

MC68HC05L28/D

HC05

MC68HC05L28
MC68HC705L28

TECHNICAL
DATA

MC68HC05L28

TECHNICAL DATA



INTRODUCTION	1
MODES OF OPERATION AND PIN DESCRIPTIONS	2
MEMORY AND REGISTERS	3
INPUT/OUTPUT PORTS	4
CORE TIMER	5
16-BIT PROGRAMMABLE TIMER	6
LIQUID CRYSTAL DISPLAY DRIVER MODULE	7
I ² C-BUS	8
A/D CONVERTER	9
RESETS AND INTERRUPTS	10
CPU CORE AND INSTRUCTION SET	11
ELECTRICAL SPECIFICATIONS	12
MECHANICAL DATA	13
ORDERING INFORMATION	14
MC68HC705L28	A

MC68HC05L28

MC68HC705L28

High-density complementary metal oxide semiconductor (HCMOS) microcontroller unit

All Trade Marks recognized. This document contains information on new products. Specifications and information herein are subject to change without notice.

All products are sold on Motorola's Terms & Conditions of Supply. In ordering a product covered by this document the Customer agrees to be bound by those Terms & Conditions and nothing contained in this document constitutes or forms part of a contract (with the exception of the contents of this Notice). A copy of Motorola's Terms & Conditions of Supply is available on request.

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 can and do vary in different applications. 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.

The Customer should ensure that it has the most up to date version of the document by contacting its local Motorola office. This document supersedes any earlier documentation relating to the products referred to herein. The information contained in this document is current at the date of publication. It may subsequently be updated, revised or withdrawn.

Conventions

Where abbreviations are used in the text, an explanation can be found in the glossary, at the back of this manual. Register and bit mnemonics are defined in the paragraphs describing them.

An overbar is used to designate an active-low signal, eg: $\overline{\text{RESET}}$.

Unless otherwise stated, shaded cells in a register diagram indicate that the bit is either unused or reserved; 'u' is used to indicate an undefined state (on reset).

CUSTOMER FEEDBACK QUESTIONNAIRE (MC68HC05L28/D)

Motorola wishes to continue to improve the quality of its documentation. We would welcome your feedback on the publication you have just received. Having used the document, please complete this card (or a photocopy of it, if you prefer).

1. How would you rate the quality of the document? Check one box in each category.

	Excellent		Poor			Excellent		Poor	
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Tables	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Readability	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Table of contents	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Understandability	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Index	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Page size/binding	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Illustrations	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Overall impression	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments:	<hr/>								

2. What is your intended use for this document? If more than one option applies, please rank them (1, 2, 3).

Selection of device for new application	<input type="checkbox"/>	Other <input type="checkbox"/> Please specify: <hr/>
System design	<input type="checkbox"/>	<hr/>
Training purposes	<input type="checkbox"/>	<hr/>

3. How well does this manual enable you to perform the task(s) outlined in question 2?

Completely			Not at all	Comments:
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<hr/>

4. How easy is it to find the information you are looking for?

Easy			Difficult	Comments:
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<hr/>

5. Is the level of technical detail in the following sections sufficient to allow you to understand how the device functions?

	Too little detail		Too much detail	
SECTION 1 INTRODUCTION	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 2 MODES OF OPERATION AND PIN DESCRIPTIONS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 3 MEMORY AND REGISTERS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 4 INPUT/OUTPUT PORTS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 5 CORE TIMER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 6 16-BIT PROGRAMMABLE TIMER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 7 LIQUID CRYSTAL DISPLAY DRIVER MODULE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 8 I ² C-BUS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 9 A/D CONVERTER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 10 RESETS AND INTERRUPTS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 11 CPU CORE AND INSTRUCTION SET	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 12 ELECTRICAL SPECIFICATIONS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 13 MECHANICAL DATA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION 14 ORDERING INFORMATION	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SECTION A MC68HC705L28	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Comments:

6. Have you found any errors? If so, please comment:

7. From your point of view, is anything missing from the document? If so, please say what:

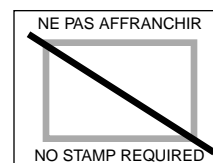


8. How could we improve this document? _____
9. How would you rate Motorola's documentation?
- | | Excellent | | Poor | |
|---|--------------------------|--------------------------|--------------------------|--------------------------|
| - In general | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| - Against other semiconductor suppliers | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
10. Which semiconductor manufacturer provides the best technical documentation? _____
11. Which company (in any field) provides the best technical documentation? _____
12. How many years have you worked with microprocessors?
- | | | | | | | | |
|------------------|--------------------------|-----------|--------------------------|-----------|--------------------------|-------------------|--------------------------|
| Less than 1 year | <input type="checkbox"/> | 1-3 years | <input type="checkbox"/> | 3-5 years | <input type="checkbox"/> | More than 5 years | <input type="checkbox"/> |
|------------------|--------------------------|-----------|--------------------------|-----------|--------------------------|-------------------|--------------------------|

- Second fold back along this line -

**By air mail
Par avion**

IBRS NUMBER PHQ-B/207/G
CCRI NUMERO PHQ-B/207/G



REPONSE PAYEE GRANDE-BRETAGNE

Motorola Ltd.,
Colvilles Road,
Kelvin Industrial Estate,
EAST KILBRIDE,
G75 8BR.
GREAT BRITAIN.

!MOTOROLA
Semiconductor Products Sector

F.A.O. Technical Publications Manager
(re: MC68HC05L28/D)

- First fold back along this line -

- Cut along this line to remove -

- Third fold back along this line -

13. Currently there is some discussion in the semiconductor industry regarding a move towards providing data sheets in electronic form. If you have any opinion on this subject, please comment. _____
14. We would be grateful if you would supply the following information (at your discretion), or attach your card.
- | | | | |
|-------------|-------|-----------|-------|
| Name: | _____ | Phone No: | _____ |
| Position: | _____ | FAX No: | _____ |
| Department: | _____ | | |
| Company: | _____ | | |
| Address: | _____ | | |

*Thank you for helping us improve our documentation,
Graham Livey, Technical Publications Manager, Motorola Ltd., Scotland.*

- Finally, tuck this edge into opposite flap -

TABLE OF CONTENTS

Paragraph Number	TITLE	Page Number
1		
INTRODUCTION		
1.1	Features.....	1-1
1.2	Mask options on the MC68HC05L28.....	1-3
1.2.1	Option register (OPT).....	1-3
2		
MODES OF OPERATION AND PIN DESCRIPTIONS		
2.1	Modes of operation.....	2-1
2.1.1	MC68HC05L28 modes of operation.....	2-2
2.1.1.1	Single chip mode.....	2-2
2.1.1.2	RAM bootloader mode.....	2-2
2.1.2	<i>MC68HC705L28 modes of operation</i>	2-4
2.1.2.1	<i>EPROM bootloader mode</i>	2-4
2.1.2.2	RAM bootloader mode.....	2-4
2.2	Pin descriptions.....	2-6
2.2.1	VDD and VSS.....	2-6
2.2.2	$\overline{\text{IRQ0}}$	2-6
2.2.3	IRQ1.....	2-6
2.2.4	IRQ2.....	2-6
2.2.5	OSC1, OSC2.....	2-7
2.2.5.1	Crystal.....	2-8
2.2.5.2	External clock.....	2-8
2.2.6	$\overline{\text{RESET}}$	2-8
2.2.7	PA0 – PA7, PB0 – PB7.....	2-9
2.2.8	PD0 – PD5.....