the EEPGM function. When set, EEPGM turns on the charge pump and enables the programming (or erasing) power to the EEPROM array. When clear, this power is switched off. This will enable pulsing of the programming voltage to be controlled internally. This bit can be read at any time, but can only be written to if EELAT = 1. If EELAT is not set, then EEPGM cannot be set. Reset clears this bit.

12.9 EEPROM Programming/Erasing Procedure

To program a byte of EEPROM:

1. Set EELAT = CPEN = 1.
2. Set ER1 = ER0 = 0.
3. Write data to the desired address.
4. Set EEPGM for a time, t_{EEPGM}.

In general, all bits should be erased before being programmed. However, if write/erase cycling is a concern, a procedure can be followed to minimize the cycling of each bit in each EEPROM byte. The erased state is 1; therefore, if any bits within the byte need to be changed from a 0 to a 1, the byte must be erased before programming. The decision whether to erase a byte before programming is summarized in **Table 12-3**.

Table 12-3. EEPROM Write/Erase Cycle Reduction

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

To erase a **byte** of EEPROM:

1. Set EELAT = 1, CPEN = 1, ER1 = 0, and ER0 = 1.
2. Write to the address to be erased.
3. Set EEPGM for a time, t_{EBYT}.

To erase a **block** of EEPROM:

1. Set EELAT = 1, CPEN = 1, ER1 = 1, and ER0 = 0.
2. Write to any address in the block.
3. Set EEPGM for a time, t_{EBLOCK}.

For a **bulk** erase:

1. Set EELAT = 1, CPEN = 1, ER1 = 1, and ER0 = 1.
2. Write to any address in the array.
3. Set EEPGM for a time, t_{EBULK}.

To terminate the programming or erase sequence, clear EEPGM, delay for a time, $t_{FPV,}$ to allow the programming voltage to fall, and then clear EELAT and CPEN to free up the buses. Following each erase or programming sequence, clear all programming control bits.

12.10 Operation in Stop Mode and Wait Mode

The RC oscillator for the EEPROM is disabled automatically when entering stop mode. To help conserve power, the user should disable the RC oscillator before entering wait mode.

Section 13. Analog-to-Digital (A/D) Converter

13.1 Contents

13.2 Introduction

The MC68HC705V12 includes a 5-channel, 8-bit, multiplexed input and a successive approximation analog-to-digital (A/D) converter.

13.3 Analog Section

This subsection describes the analog section.

13.3.1 Ratiometric Conversion

The A/D is ratiometric, with two dedicated pins supplying the reference voltages (V_{REFH} and V_{REFL}). An input voltage equal to V_{REFH} converts to $FF (full scale) and an input voltage equal to V_{REFL} converts to $00. An input voltage greater than V_{REFH} will convert to $FF with no overflow indication. For ratiometric conversions, the source of each analog input should use V_{REFH} as the supply voltage and be referenced to V_{REFL}.

13.3.2 V_{REFH} and V_{REFL}

The reference supply for the A/D is two dedicated pins rather than being driven by the system power supply lines. The voltage drops in the bonding wires of the heavily loaded system power pins would degrade the accuracy of the A/D conversion. V_{REFH} and V_{REFL} can be any voltage between V_{SSA} and V_{CCA}, as long as$V_{REFH} > V_{REFL}$; however, the accuracy of conversions is tested and guaranteed only for $V_{REFL} = V_{SSA}$ and $V_{REFH} = V_{CCA}$.

13.3.3 Accuracy and Precision

The 8-bit conversions shall be accurate to within ± 1 least significant bit (LSB) including quantization.

13.3.4 Conversion Process

The A/D reference inputs are applied to a precision internal digital-to-analog (D/A) converter. Control logic drives this D/A and the analog output is compared successively to the selected analog input which was sampled at the beginning of the conversion time. The conversion process is monotonic and has no missing codes.

13.4 Digital Section

This subsection describes the digital section.

13.4.1 Conversion Times

Each channel of conversion takes 32 clock cycles, which must be at a frequency equal to or greater than 1 MHz.

13.4.2 Internal and Master Oscillators

If the MCU bus (f_{OP}) frequency is less than 1.0 MHz, an internal RC oscillator (nominally 1.5 MHz) must be used for the A/D conversion clock. This selection is made by setting the ADRC bit in the A/D status and control registers to 1. In stop mode, the internal RC oscillator is turned off automatically, although the A/D subsystem remains enabled (ADON remains set). In wait mode the A/D subsystem remains functional. See **13.7 A/D during Wait Mode**.

When the internal RC oscillator is being used as the conversion clock, three limitations apply:

1. The conversion complete flag (COCO) must be used to determine when a conversion sequence has been completed, due to the frequency tolerance of the RC oscillator and its asynchronism with regard to the MCU bus clock.
2. The conversion process runs at the nominal 1.5 MHz rate, but the conversion results must be transferred to the MCU result registers synchronously with the MCU bus clock so conversion time is limited to a maximum of one channel per bus cycle.
3. If the system clock is running faster than the RC oscillator, the RC oscillator should be turned off and the system clock used as the conversion clock.

13.4.3 Multi-Channel Operation

A multiplexer allows the A/D converter to select one of five external analog signals and four internal reference sources.

13.5 A/D Status and Control Register

This subsection describes the function of the A/D status and control register.

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COCO	ADRC	ADON	CH4	CH3	CH2	CH1	CH0
Write:	▨							
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 13-1. A/D Status and Control Register (ADSCR)

COCO — Conversions Complete Bit

This read-only status bit is set when a conversion is completed, indicating that the A/D data register contains valid results. This bit is cleared whenever the A/D status and control register is written and a new conversion automatically started, or whenever the A/D data register is read. Once a conversion has been started by writing to the A/D status and control register, conversions of the selected channel will continue every 32 cycles until the A/D status and control register is written again. In this continuous conversion mode the A/D data register will be filled with new data, and the COCO bit set, every 32 cycles. Data from the previous conversion will be overwritten regardless of the state of the COCO bit prior to writing.

ADRC — RC Oscillator Control Bit

When ADRC is set, the A/D section runs on the internal RC oscillator instead of the CPU clock. The RC oscillator requires a time, t_{RCON}, to stabilize and results can be inaccurate during this time.

ADON — A/D On Bit

When the A/D is turned on (ADON = 1), it requires a time, t_{ADON}, for the current sources to stabilize, and results can be inaccurate during this time. This bit turns on the charge pump.

CH4–CH0 — Channel Select Bits

CN4, CH3, CH2, CH1, and CH0 form a 5-bit field which is used to select one of nine A/D channels, including four internal references. Channels $00–04 correspond to port D input pins on the MCU. Channels $10–$13 are used for internal reference points. In single-chip mode, channel $13 is reserved and converts to $00. **Table 13-1** shows the signals selected by the channel select field.

Table 13-1. A/D Channel Assignments

CH4–CH0	Signal
00–04	AD0–AD4
$10	V_{REFH}
$11	$(V_{REFH}–V_{REFL})/2$
$12	V_{REFL}
$13	Factory test
$05–$0F, $14–$1F	Unused

13.6 A/D Data Register

An 8-bit result register is provided. This register is updated each time the COCO bit is set.

Address: $001D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	D7	D6	D5	D4	D3	D2	D1	D0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	Unaffected by reset							

▨ = Unimplemented

Figure 13-2. A/D Data Register (ADDR)

13.7 A/D during Wait Mode

The A/D converter continues normal operation during wait mode. To decrease power consumption during wait mode, it is recommended that both the ADON and ADRC bits in the A/D status and control registers be cleared if the A/D converter is not being used. If the A/D converter is in use and the system clock rate is above 1.0 MHz, it is recommended that the ADRC bit be cleared.

NOTE: *As the A/D converter continues to function normally in wait mode, the COCO bit is not cleared.*

13.8 A/D during Stop Mode

In stop mode, the comparator and charge pump are turned off and the A/D ceases to function. Any pending conversion is aborted. When the clocks begin oscillation upon leaving stop mode, a finite amount of time passes before the A/D circuits stabilize enough to provide conversions to the specified accuracy. Normally, the delays built into the device when coming out of stop mode are sufficient for this purpose so that no explicit delays need to be built into the software.

NOTE: *Although the comparator and charge pump are disabled in stop mode, the A/D data and status/control registers are not modified. Disabling the A/D prior to entering stop mode will not affect the stop mode current consumption.*

Section 14. Byte Data Link Controller – Digital (BDLC–D)

14.1 Contents

14.2 Introduction

The byte data link controller (BDLC) provides access to an external serial communication multiplex bus, operating according to the SAE J1850 protocol.

14.3 Features

Features of the BDLC module include:

- *SAE J1850 Class B Data Communications Network Interface* compatible and ISO compatible for low-speed (≤125 kbps) serial data communications in automotive applications
- 10.4 kbps variable pulse width (VPW) bit format
- Digital noise filter
- Collision detection
- Hardware cyclical redundancy check (CRC) generation and checking
- Two power-saving modes with automatic wakeup on network activity
- Polling or CPU interrupts
- Block mode receive and transmit supported
- 4X receive mode, 41.6 kbps, supported
- Digital loopback mode
- Analog loopback mode
- In-frame response (IFR) types 0, 1, 2, and 3 supported

14.4 Functional Description

Figure 14-1 shows the organization of the BDLC module. The CPU interface contains the software addressable registers and provides the link between the CPU and the buffers. The buffers provide storage for data received and data to be transmitted onto the J1850 bus. The protocol handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX interface provides the link between the BDLC digital section and the analog physical interface. The wave shaping, driving, and digitizing of data is performed by the physical interface.

Use of the BDLC module in message networking fully implements the *SAE Standard J1850 Class B Data Communication Network Interface* specification.

NOTE: *It is recommended that the reader be familiar with the SAE J1850 document and ISO serial communication document prior to proceeding with this section of the specification.*

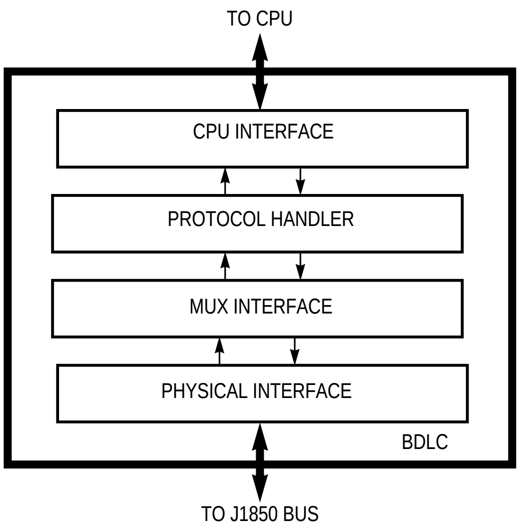


Figure 14-1. BDLC Block Diagram

.

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$003E	BDLC Analog and Roundtrip Delay Register (BARD) See page 167.	Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
		Write:			▨	▨				
		Reset:	1	1	0	0	0	1	1	1
$003A	BDLC Control Register 1 (BCR1) See page 169.	Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
		Write:					R	R		
		Reset:	1	1	1	0	0	0	0	0
$003B	BDLC Control Register 2 (BCR2) See page 171.	Read:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$003C	BDLC State Vector Register (BSVR) See page 179.	Read:	0	0	I3	I2	I1	I0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$003D	BDLC Data Register (BDR) See page 181.	Read:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
		Write:								
		Reset:	Indeterminate after reset							

▨ = Unimplemented   R = Reserved

Figure 14-2. BDLC Input/Output (I/O) Register Summary

14.4.1 BDLC Operating Modes

The BDLC has five main modes of operation which interact with the power supplies, pins, and reset of the MCU as shown in **Figure 14-3**.

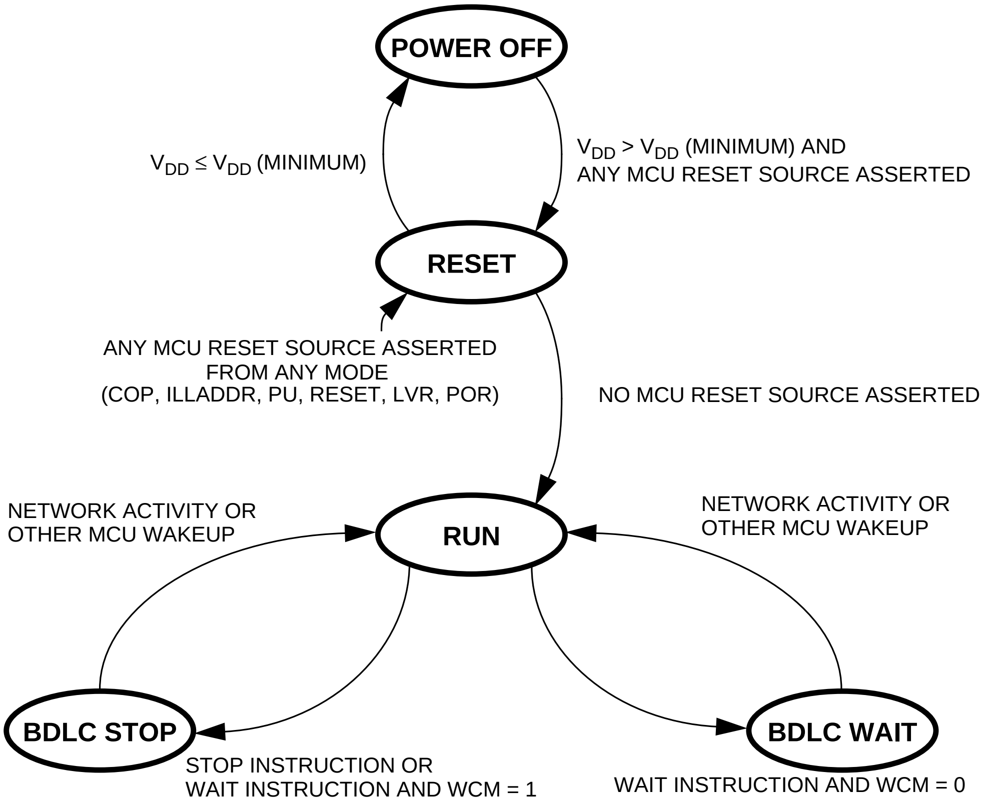

POWER OFF
$V_{DD} \leq V_{DD}$ (MINIMUM)
$V_{DD} > V_{DD}$ (MINIMUM) AND
ANY MCU RESET SOURCE ASSERTED
RESET
ANY MCU RESET SOURCE ASSERTED
FROM ANY MODE
(COP, ILLADDR, PU, RESET, LVR, POR)
NO MCU RESET SOURCE ASSERTED
NETWORK ACTIVITY OR
OTHER MCU WAKEUP
RUN
NETWORK ACTIVITY OR
OTHER MCU WAKEUP
BDLC STOP
BDLC WAIT
STOP INSTRUCTION OR
WAIT INSTRUCTION AND WCM = 1
WAIT INSTRUCTION AND WCM = 0

Figure 14-3. BDLC Operating Modes State Diagram

14.4.1.1 Power Off Mode

For the BDLC to guarantee operation, this mode is entered from reset mode whenever the BDLC supply voltage, V_{DD}, drops below its minimum specified value. The BDLC will be placed in reset mode by low-voltage reset (LVR) before being powered down. In power off mode, the pin input and output specifications are not guaranteed.

14.4.1.2 Reset Mode

This mode is entered from power off mode whenever the BDLC supply voltage, V_{DD}, rises above its minimum specified value (V_{DD} –10%) and some MCU reset source is asserted. The internal MCU reset must be asserted while powering up the BDLC or an unknown state will be entered and correct operation cannot be guaranteed. Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (such as LVR, POR, COP watchdog, reset pin, etc.) is asserted.

In reset mode, the internal BDLC voltage references are operative, V_{DD} is supplied to the internal circuits which are held in their reset state, and the internal BDLC system clock is running. Registers will assume their reset condition. Because outputs are held in their programmed reset state, inputs and network activity are ignored.

14.4.1.3 Run Mode

This mode is entered from reset mode after all MCU reset sources are no longer asserted. Run mode is entered from the BDLC wait mode whenever activity is sensed on the J1850 bus.

Run mode is entered from the BDLC stop mode whenever network activity is sensed, although messages will not be received properly until the clocks have stabilized and the CPU is also in run mode.

In this mode, normal network operation takes place. The user should ensure that all BDLC transmissions have ceased before exiting this mode.

14.4.1.4 BDLC Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the BCR1 register is cleared previously.

In this mode, the BDLC internal clocks continue to run. The first passive-to-active transition of the bus generates a CPU interrupt request from the BDLC, which wakes up the BDLC and the CPU. In addition, if the BDLC receives a valid end-of-frame (EOF) symbol while operating in wait mode, then the BDLC also will generate a CPU interrupt request, which wakes up the BDLC and the CPU. See **14.8.1 Wait Mode**.

14.4.1.5 BDLC Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BCR1 is set previously.

In this mode, the BDLC internal clocks are stopped but the physical interface circuitry is placed in a low-power mode and awaits network activity. If network activity is sensed, then a CPU interrupt request will be generated, restarting the BDLC internal clocks. See **14.8.2 Stop Mode**.

14.4.1.6 Digital Loopback Mode

When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node’s internal circuits or elsewhere in the network, including the node’s analog physical interface. In this mode, the transmit digital output pin (BDTxD) and the receive digital input pin (BDRxD) of the digital interface are disconnected from the analog physical interface and tied together to allow the digital portion of the BDLC to transmit and receive its own messages without driving the J1850 bus.

14.4.1.7 Analog Loopback Mode

Analog loopback mode is used to determine if a bus fault has been caused by a failure in the node's off-chip analog transceiver or elsewhere in the network. The BDLC analog loopback mode does not modify the digital transmit or receive functions of the BDLC. It does, however, ensure that once analog loopback mode is exited, the BDLC will wait for an idle bus condition before participation in network communication resumes. If the off-chip analog transceiver has a loopback mode, it usually causes the input to the output drive stage to be looped back into the receiver, allowing the node to receive messages it has transmitted without driving the J1850 bus. In this mode, the output to the J1850 bus typically is high impedance. This allows the communication path through the analog transceiver to be tested without interfering with network activity. Using the BDLC analog loopback mode in conjunction with the analog transceiver's loopback mode ensures that, once the off-chip analog transceiver has exited loopback mode, the BCLD will not begin communicating before a known condition exists on the J1850 bus.

14.5 BDLC MUX Interface

The MUX interface is responsible for bit encoding/decoding and digital noise filtering between the protocol handler and the physical interface.

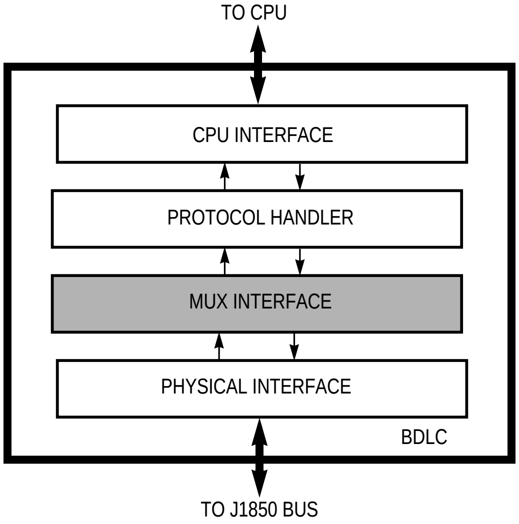


Figure 14-4. BDLC Block Diagram

14.5.1 Rx Digital Filter

The receiver section of the BDLC includes a digital low pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in **Figure 14-5**.

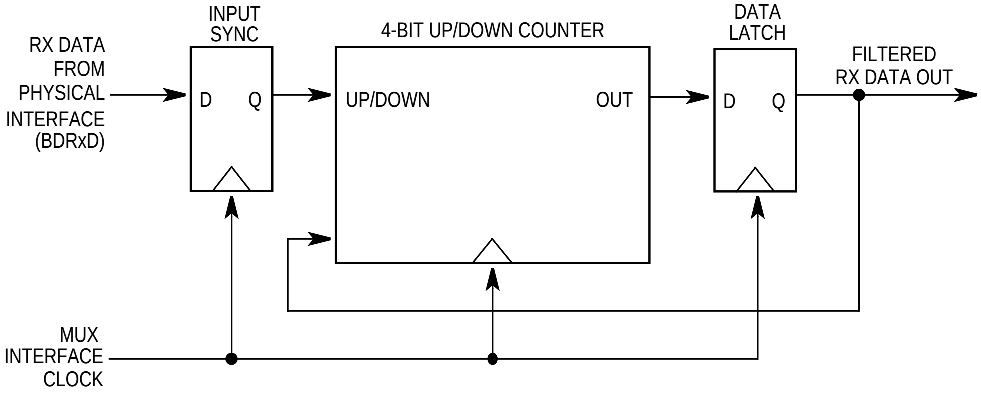


Figure 14-5. BDLC Rx Digital Filter Block Diagram

14.5.1.1 Operation

The clock for the digital filter is provided by the MUX interface clock (see f_{BDLC} parameter in **Table 14-3**). At each positive edge of the clock signal, the current state of the receiver physical interface (BDRxD) signal is sampled. The BDRxD signal state is used to determine whether the counter should increment or decrement at the next negative edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. Therefore, the counter will thus progress either up toward 15 if, on average, the BDRxD signal remains high or progress down toward 0 if, on average, the BDRxD signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can be decremented only from this state.

Alternatively, should the counter eventually reach the value 0, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can be incremented only from this state.

The data latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the signal.

14.5.1.2 Performance

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the BDRxD signal transitions, then there will be a delay before that transition appears at the filtered Rx data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This filter delay must be taken into account when performing message arbitration.

For example, if the frequency of the MUX interface clock (f_{BDLC}) is 1.0486 MHz, then the period (t_{BDLC}) is 954 ns and the maximum filter delay in the absence of noise will be 15.259 µs.

The effect of random noise on the BDRxD signal depends on the characteristics of the noise itself. Narrow noise pulses on the BDRxD signal will be ignored completely if they are shorter than the filter delay. This provides a degree of low pass filtering.

If noise occurs during a symbol transition, the detection of that transition can be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length, will be detected by the next stage of the BDLC’s receiver as an invalid symbol.

Noise pulses that are longer than the shortest allowable symbol length will be detected normally as an invalid symbol or as invalid data when the frame’s CRC is checked.

14.5.2 J1850 Frame Format

All messages transmitted on the J1850 bus are structured using the format shown in **Figure 14-6**.

J1850 states that each message has a maximum length of 101 PWM (pulse width modulation) bit times or 12 VPW (variable pulse width) bytes, excluding SOF, EOD, NB, and EOF, with each byte transmitted most significant bit (MSB) first.

All VPW symbol lengths in the following descriptions are typical values at a 10.4-kbps bit rate.

SOF — Start-of-Frame Symbol

All messages transmitted onto the J1850 bus must begin with a long-active 200-µs period SOF symbol. This indicates the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

Data — In-Message Data Bytes

The data bytes contained in the message include the message priority/type, message ID byte (typically the physical address of the responder), and any actual data being transmitted to the receiving node. The message format used by the BDLC is similar to the 3-byte consolidated header message format outlined by the SAE J1850 document. See *SAE J1850 Class B Data Communications Network Interface* for more information about 1- and 3-byte headers.

Messages transmitted by the BDLC onto the J1850 bus must contain at least one data byte, and, therefore, can be as short as one data byte and one CRC byte. Each data byte in the message is eight bits in length and is transmitted MSB to LSB (least significant bit).

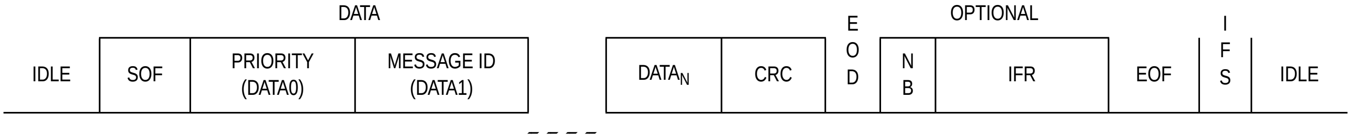


Figure 14-6. J1850 Bus Message Format (VPW)

CRC — Cyclical Redundancy Check Byte

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus. It also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial $X^8 + X^4 + X^3 + X^2 + 1$. The remainder polynomial initially is set to all 1s. Each byte in the message after the start-of-frame (SOF) symbol is processed serially through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes, in MSB-to-LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and end of data symbols (EOD) but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial will equal $X^7 + X^6 + X^2$ = $C4, regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC will recognize this as a CRC error and set the CRC error flag in the BSVR.

EOD — End-of-Data Symbol

The EOD symbol is a long 200-µs passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

IFR — In-Frame Response Bytes

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the *SAE J1850 Class B Data Communications Network Interface* specification.

EOF — End-of-Frame Symbol

This symbol is a long 280-µs passive period on the J1850 bus and is longer than an end-of-data (EOD) symbol, which signifies the end of a message. Since an EOF symbol is longer than a 200-µs EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

IFS — Inter-Frame Separation Symbol

The IFS symbol is a 20-µs passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node after the completion of the end-of-frame (EOF) period and, therefore is seen as a 300-µs passive period.

When the last byte of a message has been transmitted onto the J1850 bus and the EOF symbol time has expired, all nodes then must wait for the IFS symbol time to expire before transmitting a start-of-frame (SOF) symbol, marking the beginning of another message.

However, if the BDLC is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it will synchronize internally to that edge.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. To allow for individual clock tolerances, receivers must synchronize to any SOF occurring during an IFS period.

BREAK — Break

The BDLC cannot transmit a BREAK symbol.

If the BDLC is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred and halts transmission.

If the BDLC detects a BREAK symbol while receiving a message, it treats the BREAK as a reception error and sets the invalid symbol flag in the BSVR, also ignoring the frame it was receiving. If while receiving a message in 4X mode, the BDLC detects a BREAK symbol, it treats the BREAK as a reception error, sets the invalid symbol flag, and exits 4X mode (for example, the RX4XE bit in BCR2 is cleared automatically). If bus control is required after the BREAK symbol is received and the IFS time has elapsed, the programmer must resend the transmission byte using highest priority.

IDLE — Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period (for example, > 300 µs). Any node sensing an idle bus condition can begin transmission immediately.

14.5.3 J1850 VPW Symbols

Huntsinger's variable pulse width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions and by the level of the bus between transitions (for instance, active or passive). Active and passive bits are used alternately. This encoding technique is used to reduce the number of bus transitions for a given bit rate.

Each logic 1 or logic 0 contains a single transition and can be at either the active or passive level and one of two lengths, either 64 µs or 128 µs (t_{NOM} at 10.4 kbps baud rate), depending upon the encoding of the previous bit. The start-of-frame (SOF), end-of-data (EOD), end-of-frame (EOF), and inter-frame separation (IFS) symbols always will be encoded at an assigned level and length. See **Figure 14-7**.

Each message will begin with an SOF symbol, an active symbol, and, therefore, each data byte (including the CRC byte) will begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

All VPW bit lengths stated in the following descriptions are typical values at a 10.4-kbps bit rate. EOF, EOD, IFS, and IDLE, however, are not driven J1850 bus states. They are passive bus periods observed by each node's CPU.

Logic 0

A logic 0 is defined as either:

- An active-to-passive transition followed by a passive period 64 µs in length, or
- A passive-to-active transition followed by an active period 128 µs in length

See **Figure 14-7(a)**.

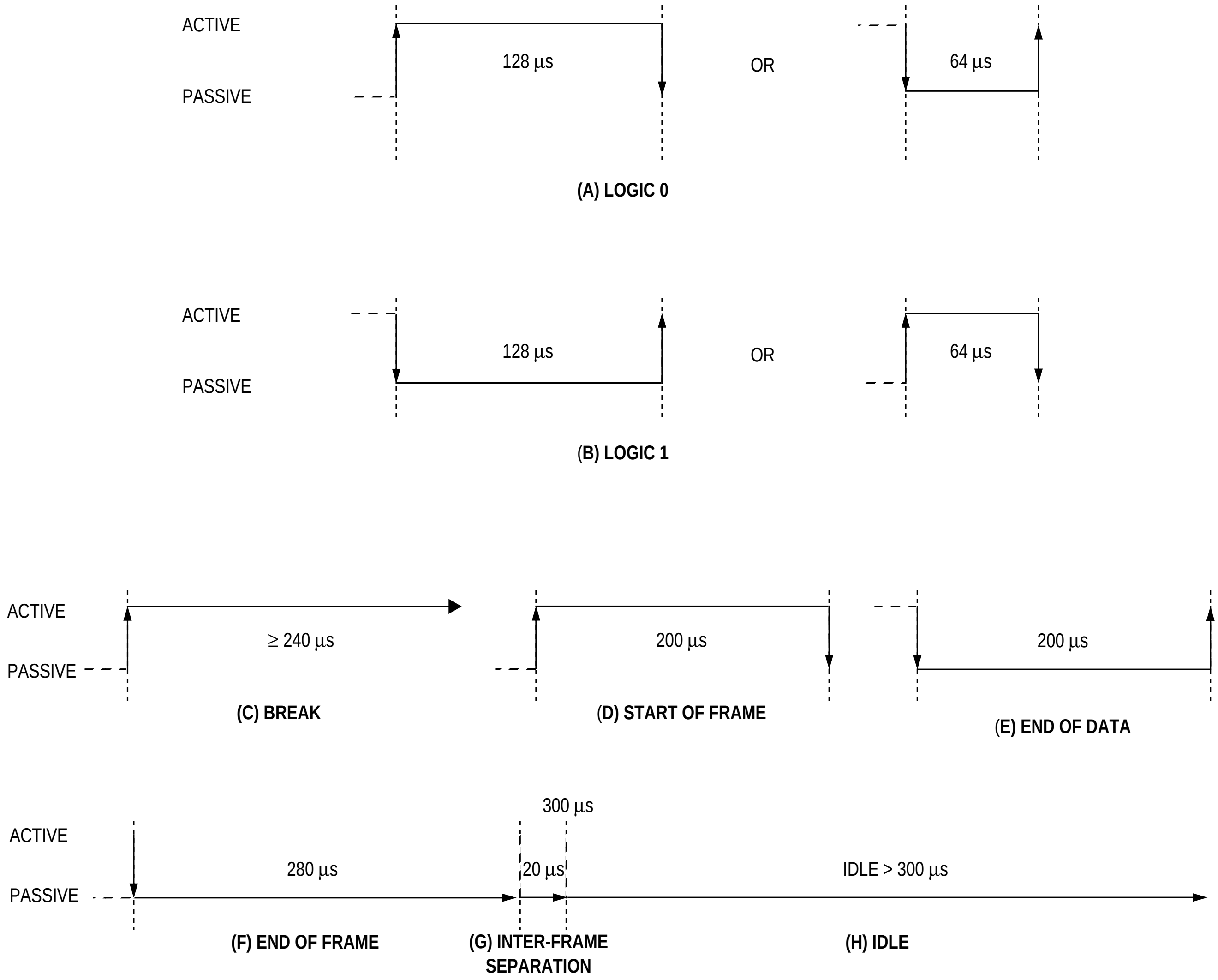


Figure 14-7. J1850 VPW Symbols with Nominal Symbol Times

Logic 1

A logic 1 is defined as either:

- An active-to-passive transition followed by a passive period 128 µs in length, or
- A passive-to-active transition followed by an active period 64 µs in length

See **Figure 14-7**(**b**).

Normalization Bit (NB)

The NB symbol has the same property as a logic 1 or a logic 0. It is only used in IFR message responses.

Break Signal (BREAK)

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 μs (see **Figure 14-7(c)**).

Start-of-Frame Symbol (SOF)

The SOF symbol is defined as passive-to-active transition followed by an active period 200 μs in length (see **Figure 14-7(d)**). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

End-of-Data Symbol (EOD)

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 μs in length (see **Figure 14-7(e)**).

End-of-Frame Symbol (EOF)

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280 μs in length (see **Figure 14-7(f)**). If no IFR byte is transmitted after an EOD symbol is transmitted, after another 80 μs the EOD becomes an EOF, indicating completion of the message.

Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 300 μs in length. The 20-μs IFS symbol contains no transition, since when it is used it always appends to a 280-μs EOF symbol (see **Figure 14-7(g)**).

Idle

An idle is defined as a passive period greater than 300 μs in length.

14.5.4 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for **receiving** data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases, the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC for determining what symbol is being received is equal to a single period of the MUX interface clock (t_{BDLC}), an apparent separation in these maximum time/minimum time concurrences equals one cycle of t_{BDLC}.

This one clock resolution allows the BDLC to differentiate properly between the different bits and symbols. This is done without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus, which has varying oscillator frequencies.

In Huntsinger's variable pulse width (VPW) modulation bit encoding, the tolerances for both the passive and active data bits received and the symbols received are defined with no gaps between definitions. For example, the maximum length of a passive logic 0 is equal to the minimum length of a passive logic 1, and the maximum length of an active logic 0 is equal to the minimum length of a valid SOF symbol.

Invalid Passive Bit

See **Figure 14-8(1)**. If the passive-to-active received transition beginning the next data bit or symbol occurs between the active-to-passive transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

Valid Passive Logic 0

See **Figure 14-8(2)**. If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 0.

Valid Passive Logic 1

See **Figure 14-8(3)**. If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 1.

Valid EOD Symbol

See **Figure 14-8(4)**. If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid end-of-data symbol (EOD).

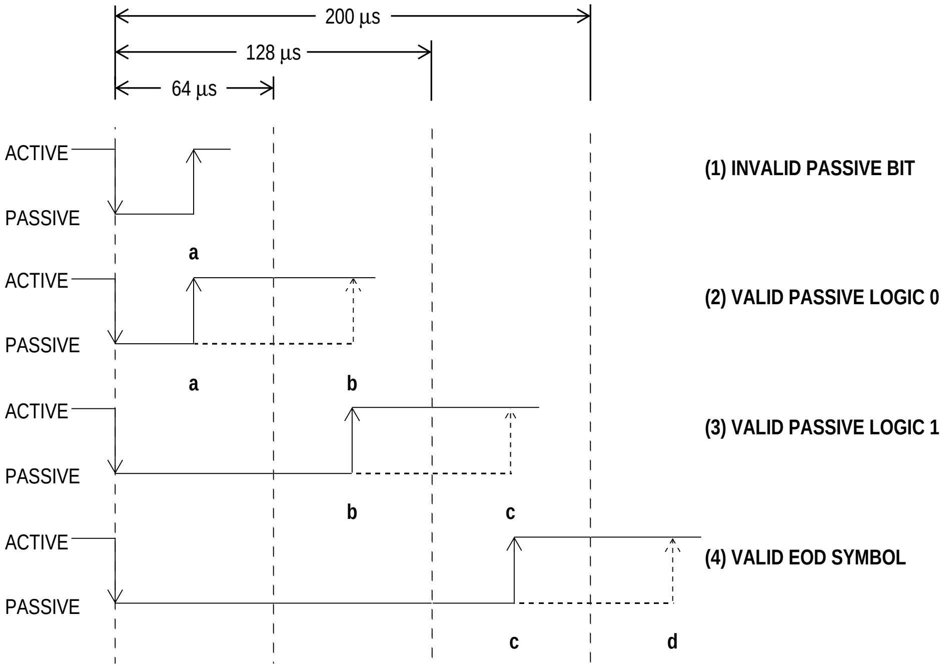


Figure 14-8. J1850 VPW Received Passive Symbol Times

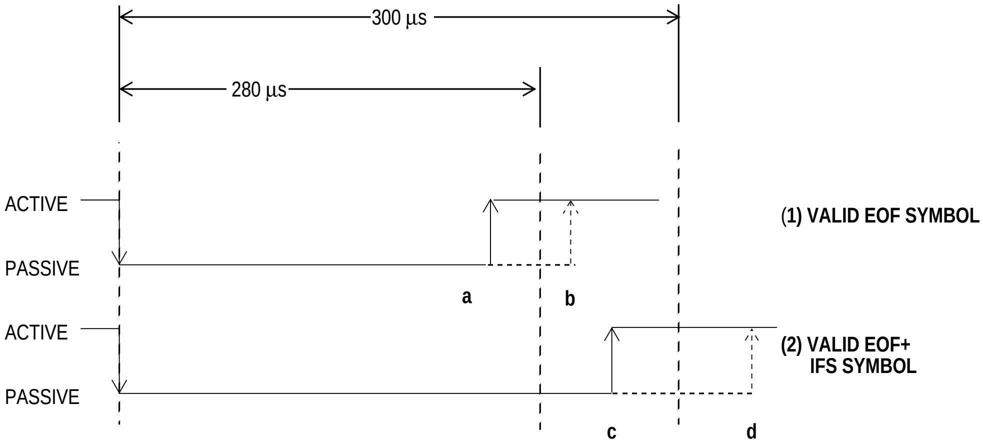


Figure 14-9. J1850 VPW Received Passive EOF and IFS Symbol Times

Valid EOF and IFS Symbols

In **Figure 14-9(1)**, if the passive-to-active received transition beginning the SOF symbol of the next message occurs between **a** and **b**, the current symbol will be considered a valid end-of-frame (EOF) symbol.

See **Figure 14-9(2)**. If the passive-to-active received transition beginning the SOF symbol of the next message occurs between **c** and **d**, the current symbol will be considered a valid EOF symbol followed by a valid inter-frame separation symbol (IFS). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others and immediately begin transmitting. Therefore, any time a node waiting to transmit detects a passive-to-active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

Idle Bus

In **Figure 14-9(2)**, if the passive-to-active received transition beginning the start-of-frame (SOF) symbol of the next message does not occur before **d,** the bus is considered to be idle, and any node wanting to transmit a message may do so immediately.

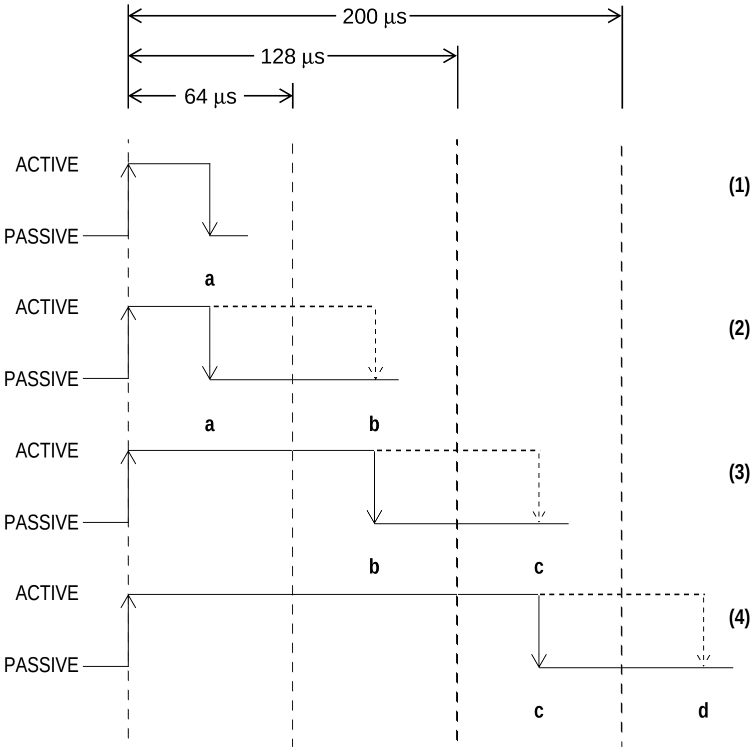


Figure 14-10. J1850 VPW Received Active Symbol Times

Invalid Active Bit

In **Figure 14-10(1)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between the passive-to-active transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

Valid Active Logic 1

In **Figure 14-10(2)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 1.

Valid Active Logic 0

In **Figure 14-10(3)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 0.

Valid SOF Symbol

In **Figure 14-10**(**4**), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid SOF symbol.

Valid BREAK Symbol

In **Figure 14-11**, if the next active-to-passive received transition does not occur until after **e**, the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by a start-of-frame (SOF) symbol beginning the next message to be transmitted onto the J1850 bus. See **14.5.2 J1850 Frame Format** for BDLC response to BREAK symbols.

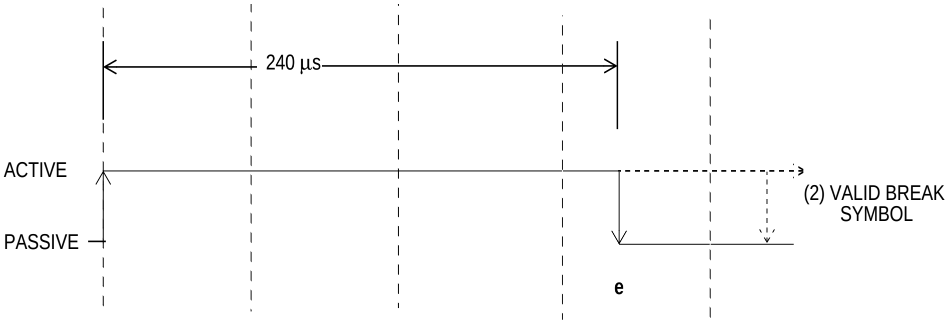


Figure 14-11. J1850 VPW Received BREAK Symbol Times

14.5.5 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC wants to transmit onto the J1850 bus, but detects that another message is in progress, it waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration will occur beginning with the first bit after the SOF symbol and continue with each bit thereafter. If a write to the BDR (for instance, to initiate transmission) occurred on or before 104 • t_{BDLC} from the received rising edge, then the BDLC will transmit and arbitrate for the bus. If a CPU write to the BDR occurred after 104 • t_{BDLC} from the detection of the rising edge, then the BDLC will not transmit, but will wait for the next IFS period to expire before attempting to transmit the byte.

The variable pulse width modulation (VPW) symbols and J1850 bus electrical characteristics are chosen carefully so that a logic 0 (active or passive type) will always dominate over a logic 1 (active or passive type) simultaneously transmitted. Hence, logic 0s are said to be dominant and logic 1s are said to be recessive.

Whenever a node detects a dominant bit on BDRxD when it transmitted a recessive bit, it loses arbitration and immediately stops transmitting. This is known as bitwise arbitration.

Since a logic 0 dominates a logic 1, the message with the lowest value will have the highest priority and will always win arbitration. For instance, a message with priority 000 will win arbitration over a message with priority 011.

This method of arbitration will work no matter how many bits of priority encoding are contained in the message.

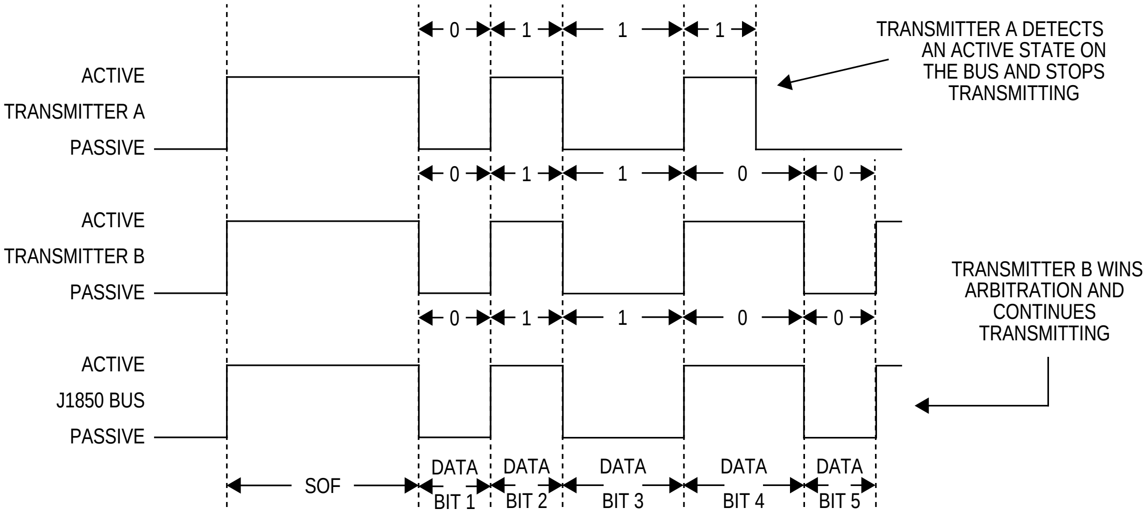


Figure 14-12. J1850 VPW Bitwise Arbitrations

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is stopped immediately unless it occurs on the 8th bit of a byte. In this case, the BDLC automatically will append up to two extra logic 1 bits and then stop transmitting. These two extra bits will be arbitrated normally and thus will not interfere with another message. The second logic 1 bit will not be sent if the first loses arbitration. If the BDLC has lost arbitration to another valid message, then the two extra logic 1s will not corrupt the current message. However, if the BDLC has lost arbitration due to noise on the bus, then the two extra logic 1s will ensure that the current message will be detected and ignored as a noise-corrupted message.

14.6 BDLC Protocol Handler

The protocol handler is responsible for framing, arbitration, CRC generation/checking, and error detection. The protocol handler conforms to *SAE J1850 Class B Data Communications Network Interface*.

NOTE: *Motorola assumes that the reader is familiar with the J1850 specification before reading this protocol handler description.*

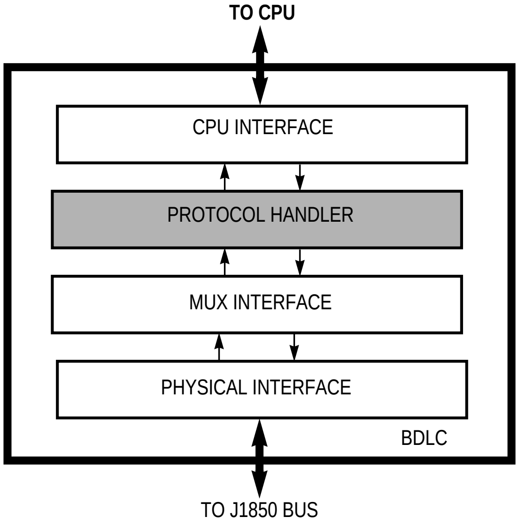


Figure 14-13. BDLC Block Diagram

14.6.1 Protocol Architecture

The protocol handler contains the state machine, Rx shadow register, Tx shadow register, Rx shift register, Tx shift register, and loopback multiplexer as shown in **Figure 14-14**.

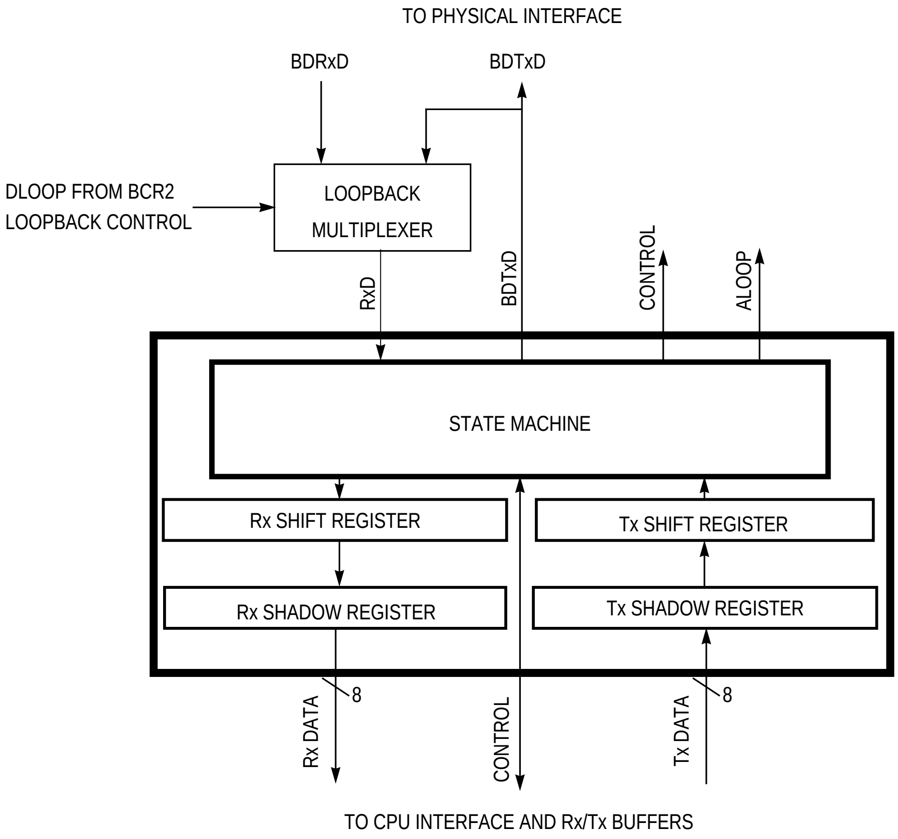


Figure 14-14. BDLC Protocol Handler Outline

14.6.2 Rx and Tx Shift Registers

The Rx shift register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx shadow register. The Tx shift register takes data, in parallel form, from the Tx shadow register and presents it serially to the state machine so that it can be transmitted onto the J1850 bus.

14.6.3 Rx and Tx Shadow Registers

Immediately after the Rx shift register has completed shifting in a byte of data, this data is transferred to the Rx shadow register and RDRF or RXIFR is set (see **14.7.4 BDLC State Vector Register**). An interrupt is generated if the interrupt enable bit (IE) in BCR1 is set. After the transfer takes place, this new data byte in the Rx shadow register is available to the CPU interface, and the Rx shift register is ready to shift in the next byte of data. Data in the Rx shadow register must be retrieved by the CPU before it is overwritten by new data from the Rx shift register.

Once the Tx shift register has completed its shifting operation for the current byte, the data byte in the Tx shadow register is loaded into the Tx shift register. After this transfer takes place, the Tx shadow register is ready to accept new data from the CPU when the TDRE flag in the BSVR is set.

14.6.4 Digital Loopback Multiplexer

The digital loopback multiplexer connects RxD to either BDTxD or BDRxD, depending on the state of the DLOOP bit in the BCR2 (see **14.7.3 BDLC Control Register 2**).

14.6.5 State Machine

All functions associated with performing the protocol are executed or controlled by the state machine. The state machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following sections describe the BDLC’s actions in a variety of situations.

14.6.5.1 4X Mode

The BDLC can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 variable pulse width modulation (VPW) operation. The BDLC cannot transmit in 4X mode, but it can receive messages in 4X mode, if the RX4XE bit is set in BCR2. If the RX4XE bit is not set in the BCR2, any 4X message on the J1850 bus is treated as noise by the BDLC and is ignored.

14.6.5.2 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the BDLC does allow for a special block mode of operation of the receiver. As far as the BDLC is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

14.6.5.3 Transmitting a Message in Block Mode

A block mode message is transmitted inherently by simply loading the bytes one by one into the BDR until the message is complete. The programmer should wait until the TDRE flag (see **14.7.4 BDLC State Vector Register**) is set prior to writing a new byte of data into the BDR. The BDLC does not contain any predefined maximum J1850 message length requirement.

14.6.5.4 J1850 Bus Errors

The BDLC detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

Transmission Error

If the message transmitted by the BDLC contains invalid bits or framing symbols on non-byte boundaries, this constitutes a transmission error. When a transmission error is detected, the BDLC immediately will cease transmitting. The error condition is reflected in the BSVR (see **Table 14-5**). If the interrupt enable bit (IE in BCR1) is set, a CPU interrupt request from the BDLC is generated.

CRC Error

A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed and the CRC calculation result is not equal. The CRC code will detect any single and 2-bit errors, as well as all 8-bit burst errors and almost all other types of errors. The CRC error flag (in BSVR) is set when a CRC error is detected. See **14.7.4 BDLC State Vector Register**.

Symbol Error

A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. The invalid symbol is set when a symbol error is detected. See **14.7.4 BDLC State Vector Register**.

Framing Error

A framing error is detected if an EOD or EOF symbol is detected on a non-byte boundary from the J1850 bus. A framing error also is detected if the BDLC is transmitting the EOD and instead receives an active symbol. The symbol invalid, or the out-of-range flag, is set when a framing error is detected. See **14.7.4 BDLC State Vector Register**.

Bus Fault

If a bus fault occurs, the response of the BDLC will depend upon the type of bus fault.

If the bus is shorted to battery, the BDLC will wait for the bus to fall to a passive state before it will attempt to transmit a message. As long as the short remains, the BDLC will never attempt to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC will see an idle bus, begin to transmit the message, and then detect a transmission error (in BSVR), since the short to ground would not allow the bus to be driven to the active (dominant) SOF state. The BDLC will abort that transmission and wait for the next CPU command to transmit.

In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC will resume normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus. See **14.7.4 BDLC State Vector Register**.

BREAK — Break

If a BREAK symbol is received while the BDLC is transmitting or receiving, an invalid symbol (in BSVR) interrupt will be generated. Reading the BSVR (see **14.7.4 BDLC State Vector Register**) will clear this interrupt condition. The BDLC will wait for the bus to idle, then wait for a start-of-frame (SOF) symbol.

The BDLC cannot transmit a BREAK symbol. It only can receive a BREAK symbol from the J1850 bus.

14.6.5.5 Summary

Table 14-1. BDLC J1850 Bus Error Summary

Error Condition	BDLC Function
Transmission error	For invalid bits or framing symbols on non-byte boundaries, invalid symbol interrupt will be generated. BDLC stops transmission.
Cyclical redundancy check (CRC) error	CRC error interrupt will be generated. The BDLC will wait for EOF.
Invalid symbol: BDLC transmits, but receives invalid bits (noise)	The BDLC will abort transmission immediately. Invalid symbol interrupt will be generated.
Framing error	Invalid symbol interrupt will be generated. The BDLC will wait for end of frame (EOF).
Bus short to V_{DD}	The BDLC will not transmit until the bus is idle. Invalid symbol interrupt will be generated. EOF interrupt also must be seen before another transmission attempt. Depending on length of the short, LOA flag also may be set.
Bus short to GND	Thermal overload will shut down physical interface. Fault condition is seen as invalid symbol flag. EOF interrupt must also be seen before another transmission attempt.
BDLC receives BREAK symbol	Invalid symbol interrupt will be generated. The BDLC will wait for the next valid SOF.

14.7 BDLC CPU Interface

The CPU interface provides the interface between the CPU and the BDLC and consists of five user registers.

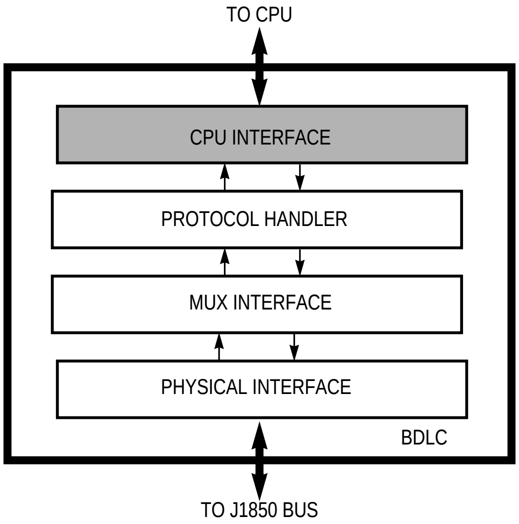


Figure 14-15. BDLC Block Diagram

14.7.1 BDLC Analog and Roundtrip Delay

This register programs the BDLC to compensate for various delays of different external transceivers. The default delay value is 16 μs. Timing adjustments from 9 μs to 24 μs in steps of 1 μs are available. The BARD register can be written only once after each reset, after which they become read-only bits. The register may be read at any time.

Address: $003E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
Write:			▨	▨				
Reset:	1	1	0	0	0	1	1	1

▨ = Unimplemented

Figure 14-16. BDLC Analog and Roundtrip Delay Register (BARD)

ATE — Analog Transceiver Enable Bit

The analog transceiver enable (ATE) bit is used to select either the on-board or an off-chip analog transceiver.

1 = Select on-board analog transceiver
0 = Select off-chip analog transceiver

NOTE: *This device does not contain an on-board transceiver. This bit should be programmed to a logic 0 for proper operation.*

RXPOL — Receive Pin Polarity Bit

The receive pin polarity (RXPOL) bit is used to select the polarity of an incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding it back to the digital receive pin.

1 = Select normal/true polarity; true non-inverted signal from the J1850 bus; for example, the external transceiver does not invert the receive signal
0 = Select inverted polarity, where an external transceiver inverts the receive signal from the J1850 bus

BO3–BO0 — BARD Offset Bits

Table 14-2 shows the expected transceiver delay with respect to BARD offset values.

Table 14-2. BDLC Transceiver Delay

BARD Offset Bits BO[3:0]	Corresponding Expected Transceiver's Delays (μs)
0000	9
0001	10
0010	11
0011	12
0100	13
0101	14
0110	15
0111	16
1000	17
1001	18
1010	19
1011	20
1100	21
1101	22
1110	23
1111	24

14.7.2 BDLC Control Register 1

This register is used to configure and control the BDLC.

Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
Write:					R	R		
Reset:	1	1	1	0	0	0	0	0

R = Reserved

Figure 14-17. BDLC Control Register 1 (BCR1)

IMSG — Ignore Message Bit

This bit is used to disable the receiver until a new start-of-frame (SOF) is detected.

1 = Disable receiver. When set, all BDLC interrupt requests will be masked (except $20 in BSVR) and the status bits will be held in their reset state. If this bit is set while the BDLC is receiving a message, the rest of the incoming message will be ignored.

0 = Enable receiver. This bit is cleared automatically by the reception of an SOF symbol or a BREAK symbol. It will then generate interrupt requests and will allow changes of the status register to occur. However, these interrupts may still be masked by the interrupt enable (IE) bit.

CLKS — Clock Bit

For J1850 bus communications to take place, the nominal BDLC operating frequency (f_{BDLC}) must always be 1.048576 MHz or 1 MHz. The CLKS register bit allows the user to select the frequency (1.048576 MHz or 1 MHz) used to automatically adjust symbol timing.

1 = Binary frequency (1.048576 MHz) selected for f_{BDLC}

0 = Integer frequency (1 MHz) selected for f_{BDLC}

R1 and R0 — Rate Select Bits

These bits determine the amount by which the frequency of the MCU system clock is divided to form the MUX interface clock (f_{BDLC}) which defines the basic timing resolution of the MUX interface. They may be written only once after reset, after which they become read-only bits.

The nominal frequency of f_{BDLC} must always be 1.048576 MHz or 1.0 MHz for J1850 bus communications to take place. Hence, the value programmed into these bits is dependent on the chosen MCU system clock frequency per **Table 14-3**.

Table 14-3. BDLC Rate Selection

f_{XCLK} Frequency	R1	R0	Division	f_{BDLC}
1.049 MHz	0	0	1	1.049 MHz
2.097 MHz	0	1	2	1.049 MHz
4.194 MHz(1)	1	0	4	1.049 MHz
8.389 MHz(1)	1	1	8	1.049 MHz
1.000 MHz	0	0	1	1.00 MHz
2.000 MHz	0	1	2	1.00 MHz
4.000 MHz(1)	1	0	4	1.00 MHz
8.000 MHz(1)	1	1	8	1.00 MHz

1. Invalid option on this MCU

IE— Interrupt Enable Bit

This bit determines whether the BDLC will generate CPU interrupt requests in run mode. It does not affect CPU interrupt requests when exiting the BDLC stop or BDLC wait modes. Interrupt requests will be maintained until all of the interrupt request sources are cleared by performing the specified actions upon the BDLC's registers. Interrupts that were pending at the time that this bit is cleared may be lost.

1 = Enable interrupt requests from BDLC
0 = Disable interrupt requests from BDLC

If the programmer does not wish to use the interrupt capability of the BDLC, the BDLC state vector register (BSVR) can be polled periodically by the programmer to determine BDLC states. See **14.7.4 BDLC State Vector Register** for a description of the BSVR.

WCM — Wait Clock Mode Bit

This bit determines the operation of the BDLC during CPU wait mode. See **14.8.2 Stop Mode** and **14.8.1 Wait Mode** for more details on its use.

1 = Stop BDLC internal clocks during CPU wait mode
0 = Run BDLC internal clocks during CPU wait mode

14.7.3 BDLC Control Register 2

This register controls transmitter operations of the BDLC. It is recommended that BSET and BCLR instructions be used to manipulate data in this register to ensure that the register's content does not change inadvertently.

Address: $003B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
Reset:	1	1	0	0	0	0	0	0

Figure 14-18. BDLC Control Register 2 (BCR2)

ALOOP — Analog Loopback Mode Bit

This bit determines whether the J1850 bus will be driven by the analog physical interface's final drive stage. The programmer can use this bit to reset the BDLC state machine to a known state after the off-chip analog transceiver is placed in loopback mode. When the user clears ALOOP, to indicate that the off-chip analog transceiver is no longer in loopback mode, the BDLC waits for an EOF symbol before attempting to transmit. Most transceivers have the ALOOP feature available.

1 = Input to the analog physical interface's final drive stage is looped back to the BDLC receiver. The J1850 bus is not driven.
0 = The J1850 bus will be driven by the BDLC. After the bit is cleared, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol time (t_{TRV4}) before message reception or

a minimum of inter-frame symbol time (t_{TRV6}) before message transmission. See **17.12 BDLC Transmitter VPW Symbol Timings (BARD) Bits BO[3:0] = 0111**.

DLOOP — Digital Loopback Mode Bit

This bit determines the source to which the digital receive input (BDRxD) is connected and can be used to isolate bus fault conditions (see **Figure 14-14**). If a fault condition has been detected on the bus, this control bit allows the programmer to connect the digital transmit output to the digital receive input. In this configuration, data sent from the transmit buffer will be reflected back into the receive buffer. If no faults exist in the BDLC, the fault is in the physical interface block or elsewhere on the J1850 bus.

1 = When set, BDRxD is connected to BDTxD. The BDLC is now in digital loopback mode.

0 = When cleared, BDTxD is not connected to BDRxD. The BDLC is taken out of digital loopback mode and can now drive or receive the J1850 bus normally (given ALOOP is not set). After writing DLOOP to 0, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol (t_{tv4}) time before allowing a reception of a message. The BDLC requires the bus to be idle for a minimum of inter-frame separator symbol (t_{tv6}) time before allowing any message to be transmitted.

RX4XE — Receive 4X Enable Bit

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbps) or receive only at 41.6 kbps. This feature is useful for fast downloading of data into a J1850 node for diagnostic or factory programming of the node.

1 = When set, the BDLC is put in 4X receive-only operation.

0 = When cleared, the BDLC transmits and receives at 10.4 kbps. Reception of a BREAK symbol automatically clears this bit and sets BDLC state vector register (BSVR) to $001C.

NBFS — Normalization Bit Format Select Bit

This bit controls the format of the normalization bit (NB). (See **Figure 14-19**.) SAE J1850 strongly encourages using an active long (logic 0) for in-frame responses containing cyclical redundancy check (CRC) and an active short (logic 1) for in-frame responses without CRC.

1 = NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) does not end with a CRC byte.

0 = NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) does not end with a CRC byte.

TEOD — Transmit End-of-Data Bit

This bit is set by the programmer to indicate the end of a message is being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte. This bit also is used to end an in-frame response (IFR). If the transmit shadow register is full when TEOD is set, the CRC byte will be transmitted after the current byte in the Tx shift register and the byte in the Tx shadow register have been transmitted. (See **14.6.3 Rx and Tx Shadow Registers** for a description of the transmit shadow register.) Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC state vector register (BSVR) is cleared to allow lower priority interrupts to occur. See **14.7.4 BDLC State Vector Register**.

1 = Transmit end-of-data (EOD) symbol

0 = The TEOD bit will be cleared automatically at the rising edge of the first CRC bit that is sent or if an error is detected. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol or an error condition occurs.

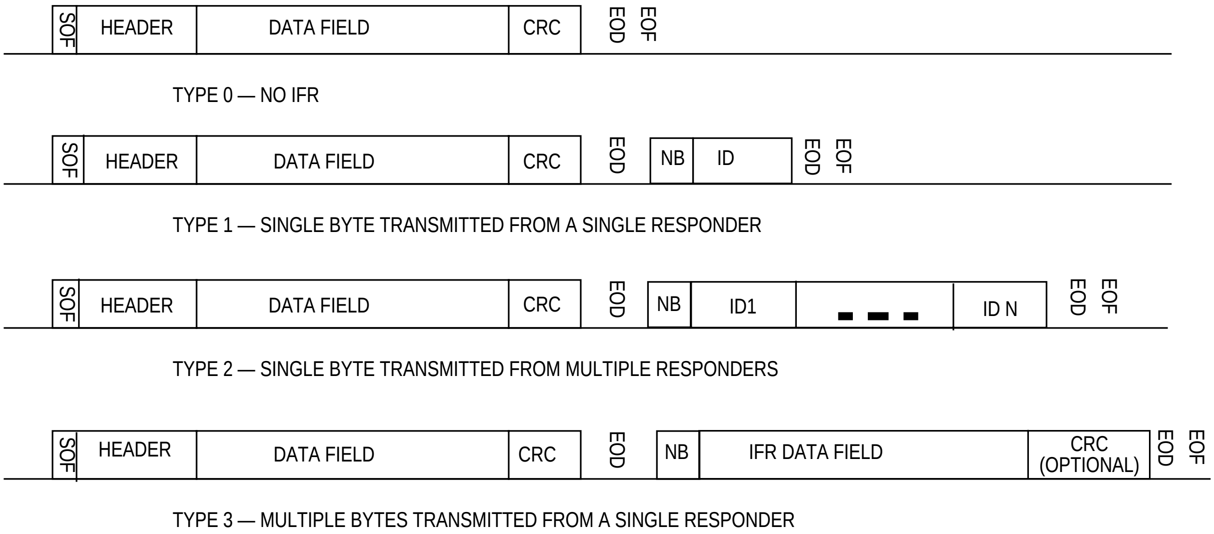
TSIFR, TMIFR1, and TMIFR0 — Transmit In-Frame Response Control Bits

These three bits control the type of in-frame response being sent. The programmer should not set more than one of these control bits to a 1 at any given time. However, if more than one of these three control bits are set to 1, the priority encoding logic will force these register bits to a known value as shown in **Table 14-4**. For example, if 011 is written to TSIFR, TMIFR1, and TMIFR0, then internally they will be encoded as 010. However, when these bits are read back, they will read 011.

Table 14-4. BDLC Transmit In-Frame Response Control Bit Priority Encoding

Write/Read TSIFR	Write/Read TMIFR1	Write/Read TMIFR0	Actual TSIFR	Actual TMIFR1	Actual TMIFR0
0	0	0	0	0	0
1	X	X	1	0	0
0	1	X	0	1	0
0	0	1	0	0	1

The BDLC supports the in-frame response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown in **Figure 14-19**. The purpose of the in-frame response modes is to allow multiple nodes to acknowledge receipt of the data by responding with their personal ID or physical address in a concatenated manner after they have seen the EOD symbol. If transmission arbitration is lost by a node while sending its response, it continues to transmit its ID/address until observing its unique byte in the response stream. For VPW modulation, the first bit of the IFR is always passive; therefore, an active normalization bit must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync their responses.



TYPE 0 — NO IFR

TYPE 1 — SINGLE BYTE TRANSMITTED FROM A SINGLE RESPONDER

TYPE 2 — SINGLE BYTE TRANSMITTED FROM MULTIPLE RESPONDERS

TYPE 3 — MULTIPLE BYTES TRANSMITTED FROM A SINGLE RESPONDER

NB = Normalization bit
ID = Identifier, usually the physical address of the responder(s)

Figure 14-19. Types of In-Frame Response (IFR)

TSIFR — Transmit Single Byte IFR with No CRC (Type 1 or 2) Bit

The TSIFR bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node. See **Figure 14-19**.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by the byte in the BDR.

0 = The TSIFR bit will be cleared automatically, once the BDLC has successfully transmitted the byte in the BDR onto the bus, or TEOD is set, or an error is detected on the bus.

If the programmer attempts to set the TSIFR bit immediately after the EOD symbol has been received from the bus, the TSIFR bit will remain in the reset state and no attempt will be made to transmit the IFR byte.

If a loss of arbitration occurs when the BDLC attempts to transmit and after the IFR byte winning arbitration completes transmission, the BDLC will again attempt to transmit the BDR (with no normalization bit). The BDLC will continue transmission attempts until an error is detected on the bus, or TEOD is set, or the BDLC transmission is successful.

If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits **will not** be sent out because the BDLC will attempt to retransmit the byte in the transmit shift register after the IRF byte winning arbitration completes transmission.

TMIFR1 — Transmit Multiple Byte IFR with CRC (Type 3) Bit

The TMIFR1 bit requests the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are still subject to J1850 message length maximums (see **14.5.2 J1850 Frame Format**). See **Figure 14-19**.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

0 = The TMIFR1 bit will be cleared automatically, once the BDLC has successfully transmitted the CRC byte and EOD symbol, by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR1 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see **14.7.4 BDLC State Vector Register**) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BDLC control register 2 (BCR2). This will instruct the BDLC to transmit a CRC byte once the byte in the BDR is transmitted, and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if the programmer wishes to transmit a single byte followed by a CRC byte, the programmer should load the byte into the BDR before the EOD symbol has been received, and then set the TMIFR1 bit. Once the TDRE interrupt occurs, the programmer should then set the TEOD bit in the BCR2. This will result in the byte in the BDR being the only byte transmitted before the IFR CRC byte, and no TDRE interrupt will be generated.

If the programmer attempts to set the TMIFR1 bit immediately after the EOD symbol has been received from the bus, the TMIFR1 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting any byte of a multiple byte IFR, the BDLC will go to the loss of arbitration state, set the appropriate flag, and cease transmission.

If the BDLC loses arbitration during the IFR, the TMIFR1 bit will be cleared and **no attempt** will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits will be sent out.

NOTE: *The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.*

TMIFR0 — Transmit Multiple Byte IFR without CRC (Type 3) Bit

The TMIFR0 bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums (see **14.5.2 J1850 Frame Format**). See **Figure 14-19**.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the BDR.

0 = The TMIFR0 bit will be cleared automatically, once the BDLC has successfully transmitted the EOD symbol, by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR0 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see **14.7.4 BDLC State Vector Register**) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BCR2. This will instruct the BDLC to transmit an EOD symbol once the byte in the BDR is transmitted, indicating the end of the IFR portion of the message frame. The BDLC will not append a CRC when the TMIFR0 is set.

If the programmer attempts to set the TMIFR0 bit after the EOD symbol has been received from the bus, the TMIFR0 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting, the TMIFR0 bit will be cleared, and **no attempt** will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits (active short bits) will be sent out.

NOTE: *The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on to the J1850 bus from a corrupted message.*

14.7.4 BDLC State Vector Register

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a multiplex protocol. It provides an index offset that is directly related to the BDLC's current state, which can be used with a user-supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Address: $003C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	I3	I2	I1	I0	0	0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-20. BDLC State Vector Register (BSVR)

I0, I1, I2, and I3 — Interrupt Source Bits

These bits indicate the source of the interrupt request that currently is pending. The encoding of these bits is listed in **Table 14-5**.

Table 14-5. BDLC Interrupt Sources

BSVR	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No interrupts pending	0 (Lowest)
$04	0	0	0	1	Received EOF	1
$08	0	0	1	0	Received IFR byte (RXIFR)	2
$0C	0	0	1	1	BDLC Rx data register full (RDRF)	3
$10	0	1	0	0	BDLC Tx data register empty (TDRE)	4
$14	0	1	0	1	Loss of arbitration	5
$18	0	1	1	0	Cyclical redundancy check (CRC) error	6
$1C	0	1	1	1	Symbol invalid or out of range	7
$20	1	0	0	0	Wakeup	8 (Highest)

Bits I0, I1, I2, and I3 are cleared by a read of the BSVR except when the BDLC data register needs servicing (RDRF, RXIFR, or TDRE conditions). RXIFR and RDRF can be cleared only by a read of the BSVR followed by a read of the BDLC data register (BDR). TDRE can either be cleared by a read of the BSVR followed by a write to the BDLC BDR or by setting the TEOD bit in BCR2.

Upon receiving a BDLC interrupt, the user can read the value within the BSVR, transferring it to the CPU’s index register. The value can then be used to index into a jump table, with entries four bytes apart, to quickly enter the appropriate service routine. For example:

```
Service     LDX   BSVR         Fetch State Vector Number
            JMP   JMPTAB,X     Enter service routine,
*                              (must end in RTI)
*
JMPTAB      JMP   SERVE0       Service condition #0
            NOP
            JMP   SERVE1       Service condition #1
            NOP
            JMP   SERVE2       Service condition #2
            NOP
*
            JMP   SERVE8       Service condition #8
            END
```

NOTE: *The NOPs are used only to align the JMPs onto 4-byte boundaries so that the value in the BSVR can be used intact. Each of the service routines must end with an RTI instruction to guarantee correct continued operation of the device. Note also that the first entry can be omitted since it corresponds to no interrupt occurring.*

The service routines should clear all of the sources that are causing the pending interrupts. Note that the clearing of a high priority interrupt may still leave a lower priority interrupt pending, in which case bits I0, I1, and I2 of the BSVR will then reflect the source of the remaining interrupt request.

If fewer states are used or if a different software approach is taken, the jump table can be made smaller or omitted altogether.

14.7.5 BDLC Data Register

Address: $003D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
Reset:	Indeterminate after reset							

Figure 14-21. BDLC Data Register (BDR)

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC. It is also used to pass data received from the J1850 bus to the CPU. Each data byte (after the first one) should be written only after a Tx data register empty (TDRE) state is indicated in the BSVR.

Data read from this register will be the last data byte received from the J1850 bus. This received data should only be read after an Rx data register full (RDRF) interrupt has occurred. See **14.7.4 BDLC State Vector Register**.

The BDR is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next data byte. The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag is set to indicate that a new byte of data has been received. The programmer has one BDLC byte reception time to read the shadow register and clear the RDRF flag before the shadow register is overwritten by the newly received byte.

To abort an in-progress transmission, the programmer should stop loading data into the BDR. This will cause a transmitter underrun error and the BDLC automatically will disable the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte plus two extra logic 1s have been transmitted. The receiver will pick this up as an error and relay it in the state vector register as an invalid symbol error.

NOTE: *The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.*

14.8 Low-Power Modes

The following information concerns wait mode and stop mode.

14.8.1 Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and the WCM bit in BDLC control register 1 (BCR1) is previously clear. In BDLC wait mode, the BDLC cannot drive any data.

A subsequent successfully received message, including one that is in progress at the time that this mode is entered, will cause the BDLC to wake up and generate a CPU interrupt request if the interrupt enable (IE) bit in the BDLC control register 1 (BCR1) is previously set (see **14.7.2 BDLC Control Register 1** for a better understanding of IE). This results in less of a power savings, but the BDLC is guaranteed to receive correctly the message which woke it up, since the BDLC internal operating clocks are kept running.

NOTE: *Ensuring that all transmissions are complete or aborted before putting the BDLC into wait mode is important.*

14.8.2 Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BDLC control register 1 (BCR1) is previously set. This is the lowest power mode that the BDLC can enter.

A subsequent passive-to-active transition on the J1850 bus will cause the BDLC to wake up and generate a non-maskable CPU interrupt request. When a STOP instruction is used to put the BDLC in stop mode, the BDLC is not guaranteed to correctly receive the message which woke it up, since it may take some time for the BDLC internal operating clocks to restart and stabilize. If a WAIT instruction is used to put the BDLC in stop mode, the BDLC is guaranteed to correctly receive the byte which woke it up, if and only if an end-of-frame (EOF) has been detected prior to issuing the WAIT instruction by the CPU. Otherwise, the BDLC will not correctly receive the byte that woke it up.

If this mode is entered while the BDLC is receiving a message, the first subsequent received edge will cause the BDLC to wake up immediately, generate a CPU interrupt request, and wait for the BDLC internal operating clocks to restart and stabilize before normal communications can resume. Therefore, the BDLC is not guaranteed to receive that message correctly.

NOTE: *Ensuring that all transmissions are complete or aborted prior to putting the BDLC into stop mode is important.*

Section 15. Gauge Drivers

15.1 Contents

15.2 Introduction

The MC68HC705V12 contains on-chip circuitry to drive six cross coil air core gauges. Four of the gauge drivers are 3-pin drivers intended for 180° gauges (minor gauges) and two of the drivers are full 4-pin H-bridge drivers for 360° gauges (major gauges). The output drivers for both major and minor gauges operate in a current drive mode. That is, the current in the gauge coils is controlled rather than the voltage across the coil. The maximum amount of current that can be driven into any coil is set by the value of the resistance between the I_{Max} and V_{SSA} pins. The current driven into each coil is set by writing a hex value to the current magnitude registers and the direction of current is selected by setting or clearing the appropriate bits in the current direction registers. The ratio of the current used to set the gauge deflection angle is software configured. No particular drive technique is implemented in hardware.

15.3 Gauge System Overview

The circuitry contained within the MC68HC705V12 provides a great deal of flexibility for driving the coils. The user specifies coil currents rather than degrees of deflection. This allows the software to drive the coil currents in a variety of ways. The user must specify the magnitude of the current as well as the direction it should flow for full H-bridge drivers. Half H-bridge drivers require specification of a magnitude only. Eight full H-bridge drivers and four half H-bridge drivers support two 360° and four 180° gauges.

Figure 15-1 is a block diagram of the gauge driver module within the MC68HC705V12. Each of the blocks requiring more description is described in the following subsections.

There are 20 coil driver pins on the MC68HC705V12. These are grouped into two types. The pins whose names start with MAJ are full H-bridge coil drivers. A or B in the pin name indicates major gauge A or B. A 1 or 2 in the name refers to coil 1 or coil 2 within the same gauge. It is important to keep coils within the same gauge connected to the same A or B coil driver pins. The + or – in the pin name indicates the direction of current flow according to this convention: The current direction positive current means current flow is out of the pin with the + in its name and into the pin with – in its name. Negative current means current is flowing into the pin with the + in its name and out of the pin with – in its name.

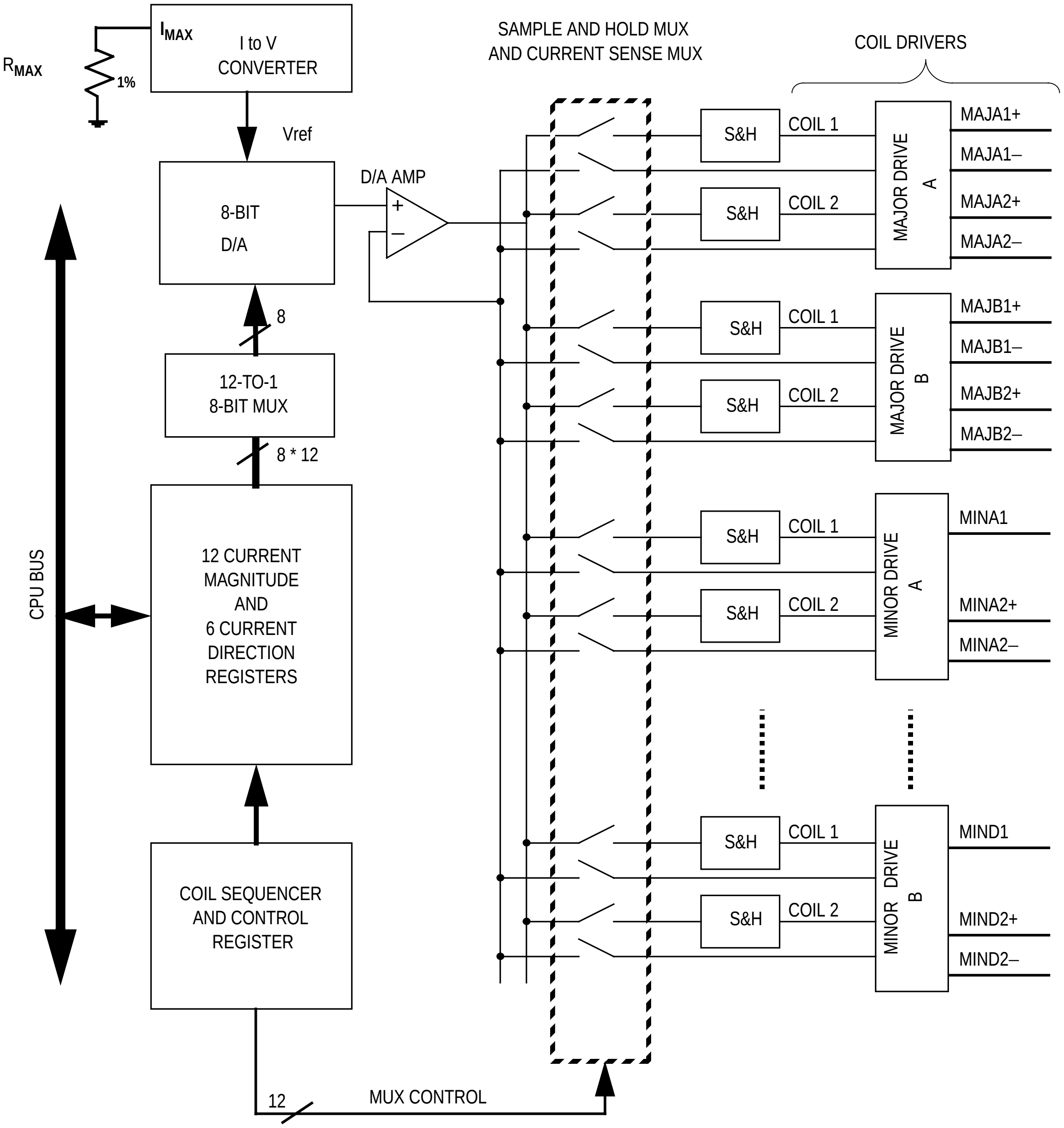


Figure 15-1. Gauge Driver Block Diagram

15.4 Coil Drivers

To support both 180° and 360° gauges and to keep the pin count as low as possible, it is necessary to use two different types of coil drivers:

- Full H-bridge drivers
- Half H-bridge drivers

Major gauges will require two of the full H-bridge drivers and the minor gauges will require one full H-bridge driver and one half H-bridge driver. A full H-bridge driver uses two pins and is capable of driving a controlled current in either direction in a single coil. A half H-bridge driver uses only one pin and can sink a controlled amount of current in one direction only. The amount of current flowing through the coils and its direction in the case of the full H-bridge driver are controlled through the current magnitude registers (CMR) and the current direction registers (CDR) described here.

All of the components shown in **Figure 15-2** and **Figure 15-3** are internal components except for the gauge coils. The resistive and inductive properties of the external coils are expected to fall within the ranges of R_{Coil} and L_{Coil} shown in **Section 17. Electrical Specifications**. The resistance is important for calculating minimum operating voltages and power dissipation (see **15.12 External Component Considerations**), and the inductance is important in determining settling time (a part of t_{GCS}) and controlling the rate of change of the current driven in the coils.

For consistency, note that the dot on the coil is always connected to the + pin in the coil driver, or, in the case of the half H-bridge driver, it is connected to the positive supply pin.

The internal resistor R_I is used to measure how much current is flowing in the coil. The op-amp shown in this diagram is actually built only once and is shared among all 12 coil drivers through a multiplexer to reduce manufacturing variability among drivers. To determine what voltage must come from the digital-to-analog (D/A) output, the maximum current level set by the external R_{MAX} resistor is converted to a reference voltage input to the D/A. This reference voltage sets the maximum output voltage of the D/A (with an input of $FF).

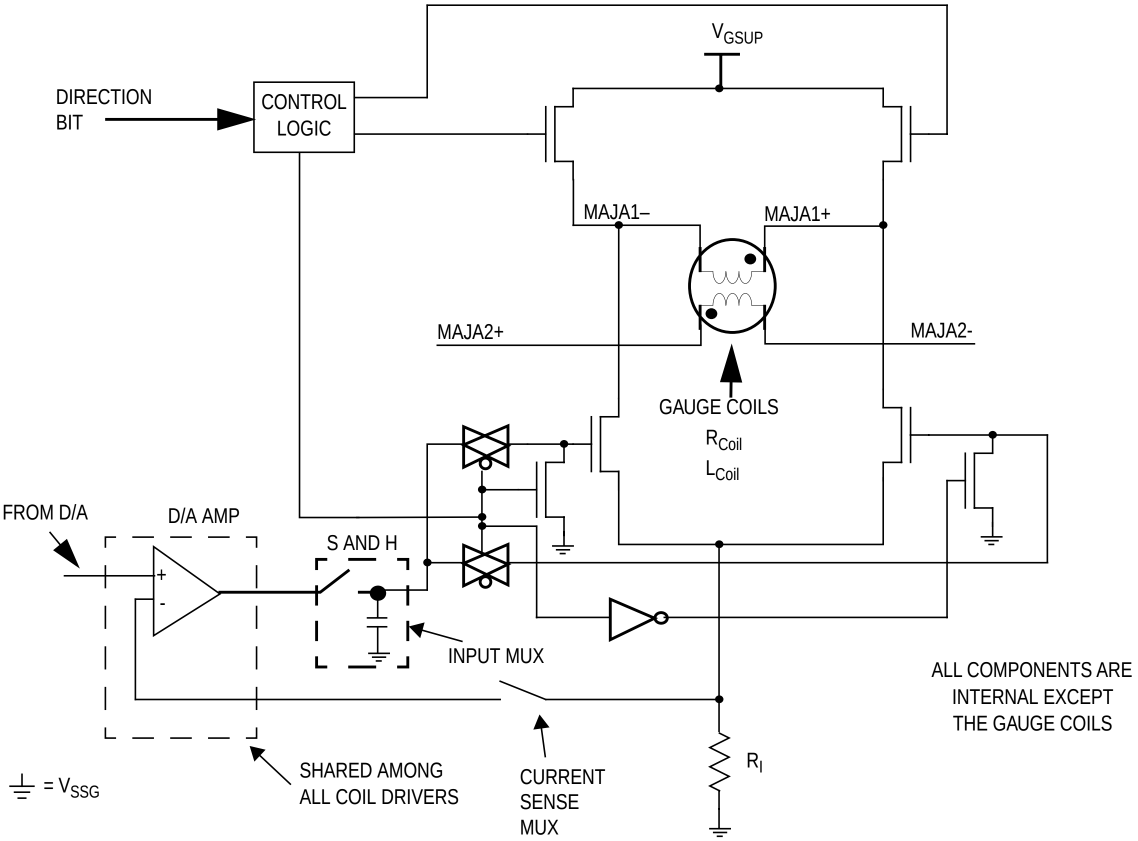


Figure 15-2. Full H-Bridge Coil Driver

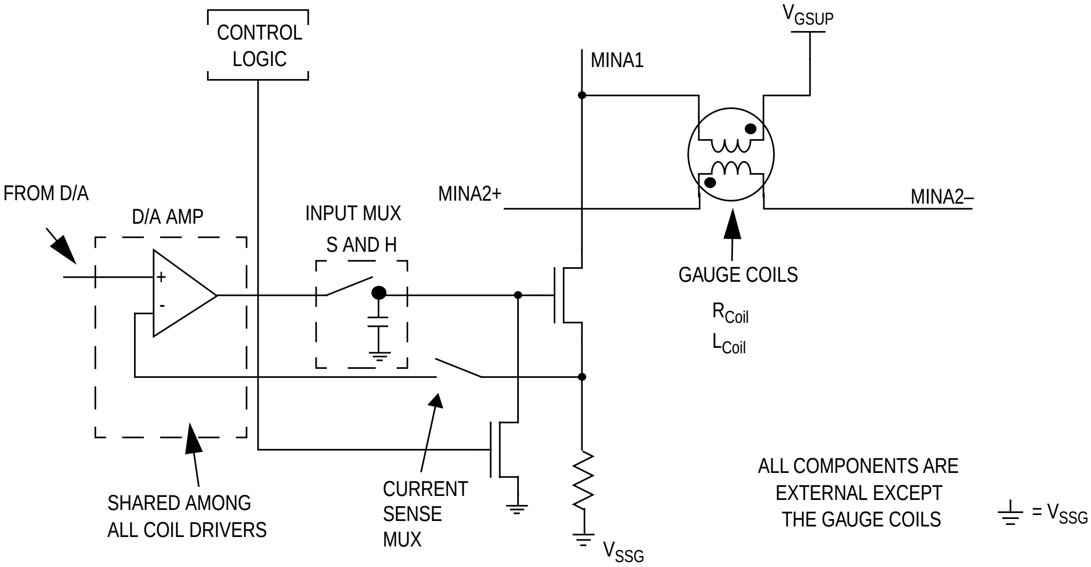
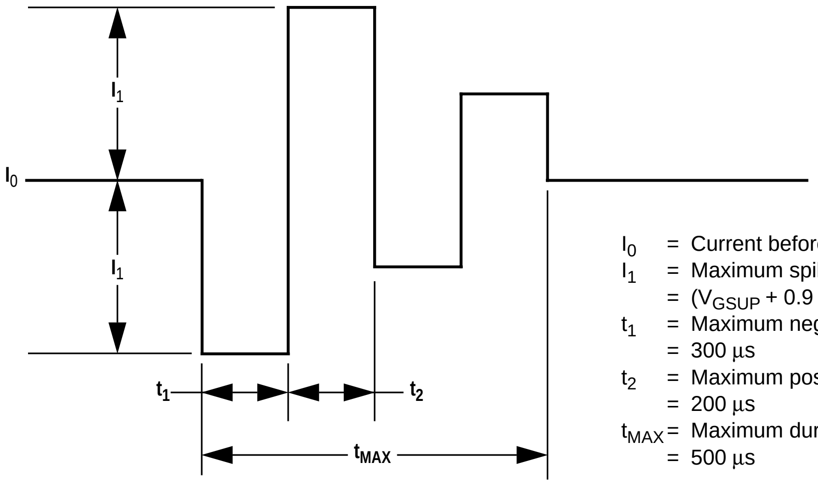


Figure 15-3. Half H-Bridge Coil Driver

15.5 Technical Note

An auto-zeroing scheme is implemented in the MC68HC705V12 to reduce errors internal to the chip. Prior to each coil update, during the auto-zero phase, the amplifier output is disconnected from all FET gates (see **Figure 15-2**) while the input is connected to its new input voltage. On completion of the auto-zero cycle, the amplifier output is connected to the appropriate FET gate. Since the FET gate and amplifier output are typically not at the same potential, the FET gate is momentarily pulled down, until the amplifier control loop re-establishes the correct gate potential. This abrupt change in gate potential results in voltage spikes, and thus the current spikes in the gauge coil. The magnitude, duration, and number of spikes are dependent on the coil resistance, gauge supply voltage (V_{GSUP}), and the coil current prior to the auto-zero cycle (**Figure 15-4**). Only the worst case spike durations, under the specified test conditions, are shown. Typically, the spike duration and total spike duration is smaller.

NOTE: *Due to the positive and negative spikes, there is negligible d.c. error introduced.*



I_0 = Current before spikes
I_1 = Maximum spike magnitude
= $(V_{GSUP} + 0.9\ V)/R_{Coil}$
t_1 = Maximum negative spike duration
= 300 µs
t_2 = Maximum positive spike duration
= 200 µs
t_{MAX} = Maximum duration of all spikes
= 500 µs

Conditions: V_{GSUP} = 8.00 V
L_{Coil} = 30 mH
R_{Coil} = 200 W (typical)
T_A = 27°C

Figure 15-4. Specification for Current Spikes

15.6 Gauge Driver Control Registers

The gauge driver module requires the use of four types of control registers:

- The gauge enable register enables or disables individual gauges.
- The current magnitude registers set the amount of current to flow in a particular coil.
- The current direction register determines which direction the current will flow in a coil.
- The scan control register controls how the 12 coil drivers will be sequenced and updated by the analog multiplexers and control logic.

Each register is described in more detail in the following sections.

15.6.1 Gauge Enable Register

All bits in this register are used to select which of the six gauges will be driven when the gauge module is active. If any bit of bits 7-2 is set, the gauge module will become active.

When all bits are cleared, the gauge module is considered off. As much circuitry as possible is shut off to conserve power. The D/A, the coil sequencing logic, and the coil current measurement circuits are turned off. All high-side drivers in the H-bridge drivers are left on to absorb any transient current that may be generated when the drivers are initially turned on or off; all low-side drivers are high-Z.

The effects of these bits on the scanning sequence of the gauges are described in **15.7 Coil Sequencer and Control**.

Address: $0020

	BIt 7	6	5	4	3	2	1	Bit 0
Read: Write:	MJAON	MJBON	MIAON	MIBON	MICON	MIDON	CMPS	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 15-5. Gauge Enable Register (GER)

MJAON — Major Gauge A On Bit

This bit controls whether major gauge A is on or off.
1 = Gauge is on.
0 = Gauge is off.

MJBON — Major Gauge B On Bit

This bit controls whether major gauge B is on or off.
1 = Gauge is on.
0 = Gauge is off.

MIAON — Minor Gauge A On Bit

This bit controls whether minor gauge A is on or off.
1 = Gauge is on.
0 = Gauge is off.

MIBON — Minor Gauge B On Bit

This bit controls whether minor gauge B is on or off.
1 = Gauge is on.
0 = Gauge is off.

MICON — Minor Gauge C On Bit

This bit controls whether minor gauge C is on or off.
1 = Gauge is on.
0 = Gauge is off.

MIDON — Minor Gauge D On Bit

This bit controls whether minor gauge D is on or off.
1 = Gauge is on.
0 = Gauge is off.

CMPS — Feedback Compensation Select

This bit is provided to enable the user to select between one of two gauge driver feedback paths, depending upon the characteristics of the load.

1 = Alternate feedback circuit
0 = Default feedback circuit

15.6.2 Current Magnitude Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0022	MAJA1 Magnitude Register (MAJA1)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0023	MAJA2 Magnitude Register (MAJA2)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0024	MAJB1 Magnitude Register (MAJB1)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0025	MAJB2 Magnitude Register (MAJB2)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0026	MINA1 Magnitude Register (MINA1)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0027	MINA2 Magnitude Register (MINA2)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0028	MINB1 Magnitude Register (MINB1)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0

Figure 15-6. Current Magnitude Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0029	MINB2 Magnitude Register (MINB2)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002A	MINC1 Magnitude Register (MINC1)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002B	MINC2 Magnitude Register (MINC2)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002C	MIND1 Magnitude Register (MIND1)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002D	MIND2 Magnitude Register (MIND2)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Figure 15-6. Current Magnitude Registers (Continued)

The naming convention used in the CMRs in **Figure 15-6** indicates whether it is a major or minor gauge driver, which major or minor gauge (A, B, C, etc.), and which coil within the gauge is affected (coil 1 or coil 2).

Each of the magnitude registers is double buffered to keep both coil currents within the same gauge as closely coupled as possible. Transfer of data from the master to the slave buffers in these registers is under control of the coil sequencer and control logic and is described in **15.7 Coil Sequencer and Control**. A read of any of the CMRs will return only the contents of the slave buffer. If a read of one of the CMRs takes place after a write of data but before the master-to-slave transfer takes place, the data read may be different from the data written. The master register will always hold the contents of the last write. Reset clears all bits.

The 8-bit value written to these registers will determine the amount of current that will flow in each of the 12 coils. For example, MAJA1 controls the magnitude of the current between the MAJA1+ and MAJA1– pins.

The theoretical current that will flow between the + and – pins is given by this equation:

$$I = \left((ICM)X\left(\frac{\text{Reg value}}{255}\right)\right)$$

I_{CM} is the maximum current that can be driven into any of the coil drivers and is given by this equation:

$I_{CM} = (I_{MAX} \text{ x } 10) \text{ x } (1 + E_{CA} + E_{MAX})$

E_{CA} is the total internal error in generating I_{CM} from I_{MAX} and is shown in **17.11 Gauge Driver Electricals**.

The E_{MAX} is the error tolerance of the R_{MAX} resistor and is shown in **17.11 Gauge Driver Electricals**.

The “reg value” is the base 10 representation of the value written to the magnitude registers.

I_{MAX} is set by the external resistor and is a reference current that is used to generate the coil currents.

I_{MAX} is related to the external R_{MAX} resistor by the equation:

$$I_{MAX} = \left(\frac{2.5}{R_{MAX}}\right) \text{ x } 4$$

15.6.3 Current Direction Registers

The bits in these registers control the direction of current flow in each of the full eight H-bridge drive outputs. Note that only coil 2 in the minor gauges requires a direction bit. Since coil 1 in each of the minor gauges is a half H-bridge driver, it only requires a current magnitude register.

The CDR also contains a master and slave latch. A read of any of these registers will return the value in the slave buffer.

15.6.3.1 Current Direction Register for Major A

Address: $002E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	R	R	R	R	R	0	DMJA1	DMJA2
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 15-7. MAJA Current Direction Register (DMAJA)

DMJA1 and DMJA2 — Current Direction Bits for Major Gauge A

1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.

0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.6.3.2 Current Direction Register for Major B

Address: $002E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	DMJB1	DMJB2
Reset:	0	0	0	0	0	0	0	0

Figure 15-8. MAJB Current Direction Register (DMAJB)

DMJB1 and DMJB2 — Current Direction Bits for Major Gauge B

1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.

0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.6.3.3 Current Direction Register for Minor A

Address: $0030

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

Figure 15-9. MINA Current Direction Register (DMINA)

DMIA — Current Direction Bit for Minor Gauge A

1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.
0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.6.3.4 Current Direction Register for Minor B

Address: $0031

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

Figure 15-10. MINB Current Direction Register (DMINB)

DMIB — Current Direction Bit for Minor Gauge B
1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.
0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.6.3.5 Current Direction Register for Minor C

Address: $0032

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

Figure 15-11. MINC Current Direction Register (DMINC)

DMIC — Current Direction Bit for Minor Gauge C

1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.

0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.6.3.6 Current Direction Register for Minor D

Address: $0033

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

Figure 15-12. MIND Current Direction Register (DMIND)

DMID — Current Direction Bit for Minor Gauge D

1 = Current flow will be from the – pin to the + pin on the corresponding coil driver.

0 = Current flow will be from the + pin to the – pin on the corresponding coil driver.

15.7 Coil Sequencer and Control

As shown in **Figure 15-1**, the digital/analog converter is shared among all 12 coils. The sequence in which the coils are scanned and the events that take place during the scanning process are described in this section. The scan control and status register in **15.7.2 Scan Status and Control Register** controls how the coil sequencer will operate. This register contains control bits that affect how the gauge sequencer will scan through the six gauges as well as a status bit to indicate where the scanning sequencer is in the scanning operation.

15.7.1 Scanning Sequence Description

The coil sequencer can be operated in two basic modes: automatic or manual. In either mode, each coil is updated by the D/A, muxes, and sample and hold circuits in the sequence shown in **Figure 15-1**. One time through the coil sequence is referred to as a scan cycle. It takes a time, $t_{GCS,}$ to update each coil during the scanning sequence. Since there are 12 coils in the six gauge drivers, it will take a time, 12 * $t_{GCS,}$ to complete one scan cycle.

The differences between the automatic and manual modes are discussed in the following subsections.

15.7.1.1 Automatic Mode

Once all of the coils have been updated, the sequence repeats automatically. The transfer of data in the CMR and CDRs master-to-slave buffers is performed at the beginning of each gauge update time.

When one of the coil registers associated with a gauge is written, the second register also must be written before either value will be used. The coil registers may be written in either order.

For example, if the MIND1 register is written with any value, then the MIND2 register must also be written. Otherwise, minor gauge D will not be updated on subsequent scans and the currents driven into the coils will maintain their previously programmed values. This sequence must be followed even if the data written to one magnitude register is not different from the data already in the register. The hardware works off the write operation to the registers, not off the data written.

Before the master-to-slave transfer takes place in the CDRs, each coil in a particular gauge must be updated. Because writes to the CMRs are the only requirement for transferring master to slave of the CDRs, the CDRs should be written before the CMRs are written.

15.7.1.2 Manual Mode

The user must set the SCNS bit in the scan status and control register (SSCR) to initiate a scan cycle. Once a single scan cycle takes place, the coil sequencer stops and waits for the SCNS bit to be set again before starting another scan cycle. The SCNS bit must be set at a fast enough rate (the scan period) to prevent the sample and hold circuits from drooping and introducing error and current fluctuations into the output currents. This minimum time is called the minimum scan period, t_{MSN} (see **17.11 Gauge Driver Electricals**). The transfer of data from the CMR and CDRs master-to-slave buffers is performed at the beginning of each gauge update time even if all CMRs and CDRs were not updated.

If any of the gauges are turned off by clearing the appropriate bits in the gauge enable register (GER), the time the coil sequencer would have spent updating the coils in the disabled gauge is still expended, but the coil driver remains off. This provides for a consistent scan rate regardless of the number of gauges that are enabled.

The scanning sequence for the coils is shown in **Table 15-1**. It takes a time, t_{GCS}, to update each coil. This includes time to move the data from the CMR and CDR (automatic mode), perform the digital/analog conversion, update the sample and hold circuit at the coil driver, and wait for all transient currents to settle for each coil.

Table 15-1. Coil Scanning Sequencer

Coil Number	Coil Name	Gauge Name
1	MAJA1	Major A
2	MAJA2	Major A
3	MAJB1	Major B
4	MAJB2	Major B
5	MINA1	Minor A
6	MINA2	Minor A
7	MINB1	Minor B
8	MINB2	Minor B
9	MINC1	Minor C
10	MINC2	Minor C
11	MIND1	Minor D
12	MIND2	Minor D

Because several CPU write operations may be necessary to write to the CMRs and CDRs, all of the CMRs and the CDRs contain a master and a slave buffer to help prevent unwanted fluctuations in coil currents between the writes to the three registers on a given gauge. Only the slave buffers will affect the coil currents and direction.

For coil currents to remain as consistent as possible during the scanning and updating of the gauge coil currents, the sample and hold update operation must take place in a particular way. The control logic will perform this function. When the scanning control logic is ready to advance to the next coil, this operations sequence must take place:

1. Open all sample and hold switches.
2. Increment pointer to next CMR slave register. If both CMRs for this gauge have been written, transfer new master data to slave buffer for this CMR and corresponding CDR. If both CMRs have not been written, don’t transfer data from master to slave.
3. Move slave buffer data to the D/A input.
4. Wait for the D/A output to muxes.
5. Close sample and hold mux and update direction control from CDR.

6. Wait for sample and hold to settle.
7. Go back to step 1.

15.7.2 Scan Status and Control Register

Although the CDR and CMRs can be written at any time, the user may want to write the CDR and CMRs at a particular time in the scanning sequence. Some of the bits in the SSCR give the user the information needed to synchronize the writes to the CDR and CMRs with the coil sequencer.

In addition to the sychronization bits, this register also contains a bit that affects the type of scanning that will take place (automatic or manual) and a bit to initiate a scan cycle manually when using manual mode.

Address: $0021

	BIt 7	6	5	4	3	2	1	Bit 0
Read:	SYNIE	SYNF	0	R	GCS1	GCS0	SCNS	AUTOS
Write:		▨	SYNR					
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 15-13. Scan Status and Control Register (SSCR)

SYNIE — Synchronize Interrupt Enable Bit

When this bit is set, an interrupt signal will be sent to the CPU when the SYNF bit is set. The I bit in the CPU condition code register must be cleared in order for the interrupt to be recognized by the CPU. The interrupt vector assigned to the gauge module is shown in **Table 15-2**.

1 = Interrupt is enabled.
0 = Interrupt is disabled.

SYNF — Synchronize Flag Bit

This bit is a read-only status bit and indicates that the coil sequencer has begun to service coil 11 (minor D). At this point in the scanning cycle, it is safe to write any of the CMRs or CDRs without affecting the current scan cycle. Any time this bit is set and the SYNIE bit is set, a CPU interrupt will be generated. The bit will be set even if minor D is not enabled in the GER, since the scanning sequence time is not affected by the enabling or disabling of the gauges. This bit will function in either auto or manual mode and does not affect the scanning operation in any way. It serves only as a status flag. Note that once this bit is set, the software will have a time, 2 * $t_{GCS,}$ to update the CDR and CMR register if new data is to be used in the next scan cycle.

The bit is cleared by writing a 1 to the SYNR bit and by reset.

SYNR — Synchronize Flag Reset Bit

This bit is used to clear the SYNF bit. Writing a 1 to this bit will clear the SYNF bit if the SYNF bit was set during a read of the SSCR. This bit will always read 0.

GCS1–GCS0 — Gauge Clock Select Bits

These bits determine the clock divide ratio for the clock used by the scan sequencer. This provides for the use of several different system clock rates while still providing the gauge driver module with the same scanning rate.

Table 15-2. Gauge Module Clock Select Bits

CPU Bus Clock Frequency	GCS1	GCS0	Division	Scan Cycle Time
f_{op} = 0.5 MHz	0	0	512	t_{GCS}
f_{op} = 1.0 MHz	0	1	1024	t_{GCS}
f_{op} = 2.0 MHz	1	0	2048	t_{GCS}
f_{op} = 4.0 MHz[1]	1	1	4096	t_{GCS}

[1] Must not be selected

SCNS — Scan Start Bit

When the coil sequencer is being operated in manual mode, this bit is used to initiate a scan cycle. Setting this bit starts the scan cycle. All CMRs and CDRs will transfer data from the master to the slave when this bit is set.

This bit clears automatically once the scan cycle begins to service coil 11 (minor D). The bit will clear at the proper time, even if the minor D gauge is not enabled in the GER, since the scan cycle time is not affected by the enabling or disabling of the gauges. The bit is cleared once coil 11 begins to be serviced because adequate time (2 * t_{GCS}) for the software (either interrupt driven or polled) to recognize the flag and write new data to the CDR and CMR registers for the next scan cycle should be provided. Note that after the scan cycle has finished, a new scan cycle will not begin until this bit is set again. If a 1 is written to this bit before it clears, the write will be ignored. In automatic mode, this bit has no effect.

AUTOS — Automatic Mode Select Bit

This bit selects whether the coil sequencer will operate in manual or automatic mode.

1 = Automatic mode
0 = Manual mode

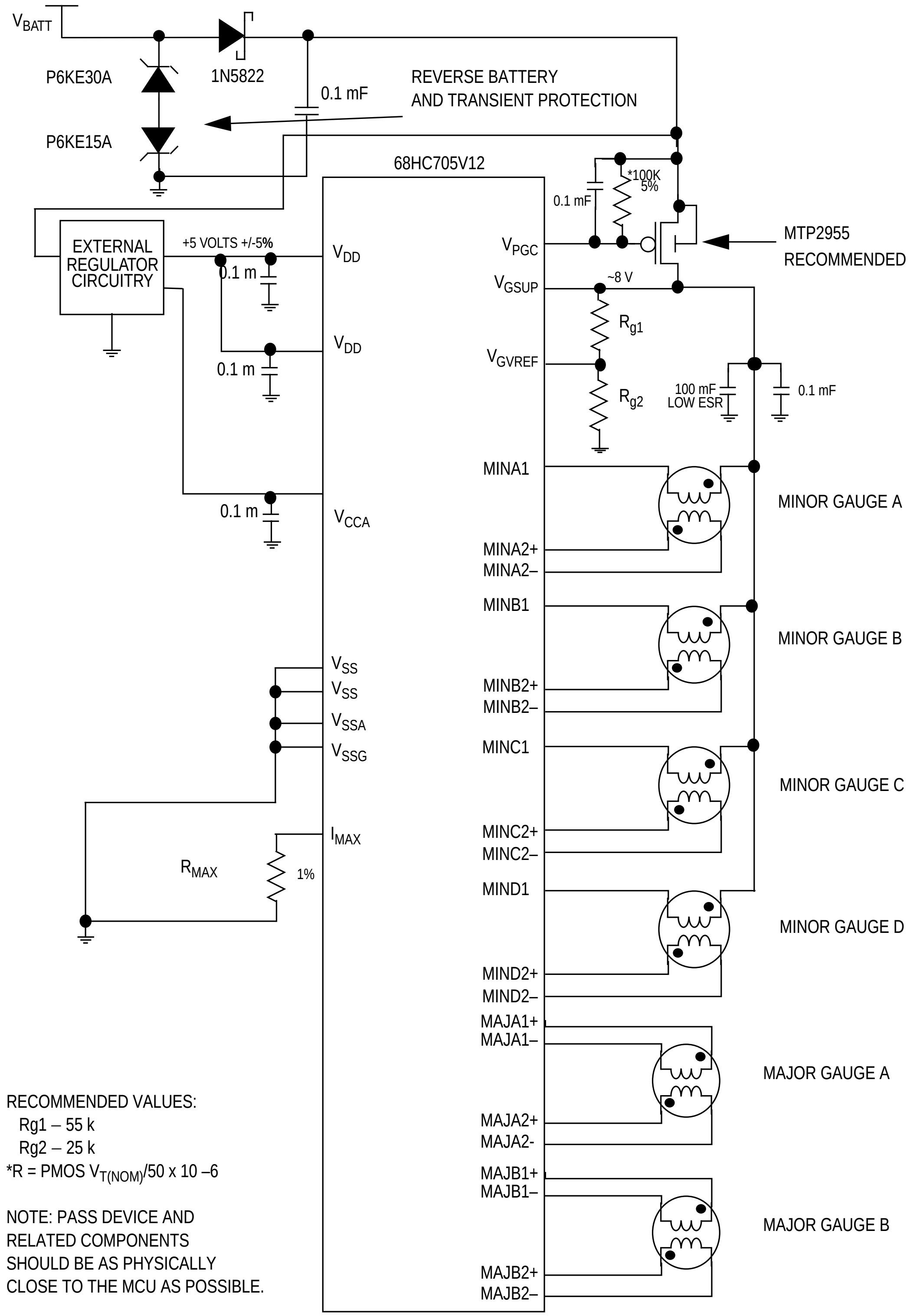
15.8 Mechanism Diagram

The diagram in **Figure 15-14** shows one way the gauge coils could be connected to the coil driver pins and how some of the other pins should be connected. The external components that have actual part numbers are merely examples of suitable components. Other components with similar operating characteristics also may be used.

15.9 Gauge Power Supply

The MC68HC705V12 contains most of the circuitry to provide the coil drivers with a regulated supply that is necessary to drive the coil. Referring to **Figure 15-3**, the gauge drive voltage, V_{GSUP}, is derived with the aid of an external P-channel enhancement mode MOSFET device which serves as the series pass devices between a +12 V supply and the V_{GSUP} pin. Two external resistors also are used to set the level of $V_{GSUP.}$ The drive to the gate of the external pass devices will be whatever is required to produce a V_{GVREF} voltage of 2.5. The value of resistors R_{G1} and R_{G2} should be chosen so that

$$V_{GSUP} * [R_{G2}/(R_{G1}+R_{G2})] = 2.5$$



RECOMMENDED VALUES:
Rg1 – 55 k
Rg2 – 25 k
*R = PMOS $V_{T(NOM)}$/50 x 10 –6

NOTE: PASS DEVICE AND
RELATED COMPONENTS
SHOULD BE AS PHYSICALLY
CLOSE TO THE MCU AS POSSIBLE.

Figure 15-14. Sample Gauge Connections to the MC68HC705V12

NOTE: *The V_{GSUP} pin requires a 100 µF low ESR capacitor for regulator stability. In addition, the V_{DD} and V_{CCA} pins should have the usual 0.1 µF bypass capacitors to V_{SS} and V_{SSA} respectively.*

To provide effective decoupling and to reduce radiated RF emissions, the small decoupling capacitors must be located as close to the supply pins as possible. The self-inductance of these capacitors and the parasitic inductance and capacitance of the interconnecting traces determine the self-resonant frequency of the decoupling network. Too low a frequency will reduce decoupling effectiveness and could increase radiated RF emissions from the system. A low-value capacitor (470 pF to 0.01 µF) placed in parallel with the other capacitors will improve the bandwidth and effectiveness of the network.

15.10 Gauge Regulator Accuracy

The on-chip portion of the regulator will contribute no more than E_{GS} percent to the variation in the V_{GSUP} voltage. The remaining errors will come from the tolerances in the R_{G1} and R_{G2} resistors off-chip.

15.11 Coil Current Accuracy

The accuracy of the current flowing between the + and – coil pins of a particular coil driver pin pair is described here.

Matching of currents between coils within the same gauge is specified in **17.11 Gauge Driver Electricals** as E_{CM}.

The absolute accuracy of the coil current that can be driven into any coil is determined by the accuracy of I_{CM} given by the equations in **15.10 Gauge Regulator Accuracy** and will be a total of (E_{CA} + E_{MAX}).

Because the D/A amp is shared among all coil drivers and between both sets of drivers in the full H-bridge drivers, there will be no difference in the magnitude of the current when the magnitude register value remains constant and only the polarity bit is changed in a coil driver.

15.12 External Component Considerations

To determine the values and tolerances of the external components required to drive the air core gauge coils, the minimum V_{BATT} voltage, at the V12 pin, and the power dissipation should be considered.

Figure 15-15 shows the components in the path between the +12 V coming in through the external devices the internal devices and into V_{SSG}.

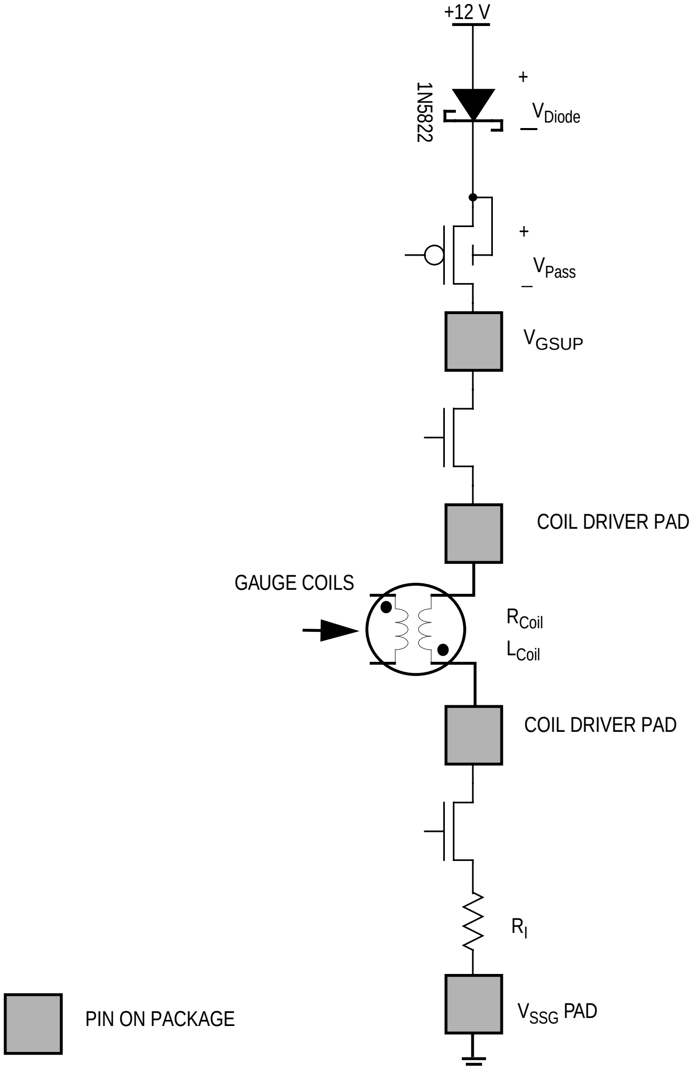


Figure 15-15. Coil Driver Current Path

15.12.1 Minimum Voltage Operation

To maintain accuracy to as low a V_{BATT} voltage as possible, the following equations should be used to calculate the range of values for the external components.

$V_{BATT(min)} = V_{GSUP(max)} + V_{DIODE} + V_{PASS}$

V_{PASS} is the drop across the external P-channel MOSFET at SF * 12 * $I_{Coil(max)}$

SF = % of max current driven by all coil drivers in application. Worst case 0.707 (45^o) assumes sin/cos drive algorithm.

V_{Diode} = Drop across reverse battery protection diode at 12 * $I_{Coil(max)}$

To solve the equation, the factors involved in generating the gauge supply voltage, V_{GSUP}, must first be calculated due to both internal tolerances and the tolerances of external resistors R_{G1} and R_{G2},

$V_{GSUP} = V_{GSUP(nom)} \text{ x } (1 \pm T_{OL})$

$V_{GSUP(nom)}$ is the V_{GSUP} voltage generated with all tolerances set to 0%.

$T_{OL} = E_{GS} + T_{OL(RG1)} + T_{OL(RG2)}$ and includes temperature effects. R_{G1} and R_{G2} are the external resistors used to set the V_{GSUP} voltage. The internal tolerances are E_{GS}.

The minimum V_{GSUP} voltage required for proper operation is given by

$I_{Coil(max)} \text{ x } [R_{Coil(max)} + R_{SI(max)}]$

Where

- R_{SI} is the total of the internal resistances from the transistors and sense resistor and is found in the electrical specifications.
- I_{Coil} is the minimum required coil current.
- R_{Coil} is the minimum coil resistance including temperature effects.

The minimum required V_{GSUP} must agree with the minimum generated V_{GSUP} of

$$V_{GSUP(min)} = V_{GSUP(nom)} \text{ x } (1–T_{OL})$$

equating the two

$$V_{GSUP(nom)} \text{ x } (1–T_{OL}) = I_{Coil(max)} \text{ x } [\ R_{Coil(max)} + R_{SI(max)}\ ]$$

$$V_{GSUP(nom)} = \frac{I_{Coil(max)} \text{ x } [R_{Coil(max)} + R_{SI(max)}]}{(1 – T_{OL})}$$

15.12.2 Power Dissipation

To keep the junction temperature to a minimum, the power consumed by the gauge drivers must be factored into the chip power dissipation equation. The total chip power dissipation combined with the thermal resistance of the package cannot exceed the maximum junction temperature, T_J. The total chip power dissipation is given by this equation:

$$P_D = P_{Gauge} + P_{Chip}$$

P_{CHIP} is the power contribution by all chip modules that are connected to the V_{DD} and V_{DDA} sources including part of the gauge module. To calculate P_{Chip}, use this equation:

$$P_{Chip} = (I_{DD} \text{ x } V_{DD}) + (I_{CCA} \text{ x } V_{CCA})$$

The power dissipation contributed by the gauge module is given by this equation:

$$P_{Gauge} = [V_{GSUP(max)} \text{ X } I_{GSUP}] + P_{GDrivers}$$

Where

$$P_{GDrivers} =$$
$$[(V_{GSUP(max)} \text{ x } I_{Coil(max)}) – (I_{Coil(max)}{}^2 \text{ x } R_{Coil(min)})] \text{ x } 12 \text{ x SF}$$

Where

- I_{GSUP} = the current consumed by the gauge module from the V_{GSUP} pin for functions other than generating coil currents

- 12 is the number of coil drivers
- SF = % of max current driven in any coil; worse case for power dissipation purposes, 0.707 (45º) assumes SIN/COS drive algorithm
- $I_{COIL(max)}$ = maximum required coil current in each coil

15.12.3 Coil Inductance Limits

Since the MCU pins will drive the gauge coils directly without any external voltage limiting devices, precautions must be taken to avoid generating voltages and currents high enough to damage the MCU. The high voltages generated by the inductive impedance of the coil will be related directly to the coil drivers. This imposes a limit on the maximum coil inductance referred to as L_{Coil} in the electrical specifications.

15.13 Operation in Wait Mode

During wait mode, the gauge driver module will continue to operate normally. The gauges will continue to be driven to the currents and directions that were last written to the CMR and CDR. In manual mode, if the CPU will be put into wait mode between scan cycles, the SYNIE bit in the SSCR should be set to enable the gauge module to generate an interrupt request (which will take the CPU out of wait mode) to properly service the gauge coils.

15.14 Operation in Stop Mode

During stop mode, the system clocks will stop operating. All bits in the GER register will be cleared automatically when stop mode is entered. No other bits in any other gauge module registers will be affected. The gauge controller sequence and control logic will be reset/initialized such that a new scan sequence will begin once the gauges are turned on during the user’s stop mode recovery sequence.

Section 16. Instruction Set

16.1 Contents

16.2 Introduction

The MCU instruction set has 62 instructions and uses eight addressing modes. The instructions include all those of the M146805 CMOS Family plus one more: the unsigned multiply (MUL) instruction. The MUL instruction allows unsigned multiplication of the contents of the accumulator (A) and the index register (X). The high-order product is stored in the index register, and the low-order product is stored in the accumulator.

16.3 Addressing Modes

The CPU uses eight addressing modes for flexibility in accessing data. The addressing modes provide eight different ways for the CPU to find the data required to execute an instruction.

The eight addressing modes are:

- Inherent
- Immediate
- Direct
- Extended
- Indexed, no offset
- Indexed, 8-bit offset
- Indexed, 16-bit offset
- Relative

16.3.1 Inherent

Inherent instructions are those that have no operand, such as return from interrupt (RTI) and stop (STOP). Some of the inherent instructions act on data in the CPU registers, such as set carry flag (SEC) and increment accumulator (INCA). Inherent instructions require no operand address and are one byte long.

16.3.2 Immediate

Immediate instructions are those that contain a value to be used in an operation with the value in the accumulator or index register. Immediate instructions require no operand address and are two bytes long. The opcode is the first byte, and the immediate data value is the second byte.

16.3.3 Direct

Direct instructions can access any of the first 256 memory locations with two bytes. The first byte is the opcode, and the second is the low byte of the operand address. In direct addressing, the CPU automatically uses $00 as the high byte of the operand address.

16.3.4 Extended

Extended instructions use three bytes and can access any address in memory. The first byte is the opcode; the second and third bytes are the high and low bytes of the operand address.

When using the Motorola assembler, the programmer does not need to specify whether an instruction is direct or extended. The assembler automatically selects the shortest form of the instruction.

16.3.5 Indexed, No Offset

Indexed instructions with no offset are 1-byte instructions that can access data with variable addresses within the first 256 memory locations. The index register contains the low byte of the effective address of the operand. The CPU automatically uses $00 as the high byte, so these instructions can address locations $0000–$00FF.

Indexed, no offset instructions are often used to move a pointer through a table or to hold the address of a frequently used RAM or I/O location.

16.3.6 Indexed, 8-Bit Offset

Indexed, 8-bit offset instructions are 2-byte instructions that can access data with variable addresses within the first 511 memory locations. The CPU adds the unsigned byte in the index register to the unsigned byte following the opcode. The sum is the effective address of the operand. These instructions can access locations $0000–$01FE.

Indexed 8-bit offset instructions are useful for selecting the kth element in an n-element table. The table can begin anywhere within the first 256 memory locations and could extend as far as location 510 ($01FE). The k value is typically in the index register, and the address of the beginning of the table is in the byte following the opcode.

16.3.7 Indexed,16-Bit Offset

Indexed, 16-bit offset instructions are 3-byte instructions that can access data with variable addresses at any location in memory. The CPU adds the unsigned byte in the index register to the two unsigned bytes following the opcode. The sum is the effective address of the operand. The first byte after the opcode is the high byte of the 16-bit offset; the second byte is the low byte of the offset.

Indexed, 16-bit offset instructions are useful for selecting the kth element in an n-element table anywhere in memory.

As with direct and extended addressing, the Motorola assembler determines the shortest form of indexed addressing.

16.3.8 Relative

Relative addressing is only for branch instructions. If the branch condition is true, the CPU finds the effective branch destination by adding the signed byte following the opcode to the contents of the program counter. If the branch condition is not true, the CPU goes to the next instruction. The offset is a signed, two’s complement byte that gives a branching range of –128 to +127 bytes from the address of the next location after the branch instruction.

When using the Motorola assembler, the programmer does not need to calculate the offset, because the assembler determines the proper offset and verifies that it is within the span of the branch.

16.4 Instruction Types

The MCU instructions fall into five categories:

- Register/memory instructions
- Read-modify-write instructions
- Jump/branch instructions
- Bit manipulation instructions
- Control instructions

16.4.1 Register/Memory Instructions

These instructions operate on CPU registers and memory locations. Most of them use two operands. One operand is in either the accumulator or the index register. The CPU finds the other operand in memory.

Table 16-1. Register/Memory Instructions

Instruction	Mnemonic
Add memory byte and carry bit to accumulator	ADC
Add memory byte to accumulator	ADD
AND memory byte with accumulator	AND
Bit test accumulator	BIT
Compare accumulator	CMP
Compare index register with memory byte	CPX
Exclusive OR accumulator with memory byte	EOR
Load accumulator with memory byte	LDA
Load Index register with memory byte	LDX
Multiply	MUL
OR accumulator with memory byte	ORA
Subtract memory byte and carry bit from accumulator	SBC
Store accumulator in memory	STA
Store index register in memory	STX
Subtract memory byte from accumulator	SUB

16.4.2 Read-Modify-Write Instructions

These instructions read a memory location or a register, modify its contents, and write the modified value back to the memory location or to the register.

NOTE: *Do not use read-modify-write operations on write-only registers.*

Table 16-2. Read-Modify-Write Instructions

Instruction	Mnemonic
Arithmetic shift left (same as LSL)	ASL
Arithmetic shift right	ASR
Bit clear	BCLR[1]
Bit set	BSET[1]
Clear register	CLR
Complement (one’s complement)	COM
Decrement	DEC
Increment	INC
Logical shift left (same as ASL)	LSL
Logical shift right	LSR
Negate (two’s complement)	NEG
Rotate left through carry bit	ROL
Rotate right through carry bit	ROR
Test for negative or zero	TST[2]

1. Unlike other read-modify-write instructions, BCLR and BSET use only direct addressing.
2. TST is an exception to the read-modify-write sequence because it does not write a replacement value.

16.4.3 Jump/Branch Instructions

Jump instructions allow the CPU to interrupt the normal sequence of the program counter. The unconditional jump instruction (JMP) and the jump-to-subroutine instruction (JSR) have no register operand. Branch instructions allow the CPU to interrupt the normal sequence of the program counter when a test condition is met. If the test condition is not met, the branch is not performed.

The BRCLR and BRSET instructions cause a branch based on the state of any readable bit in the first 256 memory locations. These 3-byte instructions use a combination of direct addressing and relative addressing. The direct address of the byte to be tested is in the byte following the opcode. The third byte is the signed offset byte. The CPU finds the effective branch destination by adding the third byte to the program counter if the specified bit tests true. The bit to be tested and its condition (set or clear) is part of the opcode. The span of branching is from –128 to +127 from the address of the next location after the branch instruction. The CPU also transfers the tested bit to the carry/borrow bit of the condition code register.

Table 16-3. Jump and Branch Instructions

Instruction	Mnemonic
Branch if carry bit clear	BCC
Branch if carry bit set	BCS
Branch if equal	BEQ
Branch if half-carry bit clear	BHCC
Branch if half-carry bit set	BHCS
Branch if higher	BHI
Branch if higher or same	BHS
Branch if $\overline{\text{IRQ}}$ pin high	BIH
Branch if $\overline{\text{IRQ}}$ pin low	BIL
Branch if lower	BLO
Branch if lower or same	BLS
Branch if interrupt mask clear	BMC
Branch if minus	BMI
Branch if interrupt mask set	BMS
Branch if not equal	BNE
Branch if plus	BPL
Branch always	BRA
Branch if bit clear	BRCLR
Branch never	BRN
Branch if bit set	BRSET
Branch to subroutine	BSR
Unconditional jump	JMP
Jump to subroutine	JSR

16.4.4 Bit Manipulation Instructions

The CPU can set or clear any writable bit in the first 256 bytes of memory, which includes I/O registers and on-chip RAM locations. The CPU can also test and branch based on the state of any bit in any of the first 256 memory locations.

Table 16-4. Bit Manipulation Instructions

Instruction	Mnemonic
Bit clear	BCLR
Branch if bit clear	BRCLR
Branch if bit set	BRSET
Bit set	BSET

16.4.5 Control Instructions

These instructions act on CPU registers and control CPU operation during program execution.

Table 16-5. Control Instructions

Instruction	Mnemonic
Clear carry bit	CLC
Clear interrupt mask	CLI
No operation	NOP
Reset stack pointer	RSP
Return from interrupt	RTI
Return from subroutine	RTS
Set carry bit	SEC
Set interrupt mask	SEI
Stop oscillator and enable $\overline{\text{IRQ}}$ pin	STOP
Software interrupt	SWI
Transfer accumulator to index register	TAX
Transfer index register to accumulator	TXA
Stop CPU clock and enable interrupts	WAIT

16.5 Instruction Set Summary

Table 16-6. Instruction Set Summary (Sheet 1 of 6)

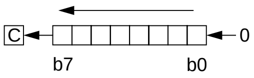
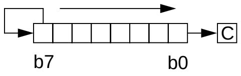
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X	Add with Carry	A ← (A) + (M) + (C)	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A9 B9 C9 D9 E9 F9	ii dd hh ll ee ff ff	2 3 4 5 4 3
ADD #*opr* ADD *opr* ADD *opr* ADD *opr*,X ADD *opr*,X ADD ,X	Add without Carry	A ← (A) + (M)	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	AB BB CB DB EB FB	ii dd hh ll ee ff ff	2 3 4 5 4 3
AND #*opr* AND *opr* AN*D opr* AND *opr*,X AND *opr*,X AND ,X	Logical AND	A ← (A) ∧ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A4 B4 C4 D4 E4 F4	ii dd hh ll ee ff ff	2 3 4 5 4 3
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X	Arithmetic Shift Left (Same as LSL)		—	—	↕	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff 	5 3 3 6 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR ,X	Arithmetic Shift Right		—	—	↕	↕	↕	DIR INH INH IX1 IX	37 47 57 67 77	dd ff 	5 3 3 6 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + *rel* ? C = 0	—	—	—	—	—	REL	24	rr	3
BCLR *n opr*	Clear Bit n	Mn ← 0	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? C = 1	—	—	—	—	—	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? Z = 1	—	—	—	—	—	REL	27	rr	3
BHCC *rel*	Branch if Half-Carry Bit Clear	PC ← (PC) + 2 + *rel* ? H = 0	—	—	—	—	—	REL	28	rr	3
BHCS *rel*	Branch if Half-Carry Bit Set	PC ← (PC) + 2 + *rel* ? H = 1	—	—	—	—	—	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? C ∨ Z = 0	—	—	—	—	—	REL	22	rr	3
BHS *rel*	Branch if Higher or Same	PC ← (PC) + 2 + *rel* ? C = 0	—	—	—	—	—	REL	24	rr	3

Table 16-6. Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
BIH *rel*	Branch if IRQ Pin High	PC ← (PC) + 2 + *rel* ? IRQ = 1	—	—	—	—	—	REL	2F	rr	3
BIL *rel*	Branch if IRQ Pin Low	PC ← (PC) + 2 + *rel* ? IRQ = 0	—	—	—	—	—	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X	Bit Test Accumulator with Memory Byte	(A) ∧ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A5 B5 C5 D5 E5 F5	ii dd hh ll ee ff ff	2 3 4 5 4 3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? C = 1	—	—	—	—	—	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? C ∨ Z = 1	—	—	—	—	—	REL	23	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? I = 0	—	—	—	—	—	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? N = 1	—	—	—	—	—	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? I = 1	—	—	—	—	—	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? Z = 0	—	—	—	—	—	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? N = 0	—	—	—	—	—	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel* ? 1 = 1	—	—	—	—	—	REL	20	rr	3
BRCLR *n opr rel*	Branch if Bit n Clear	PC ← (PC) + 2 + *rel* ? Mn = 0	—	—	—	—	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2 + *rel* ? 1 = 0	—	—	—	—	—	REL	21	rr	3
BRSET *n opr rel*	Branch if Bit n Set	PC ← (PC) + 2 + *rel* ? Mn = 1	—	—	—	—	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n opr*	Set Bit n	Mn ← 1	—	—	—	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	—	—	—	—	—	REL	AD	rr	6
CLC	Clear Carry Bit	C ← 0	—	—	—	—	0	INH	98		2
CLI	Clear Interrupt Mask	I ← 0	—	0	—	—	—	INH	9A		2

Table 16-6. Instruction Set Summary (Sheet 3 of 6)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
CLR *opr* CLRA CLRX CLR *opr*,X CLR ,X	Clear Byte	M ← $00 A ← $00 X ← $00 M ← $00 M ← $00	—	—	0	1	—	DIR INH INH IX1 IX	3F 4F 5F 6F 7F	dd ff 	5 3 3 6 5
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X	Compare Accumulator with Memory Byte	(A) – (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A1 B1 C1 D1 E1 F1	ii dd hh ll ee ff ff 	2 3 4 5 4 3
COM *opr* COMA COMX COM *opr*,X COM ,X	Complement Byte (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (A) X ← ($\overline{X}$) = $FF – (X) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	—	—	↕	↕	1	DIR INH INH IX1 IX	33 43 53 63 73	dd ff 	5 3 3 6 5
CPX #*opr* CPX *opr* CPX *opr* CPX *opr*,X CPX *opr*,X CPX ,X	Compare Index Register with Memory Byte	(X) – (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A3 B3 C3 D3 E3 F3	ii dd hh ll ee ff ff 	2 3 4 5 4 3
DEC *opr* DECA DECX DEC *opr*,X DEC ,X	Decrement Byte	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3A 4A 5A 6A 7A	dd ff 	5 3 3 6 5
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X	EXCLUSIVE OR Accumulator with Memory Byte	A ← (A) ⊕ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A8 B8 C8 D8 E8 F8	ii dd hh ll ee ff ff 	2 3 4 5 4 3
INC *opr* INCA INCX INC *opr*,X INC ,X	Increment Byte	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1	—	—	↕	↕	—	DIR INH INH IX1 IX	3C 4C 5C 6C 7C	dd ff 	5 3 3 6 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Unconditional Jump	PC ← Jump Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2

Table 16-6. Instruction Set Summary (Sheet 4 of 6)

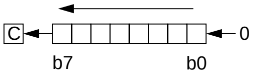
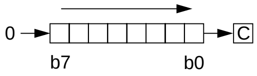
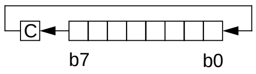
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Effective Address	—	—	—	—	—	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 7 6 5
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X	Load Accumulator with Memory Byte	A ← (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	A6 B6 C6 D6 E6 F6	ii dd hh ll ee ff ff	2 3 4 5 4 3
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X	Load Index Register with Memory Byte	X ← (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	AE BE CE DE EE FE	ii dd hh ll ee ff ff	2 3 4 5 4 3
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X	Logical Shift Left (Same as ASL)		—	—	↕	↕	↕	DIR INH INH IX1 IX	38 48 58 68 78	dd ff	5 3 3 6 5
LSR *opr* LSRA LSRX LSR *opr*,X LSR ,X	Logical Shift Right		—	—	0	↕	↕	DIR INH INH IX1 IX	34 44 54 64 74	dd ff	5 3 3 6 5
MUL	Unsigned Multiply	X : A ← (X) × (A)	0	—	—	—	0	INH	42		1 1
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X	Negate Byte (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	—	—	↕	↕	↕	DIR INH INH IX1 IX	30 40 50 60 70	dd ff	5 3 3 6 5
NOP	No Operation		—	—	—	—	—	INH	9D		2
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X	Logical OR Accumulator with Memory	A ← (A) ∨ (M)	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX	AA BA CA DA EA FA	ii dd hh ll ee ff ff	2 3 4 5 4 3
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X	Rotate Byte Left through Carry Bit		—	—	↕	↕	↕	DIR INH INH IX1 IX	39 49 59 69 79	dd ff	5 3 3 6 5

Table 16-6. Instruction Set Summary (Sheet 5 of 6)

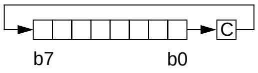
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
ROR *opr* RORA RORX ROR *opr*,X ROR ,X	Rotate Byte Right through Carry Bit		—	—	↕	↕	↕	DIR INH INH IX1 IX	36 46 56 66 76	dd ff 	5 3 3 6 5
RSP	Reset Stack Pointer	SP ← $00FF	—	—	—	—	—	INH	9C		2
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	INH	80		9
RTS	Return from Subroutine	SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	—	—	—	—	—	INH	81		6
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X	Subtract Memory Byte and Carry Bit from Accumulator	A ← (A) – (M) – (C)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A2 B2 C2 D2 E2 F2	ii dd hh ll ee ff ff	2 3 4 5 4 3
SEC	Set Carry Bit	C ← 1	—	—	—	—	1	INH	99		2
SEI	Set Interrupt Mask	I ← 1	—	1	—	—	—	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X	Store Accumulator in Memory	M ← (A)	—	—	↕	↕	—	DIR EXT IX2 IX1 IX	B7 C7 D7 E7 F7	dd hh ll ee ff ff	4 5 6 5 4
STOP	Stop Oscillator and Enable IRQ Pin		—	0	—	—	—	INH	8E		2
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X	Store Index Register In Memory	M ← (X)	—	—	↕	↕	—	DIR EXT IX2 IX1 IX	BF CF DF EF FF	dd hh ll ee ff ff	4 5 6 5 4
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X	Subtract Memory Byte from Accumulator	A ← (A) – (M)	—	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX	A0 B0 C0 D0 E0 F0	ii dd hh ll ee ff ff	2 3 4 5 4 3
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	—	1	—	—	—	INH	83		10
TAX	Transfer Accumulator to Index Register	X ← (A)	—	—	—	—	—	INH	97		2

Table 16-6. Instruction Set Summary (Sheet 6 of 6)

Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			H	I	N	Z	C				
TST *opr* TSTA TSTX TST *opr*,X TST ,X	Test Memory Byte for Negative or Zero	(M) – $00	—	—	↕	↕	—	DIR INH INH IX1 IX	3D 4D 5D 6D 7D	dd ff 	4 3 3 5 4
TXA	Transfer Index Register to Accumulator	A ← (X)	—	—	—	—	—	INH	9F		2
WAIT	Stop CPU Clock and Enable Interrupts		—	0	—	—	—	INH	8F		2

A	Accumulator
C	Carry/borrow flag
CCR	Condition code register
dd	Direct address of operand
dd rr	Direct address of operand and relative offset of branch instruction
DIR	Direct addressing mode
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing
EXT	Extended addressing mode
ff	Offset byte in indexed, 8-bit offset addressing
H	Half-carry flag
hh ll	High and low bytes of operand address in extended addressing
I	Interrupt mask
ii	Immediate operand byte
IMM	Immediate addressing mode
INH	Inherent addressing mode
IX	Indexed, no offset addressing mode
IX1	Indexed, 8-bit offset addressing mode
IX2	Indexed, 16-bit offset addressing mode
M	Memory location
N	Negative flag
n	Any bit

opr	Operand (one or two bytes)
PC	Program counter
PCH	Program counter high byte
PCL	Program counter low byte
REL	Relative addressing mode
rel	Relative program counter offset byte
rr	Relative program counter offset byte
SP	Stack pointer
X	Index register
Z	Zero flag
#	Immediate value
∧	Logical AND
∨	Logical OR
⊕	Logical EXCLUSIVE OR
( )	Contents of
–( )	Negation (two’s complement)
←	Loaded with
?	If
:	Concatenated with
↕	Set or cleared
—	Not affected

Table 16-7. Opcode Map

	Bit Manipulation		Branch	Read-Modify-Write					Control		Register/Memory						
	DIR	DIR	REL	DIR	INH	INH	IX1	IX	INH	INH	IMM	DIR	EXT	IX2	IX1	IX	
MSB LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSB LSB
0	5 BRSET0 3 DIR	5 BSET0 2 DIR	3 BRA 2 REL	5 NEG 2 DIR	3 NEGA 1 INH	3 NEGX 1 INH	6 NEG 2 IX1	5 NEG 1 IX	9 RTI 1 INH		2 SUB 2 IMM	3 SUB 2 DIR	4 SUB 3 EXT	5 SUB 3 IX2	4 SUB 2 IX1	3 SUB 1 IX	**0**
1	5 BRCLR0 3 DIR	5 BCLR0 2 DIR	3 BRN 2 REL						6 RTS 1 INH		2 CMP 2 IMM	3 CMP 2 DIR	4 CMP 3 EXT	5 CMP 3 IX2	4 CMP 2 IX1	3 CMP 1 IX	**1**
2	5 BRSET1 3 DIR	5 BSET1 2 DIR	3 BHI 2 REL		11 MUL 1 INH						2 SBC 2 IMM	3 SBC 2 DIR	4 SBC 3 EXT	5 SBC 3 IX2	4 SBC 2 IX1	3 SBC 1 IX	**2**
3	5 BRCLR1 3 DIR	5 BCLR1 2 DIR	3 BLS 2 REL	5 COM 2 DIR	3 COMA 1 INH	3 COMX 1 INH	6 COM 2 IX1	5 COM 1 IX	10 SWI 1 INH		2 CPX 2 IMM	3 CPX 2 DIR	4 CPX 3 EXT	5 CPX 3 IX2	4 CPX 2 IX1	3 CPX 1 IX	**3**
4	5 BRSET2 3 DIR	5 BSET2 2 DIR	3 BCC 2 REL	5 LSR 2 DIR	3 LSRA 1 INH	3 LSRX 1 INH	6 LSR 2 IX1	5 LSR 1 IX			2 AND 2 IMM	3 AND 2 DIR	4 AND 3 EXT	5 AND 3 IX2	4 AND 2 IX1	3 AND 1 IX	**4**
5	5 BRCLR2 3 DIR	5 BCLR2 2 DIR	3 BCS/BLO 2 REL								2 BIT 2 IMM	3 BIT 2 DIR	4 BIT 3 EXT	5 BIT 3 IX2	4 BIT 2 IX1	3 BIT 1 IX	**5**
6	5 BRSET3 3 DIR	5 BSET3 2 DIR	3 BNE 2 REL	5 ROR 2 DIR	3 RORA 1 INH	3 RORX 1 INH	6 ROR 2 IX1	5 ROR 1 IX			2 LDA 2 IMM	3 LDA 2 DIR	4 LDA 3 EXT	5 LDA 3 IX2	4 LDA 2 IX1	3 LDA 1 IX	**6**
7	5 BRCLR3 3 DIR	5 BCLR3 2 DIR	3 BEQ 2 REL	5 ASR 2 DIR	3 ASRA 1 INH	3 ASRX 1 INH	6 ASR 2 IX1	5 ASR 1 IX		2 TAX 1 INH		4 STA 2 DIR	5 STA 3 EXT	6 STA 3 IX2	5 STA 2 IX1	4 STA 1 IX	**7**
8	5 BRSET4 3 DIR	5 BSET4 2 DIR	3 BHCC 2 REL	5 ASL/LSL 2 DIR	3 ASLA/LSLA 1 INH	3 ASLX/LSLX 1 INH	6 ASL/LSL 2 IX1	5 ASL/LSL 1 IX		2 CLC 1 INH	2 EOR 2 IMM	3 EOR 2 DIR	4 EOR 3 EXT	5 EOR 3 IX2	4 EOR 2 IX1	3 EOR 1 IX	**8**
9	5 BRCLR4 3 DIR	5 BCLR4 2 DIR	3 BHCS 2 REL	5 ROL 2 DIR	3 ROLA 1 INH	3 ROLX 1 INH	6 ROL 2 IX1	5 ROL 1 IX		2 SEC 1 INH	2 ADC 2 IMM	3 ADC 2 DIR	4 ADC 3 EXT	5 ADC 3 IX2	4 ADC 2 IX1	3 ADC 1 IX	**9**
A	5 BRSET5 3 DIR	5 BSET5 2 DIR	3 BPL 2 REL	5 DEC 2 DIR	3 DECA 1 INH	3 DECX 1 INH	6 DEC 2 IX1	5 DEC 1 IX		2 CLI 1 INH	2 ORA 2 IMM	3 ORA 2 DIR	4 ORA 3 EXT	5 ORA 3 IX2	4 ORA 2 IX1	3 ORA 1 IX	**A**
B	5 BRCLR5 3 DIR	5 BCLR5 2 DIR	3 BMI 2 REL							2 SEI 1 INH	2 ADD 2 IMM	3 ADD 2 DIR	4 ADD 3 EXT	5 ADD 3 IX2	4 ADD 2 IX1	3 ADD 1 IX	**B**
C	5 BRSET6 3 DIR	5 BSET6 2 DIR	3 BMC 2 REL	5 INC 2 DIR	3 INCA 1 INH	3 INCX 1 INH	6 INC 2 IX1	5 INC 1 IX		2 RSP 1 INH		2 JMP 2 DIR	3 JMP 3 EXT	4 JMP 3 IX2	3 JMP 2 IX1	2 JMP 1 IX	**C**
D	5 BRCLR6 3 DIR	5 BCLR6 2 DIR	3 BMS 2 REL	4 TST 2 DIR	3 TSTA 1 INH	3 TSTX 1 INH	5 TST 2 IX1	4 TST 1 IX		2 NOP 1 INH	6 BSR 2 REL	5 JSR 2 DIR	6 JSR 3 EXT	7 JSR 3 IX2	6 JSR 2 IX1	5 JSR 1 IX	**D**
E	5 BRSET7 3 DIR	5 BSET7 2 DIR	3 BIL 2 REL						2 STOP 1 INH		2 LDX 2 IMM	3 LDX 2 DIR	4 LDX 3 EXT	5 LDX 3 IX2	4 LDX 2 IX1	3 LDX 1 IX	**E**
F	5 BRCLR7 3 DIR	5 BCLR7 2 DIR	3 BIH 2 REL	5 CLR 2 DIR	3 CLRA 1 INH	3 CLRX 1 INH	6 CLR 2 IX1	5 CLR 1 IX	2 WAIT 1 INH	2 TXA 1 INH		4 STX 2 DIR	5 STX 3 EXT	6 STX 3 IX2	5 STX 2 IX1	4 STX 1 IX	**F**

INH = Inherent	REL = Relative
IMM = Immediate	IX = Indexed, No Offset
DIR = Direct	IX1 = Indexed, 8-Bit Offset
EXT = Extended	IX2 = Indexed, 16-Bit Offset

	MSB LSB	0	MSB of Opcode in Hexadecimal
LSB of Opcode in Hexadecimal	0	5 BRSET0 3 DIR	Number of Cycles Opcode Mnemonic Number of Bytes/Addressing Mode

Section 17. Electrical Specifications

17.1 Contents

17.2 Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

The MCU contains circuitry to protect the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in the table here. Keep V_{In} and V_{Out} within the range $V_{SS} \leq (V_{In}$ or $V_{Out}) \leq V_{DD}$. Connect unused inputs to the appropriate voltage level, either V_{SS} or V_{DD}.

Rating	Symbol	Value	Unit
Supply voltage	V_{PGC}, V_{GSUP}, and V_{GREF} V_{DD} V_{CCA}	–0.5 to +42.0 –0.5 to +7.0 V_{DD}	V
Input voltage	V_{In}	V_{SS} –0.3 to V_{DD} +0.3	V
Current drain per pin (I/O) Current drain per pin (gauge)	I	25 50	mA
Storage temperature range	T_{STG}	–65 to +150	°C
Write/erase cycles (@ 10 ms write time and –40°C, +25°C, and +85°C)	—	10,000	Cycles
Data retention EPROM, EEPROM (–40°C to + 85°C)	—	10	Years

NOTE: *This device is not guaranteed to operate properly at the maximum ratings. Refer to* ***17.6 DC Electrical Characteristics*** *for guaranteed operating conditions.*

17.3 Operating Temperature Range

Characteristic	Symbol	Value	Unit
Operating temperature range Standard Extended	T_A	T_L to T_H 0 to +70 –40 to +85	°C
Maximum junction temperature	T_J	150	°C

17.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance PLCC (68 pin)	θ_{JA}	50	°C/W

17.5 Power Considerations

The average chip junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \text{ x } \theta_{JA}) \qquad (1)$$

Where:

T_A = ambient temperature in °C
θ_{JA} = package thermal resistance, junction to ambient in °C/W
$P_D = P_{INT} + P_{I/O}$
$P_{INT} = I_{CC} \times V_{CC}$ = chip internal power dissipation
$P_{I/O}$ = power dissipation on input and output pins (user-determined)

For most applications, $P_{I/O} \ll P_{INT}$ and can be neglected.

Ignoring $P_{I/O}$, the relationship between P_D and T_J is approximately:

$$P_D = \frac{K}{T_J + 273°C} \qquad (2)$$

Solving equations (1) and (2) for K gives:

$$= P_D \text{ x } (T_A + 273°C) + \theta_{JA} \text{ x } (P_D)^2 \qquad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

17.6 DC Electrical Characteristics

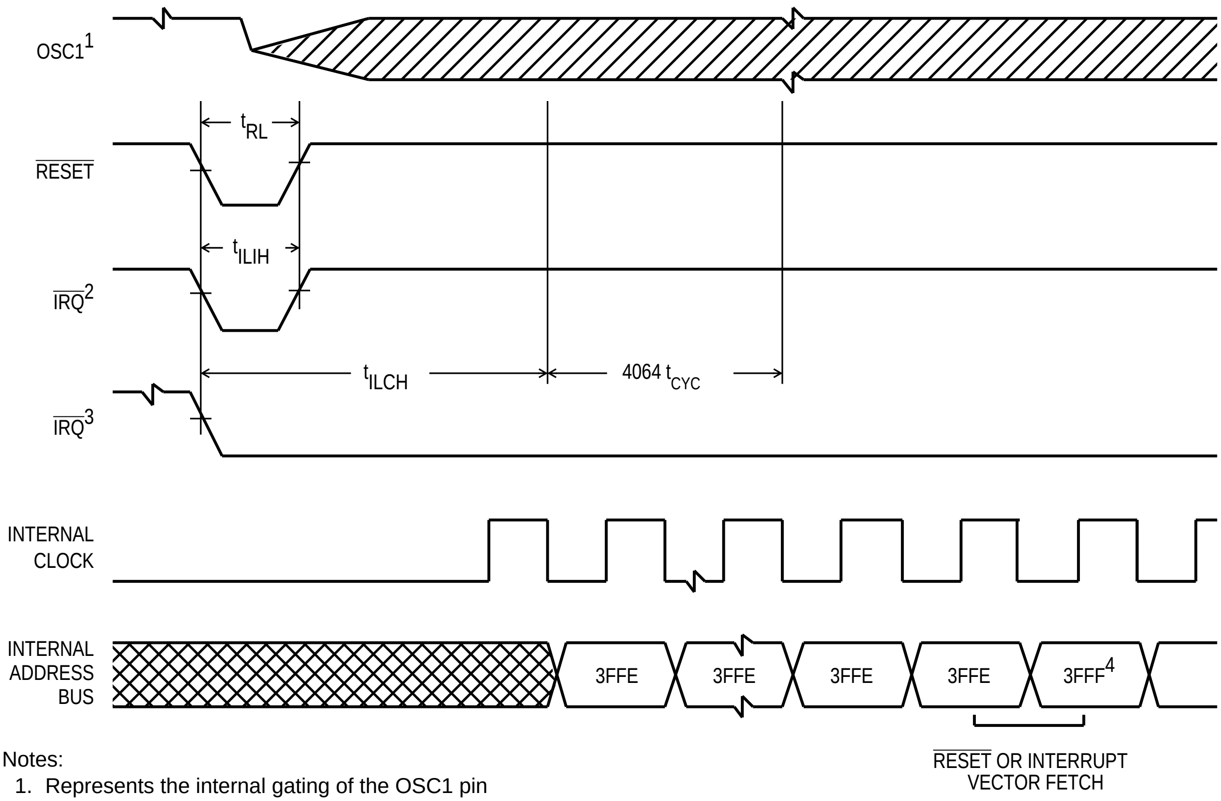
Characteristic[1] [2]	Symbol	Min	Max	Unit
Output voltage I_{Load} = 10.0 µA I_{Load} = –10.0 µA	 V_{OL} V_{OH}	 — V_{DD} –0.1	 0.1 —	 V
Output high voltage (I_{Load} –0.8 mA) port A, port B, port C, TXP	V_{OH}	V_{DD} –0.8	—	V
Output low voltage (I_{Load} = 1.6 mA) port A, port B, port C, TXP	V_{OL}	—	0.4	V
Input high voltage Port A, port B, port C port D, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, OSC1, RXP	V_{IH}	0.7 x V_{DD}	V_{DD}	V
Input low voltage Port A, port B, port C, port D, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, OSC1, RXP	V_{IL}	V_{SS}	0.3 x V_{DD}	V
EPROM/MOR programming voltage	V_{pp}	15.5	16.5	V
EPROM/MOR programming current	I_{pp}	—	10	mA
Supply current[3] Run[4] Wait SPI, TIMER, A/D, PWM, COP, LVR on[5] Wait above modules off Stop[6] LVR enabled LVR disabled	I_{DD}	 — — — — —	 10 6 4 300 200	 mA mA mA µA µA
I/O ports hi-z leakage current Port A, port B, port C	I_{oz}	—	1	µA
Input current $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$ OSC1, PD0–PD4	I_{In}	 — —	 10 1	µA
Capacitance[7] Ports (as input or output) $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$	 C_{Out} C_{In}	 — —	 12 8	pF
Low-voltage reset inhibit	V_{LVRI}	3.5	4.2	V
Low-voltage reset recover	V_{LVRR}	3.6	4.5	V
Low-voltage reset inhibit/recover hysteresis	H_{LVR}	0.1	0.3	V
V_{DD} slew rate rising[7]	S_{VDDR}	—	0.1	V/µs
V_{DD} slew rate falling[7]	S_{VDDF}	—	0.05	V/µs
POR reset voltage[7]	V_{POR}	—	100	mV

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +85°C, unless otherwise noted. All values shown reflect average measurements.
2. All coil drivers are set to the maximum current in automatic mode with no loading on the gauge pins.
3. Wait, Stop I_{DD}: All ports configured as inputs, V_{IL} = 0.2 Vdc, V_{IH} = V_{DD} –0.2 Vdc.
4. Run (Operating) I_{DD}, wait I_{DD}: Measured using external square wave clock source to OSC1 (f_{OSC} = 4.2 MHz), all inputs 0.2 Vdc from rail; no DC loads, less than 50 pF on all outputs, C_L = 20 pF on OSC2.
5. Wait I_{DD} is affected linearly by the OSC2 capacitance.
6. Stop I_{DD} measured with OSC1 = V_{SS}.
7. Not tested

17.7 Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Frequency of operation Crystal oscillator option External clock source	f_{OSC}	0.1 dc	4.2 4.2	MHz
Internal operating frequency Crystal (f_{OSC} /2) External clock (f_{OSC} /2)	f_{OP}	— dc	2.1 2.1	MHz
Cycle time ($1/f_{OP}$)	t_{CYC}	476	—	ns
Crystal oscillator startup time (crystal oscillator option)	t_{OXON}	—	100	ms
Stop recovery startup time (crystal oscillator option)	t_{ILCH}	—	100	ms
$\overline{\text{RESET}}$ pulse width low See **Figure 5-2. Reset and POR Timing Diagram**.	t_{RL}	120	—	ns
Interrupt pulse width low (edge-triggered)	t_{ILIH}	120	—	ns
Interrupt pulse period[2]	t_{ILIL}	Note 2	—	t_{CYC}
Port C interrupt pulse width high (edge-triggered)	t_{ILHI}	120	—	ns
Port C interrupt pulse period[2]	t_{IHIH}	Note 2	—	t_{CYC}
OSC1 pulse width	t_{OSC1}	90	—	ns
EPROM programming time per byte	t_{EPGM}	4	—	ms
EEPROM programming time per byte	t_{EEPGM}	10	—	ms
EEPROM erase time per byte	t_{EBYT}	10	—	ms
EEPROM erase time per block	t_{EBLOCK}	10	—	ms
EEPROM bulk erase time	t_{EBULK}	10	—	ms
EEPROM programming voltage discharge period	t_{FPV}	—	10.0	µs
RC oscillator stabilization time	t_{RCON}	—	5	t_{CYC}
16-bit timer Resolution[3] Input capture pulse width Input capture period[4]	t_{RESL} $t_{TH,\ tTL}$ t_{TLTL}	4.0 85 Note 4	— — —	t_{CYC} ns t_{CYC}

1. V_{DD} = 5.0 Vdc, V_{SS} = 0 Vdc, T_A = –40°C to +85°C, unless otherwise noted.
2. The minimum period, t_{ILIL} or $t_{IHIH,}$ should not be less than the number of cycles it takes to execute the interrupt service routine plus 19 t_{CYC}.
3. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
4. The minimum period, t_{TLTL}, should not be less than the number of cycles it takes to execute the capture interrupt service routine plus 24 t_{CYC}.



Notes:

1. Represents the internal gating of the OSC1 pin
2. IRQ pin is edge-sensitive mask option.
3. IRQ pin is level- and edge-sensitive mask option.
4. RESET vector address is shown for timing example.

Figure 17-1. Stop Recovery Timing Diagram

17.8 A/D Converter Characteristics

Characteristic[1]	Min	Max	Unit	Comments
Resolution	8	8	Bits	
Absolute accuracy V_{REFL} = 0.0 V, V_{REFH} = V_{DD}	—	±1	LSB	Include quantization
Conversion range V_{REFH} V_{REFL}	V_{REFL} V_{REFL} –0.1	V_{REFH} V_{DD} V_{REFH}	V	A/D accuracy decreases proportionately as V_{REFH} is reduced below minimum V_{CCA}. Not tested
Power-up time	—	100	µs	
Input leakage PD0–PD4 V_{REFL}, V_{REFH}	— —	±1 ±1	µA	
Conversion time[2] (Includes sampling time)	32	32	t_{AD} (Note 2)	
Monotonicity				Inherent (within total error)
Zero input reading	00	01	Hex	V_{In} = 0 V
Full-scale reading	FE	FF	Hex	V_{In} = V_{REFH}
Sample time	12	12	t_{AD} (Note 3)	
Input capacitance	—	8	pF	Not tested
Analog input voltage	V_{REFL}	V_{REFH}	V	
A/D on current stabilization time t_{ADON}	—	100	µs	
RC oscillator stabilization time t_{RCON}	—	5	µs	

[1] V_{CCA} = 5.0 ± 10% Vdc ± 10%, V_{SSA} = 0.0 Vdc, T_A = –40°C to +85°C, unless otherwise noted.

[2] t_{AD} = t_{CYC} if clock source equals MCU.

17.9 LVR Timing Diagram

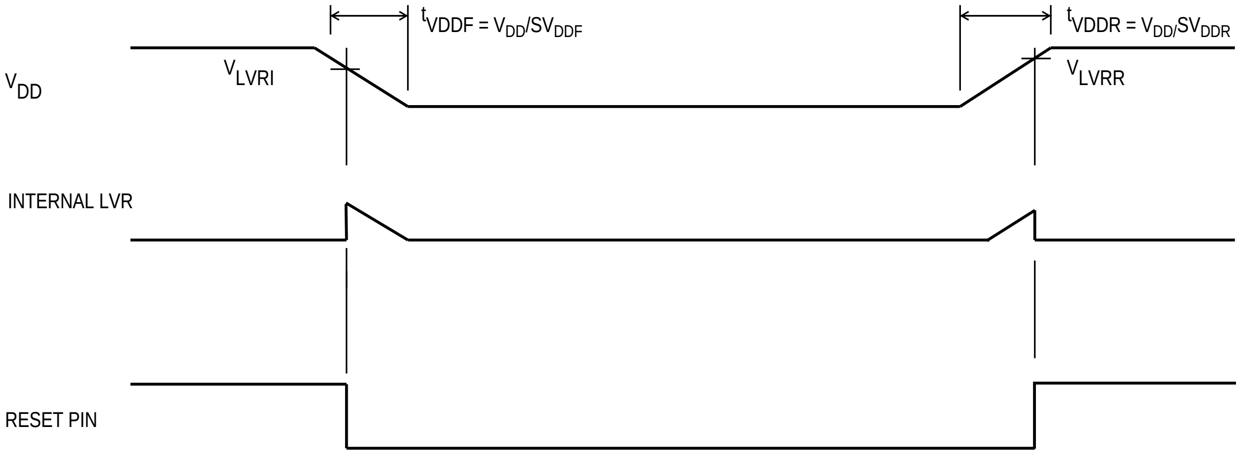


Figure 17-2. LVR Timing Diagram

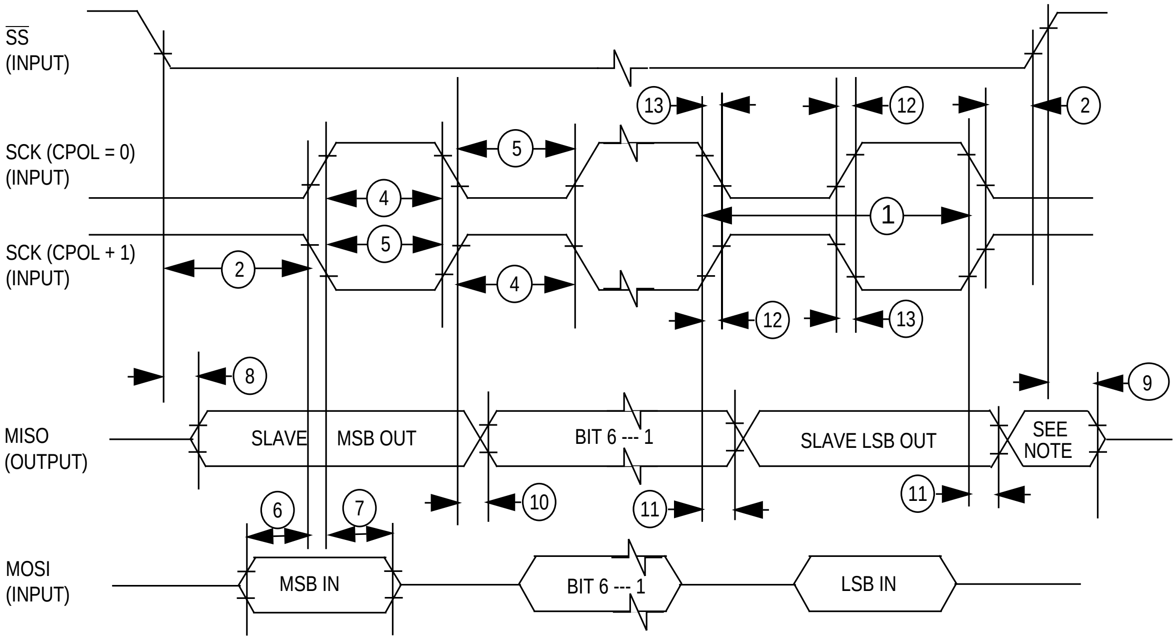
17.10 Serial Peripheral Interface (SPI) Timing

Num	Characteristic(1)	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(s)}$	dc dc	0.5 4.2	f_{OP} MHz
1	Cycle time Master Slave	$t_{CYC(m)}$ $t_{CYC(s)}$	2.0 240	— —	t_{CYC} ns
2	Enable lead time Master(2) Slave	$t_{LEAD(M)}$ lLEAD(S)	Note 2 240	— —	ns
3	Enable lag time Master(2) Slave	$t_{LAG(m)}$ $t_{LAG(s)}$	Note 2 240	— —	ns
4	Clock (SCK) high time Master Slave	$t_{w(SCKH)m}$ $t_{w(SCKH)s}$	340 190	— —	ns
5	Clock (SCK) low time Master Slave	$t_{w(SCKL)m}$ $t_{w(SCKL)s}$	340 190	— —	ns
6	Data setup time (inputs) Master Slave	$t_{SU(m)}$ $t_{SU(s)}$	100 100	— —	ns
7	Data hold time (inputs) Master Slave	$t_{H(m)}$ $t_{H(s)}$	100 100	— —	ns
8	Access time (time to data active from high-impedance state) Slave	t_A	0	120	ns
9	Disable time (hold time to high-impedance state) Slave	t_{DIS}	—	240	ns
10	Data valid (after enable edge)(3)	$t_{V(s)}$	—	240	ns
11	Data hold time (output) (after enable edge)	t_{HO}	0	—	ns
12	Rise time (20% V_{DD} to 70% V_{DD}, C_L = 200 pF) SPI outputs (SCK, MOSI, and MISO) SPI inputs (SCK, MOSI, MISO, and $\overline{SS}$)	t_{RM} t_{RS}	— —	100 2.0	ns µs
13	Fall time (20% V_{DD} to 70% V_{DD}, C_L = 200 pF) SPI outputs (SCK, MOSI, and MISO) SPI inputs (SCK, MOSI, MISO, and $\overline{SS}$)	t_{FM} t_{FS}	— —	100 2.0	ns µs

[1] V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, T_A = T_L to T_H

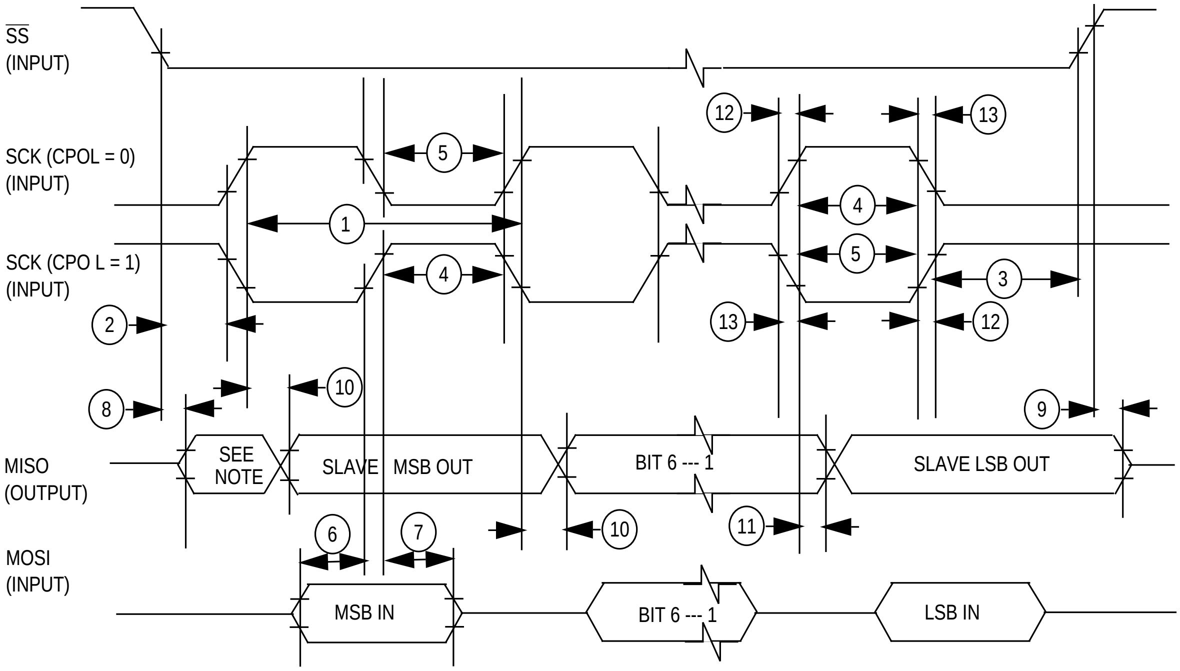
[2] Signal production depends on software.

[3] Assumes 200 pF load on all SPI pins



Note: Not defined, but normally LSB of character previously transmitted

Figure 17-3. SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

Figure 17-4. SPI Slave Timing (CPHA = 1)

17.11 Gauge Driver Electricals

Characteristic[1]	Symbol	Min[2]	Max[2]	Unit
Input current on V_{GSUP} with no coil current Input current on V_{GSUP} with coil current[3] Input current on V_{GSUP} in stop mode	I_{GSUP}	— — —	5 135 40	mA mA µA
Maximum reference current	I_{MAX}	0.47	0.57	mA
Internal total series impedance	R_{SI}	20	60	Ω
Manual scan cycle period	t_{MSN}	12 * t_{GCS}	20	ms
Coil inductance[4]	L_{Coil}	—	31	mH
Coil resistance[4]	R_{Coil}	140	270	Ω
Error tolerance of R_{MAX}	E_{MAX}		±1	%
Coil current matching error (as % of I_{CM})	E_{CM}	0	±1	%
Coil current absolute error	E_{CA}	0	±9	%
Coil current update time	t_{GCS}	—	1.67	ms
Coil current maximum	I_{CM}	—	23	mA
Gauge supply regular error	E_{GS}	—	±5	%
Monotonicity[5]		Note 5		
Coil current step size	I_{Step}	(I_{CM}/255) * 0.50	(I_{CM}/255) * 1.50	mA

1. V_{GSUP} = 7.6 Vdc, T_A = –40°C to +85°C, unless otherwise noted.
2. Minimum/maximum is dependent upon power calculation.
3. Assumes sin/cos
4. Coil is not on chip; values stated are indicative of the intended application.
5. Inherent within total error

17.12 BDLC Transmitter VPW Symbol Timings (BARD) Bits BO[3:0] = 0111

Characteristic[1]	Number	Symbol	Min	Typ	Max	Unit
Passive logic 0	10	t_{TVP1}	62	64	66	µs
Passive logic 1	11	t_{TVP2}	126	128	130	µs
Active logic 0	12	t_{TVA1}	126	128	130	µs
Active logic 1	13	t_{TVA2}	62	64	66	µs
Start of frame (SOF)	14	t_{TVA3}	198	200	202	µs
End of data (EOD)	15	t_{TVP3}	198	200	202	µs
End of frame (EOF)	16	t_{TV4}	278	280	282	µs
Inter-frame separator (IFS)	17	t_{TV6}	298	300	302	µs

[1] f_{BDLC} = 1.048576 MHz or 1.0 MHz

17.13 BDLC Receiver VPW Symbol Timings (BARD) Bits BO[3:0] = 0111

Characteristic[1]	Number	Symbol	Min	Typ	Max	Unit
Passive logic 0	10	t_{TRVP1}	34	64	96	µs
Passive logic 1	11	t_{TRVP2}	96	128	163	µs
Active logic 0	12	t_{TRVA1}	96	128	163	µs
Active logic 1	13	t_{TRVA2}	34	64	96	µs
Start of frame (SOF)	14	t_{TRVA3}	163	200	239	µs
End of data (EOD)	15	t_{TRVP3}	163	200	239	µs
End of frame (EOF)	16	t_{TRV4}	239	280	320	µs
Break	18	t_{TRV6}	239	—	—	µs

[1] f_{BDLC} = 1.048576 MHz or 1.0 MHz

NOTE: *The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{BDLC} µs due to sampling considerations.*

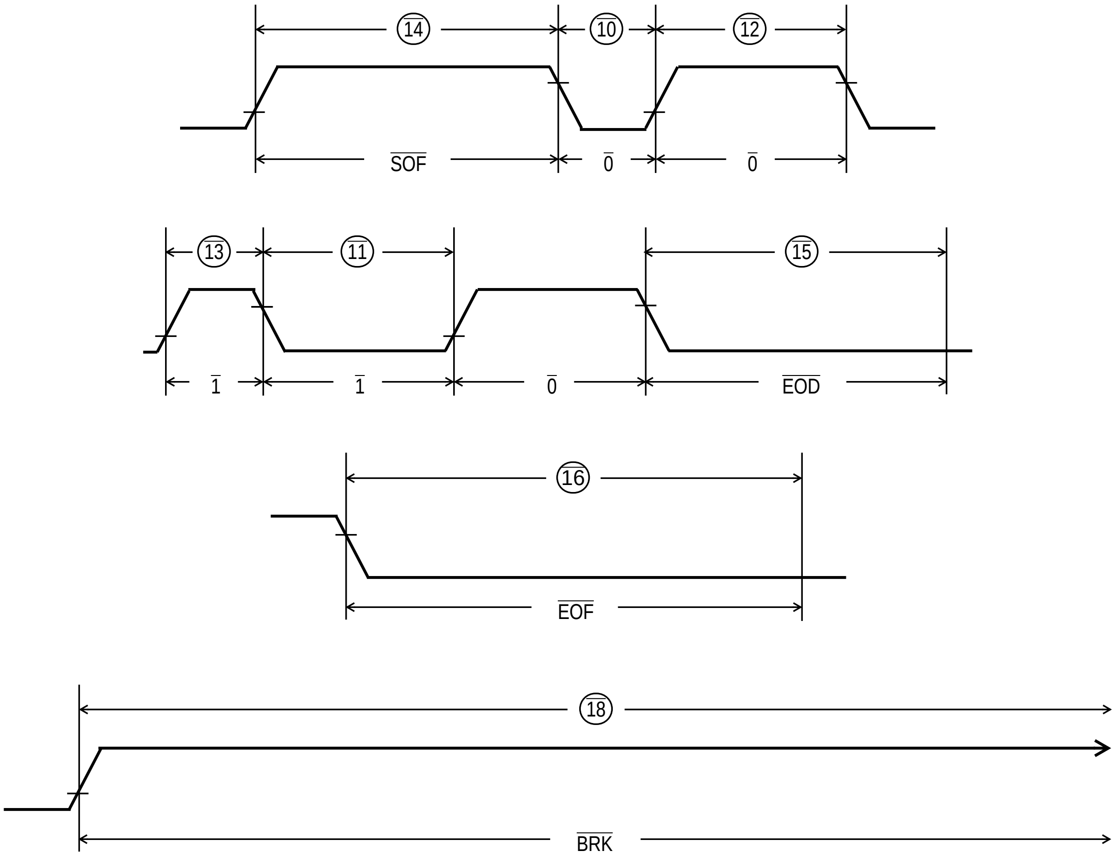


Figure 17-5. BDLC Variable Pulse Width Modulation (VPW) Symbol Timings

Section 18. Mechanical Specifications

18.1 Contents

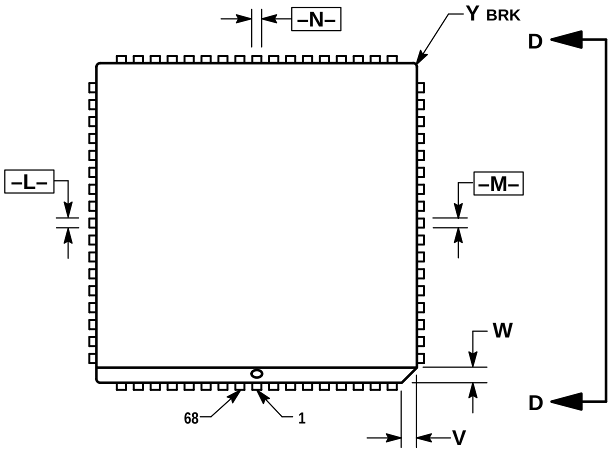
18.2 Introduction

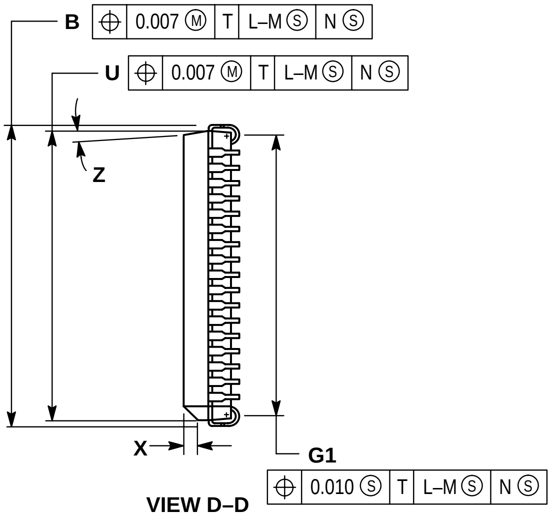
This section describes the dimensions of the 68-lead plastic leaded chip carrier (PLCC). Package dimensions available at the time of this publication are provided in this section. To verify the latest case outline specifications, contact one of the following:

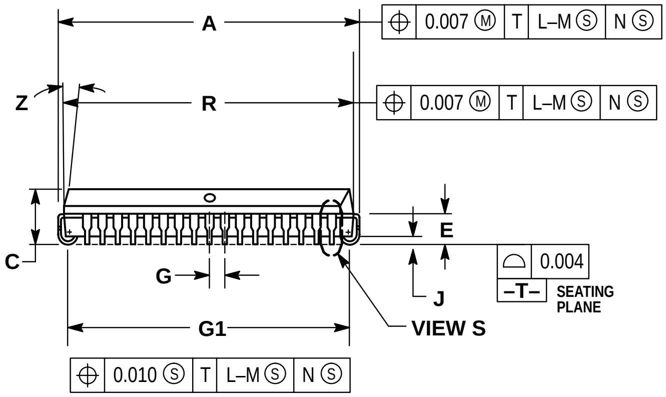
- Local Motorola Sales Office
- Motorola Mfax:
  - Phone 602-244-6609
  - EMAIL rmfax0@email.sps.mot.com
- Worldwide Web (wwweb) at http://design-net.com

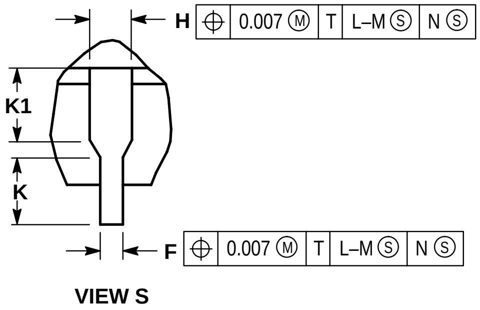
Follow Mfax or wwweb on-line instructions to retrieve the current mechanical specifications.

18.3 68-Lead Plastic Leaded Chip Carrier (PLCC)




B
⌖ 0.007 Ⓜ | T | L–M Ⓢ | N Ⓢ
U
⌖ 0.007 Ⓜ | T | L–M Ⓢ | N Ⓢ
Z
X
G1
⌖ 0.010 Ⓢ | T | L–M Ⓢ | N Ⓢ
VIEW D–D


A
⌖ 0.007 Ⓜ | T | L–M Ⓢ | N Ⓢ
Z
R
⌖ 0.007 Ⓜ | T | L–M Ⓢ | N Ⓢ
E
C
G
J
⌓ 0.004
–T– SEATING PLANE
VIEW S
G1
⌖ 0.010 Ⓢ | T | L–M Ⓢ | N Ⓢ



NOTES:

1. DATUMS L, M, AND N DETERMINED WHERE TOP OF LEAD SHOULDER EXITS PLASTIC BODY AT MOLD PARTING LINE.
2. DIMENSION G1, TRUE POSITION TO BE MEASURED AT DATUM T, SEATING PLANE.
3. DIMENSIONS R AND U DO NOT INCLUDE MOLD FLASH. ALLOWABLE MOLD FLASH IS 0.010 PER SIDE.
4. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
5. CONTROLLING DIMENSION: INCH.
6. THE PACKAGE TOP MAY BE SMALLER THAN THE PACKAGE BOTTOM BY UP TO 0.012. DIMENSIONS R AND U ARE DETERMINED AT THE OUTERMOST EXTREMES OF THE PLASTIC BODY EXCLUSIVE OF MOLD FLASH, TIE BAR BURRS, GATE BURRS AND INTERLEAD FLASH, BUT INCLUDING ANY MISMATCH BETWEEN THE TOP AND BOTTOM OF THE PLASTIC BODY.
7. DIMENSION H DOES NOT INCLUDE DAMBAR PROTRUSION OR INTRUSION. THE DAMBAR PROTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE GREATER THAN 0.037. THE DAMBAR INTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE SMALLER THAN 0.025.

	INCHES	
DIM	MIN	MAX
A	0.985	0.995
B	0.985	0.995
C	0.165	0.180
E	0.090	0.110
F	0.013	0.019
G	0.050 BSC	
H	0.026	0.032
J	0.020	—
K	0.025	—
R	0.950	0.956
U	0.950	0.956
V	0.042	0.048
W	0.042	0.048
X	0.042	0.056
Y	—	0.020
Z	2°	10°
G1	0.910	0.930
K1	0.040	—

Section 19. Ordering Information

19.1 Contents

19.2 Introduction

This section contains ordering information.

19.3 MC Order Number

Table 19-1 shows the MC order number for the available package type.

Table 19-1. MC Order Number

Package Type	Temperature Range	Order Number
68-lead plastic leaded chip carrier (PLCC)	–40°C to 85°C	MC68HC705V12CFN[1]

[1] FN = Plastic leaded chip carrier (PLCC)

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140
or 1-800-441-2447. Customer Focus Center, 1-800-521-6274

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan.
81-3-3440-8573

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong. 852-26668334

Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/;
TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848

HOME PAGE: http://motorola.com/sps/

Mfax is a trademark of Motorola, Inc.

MOTOROLA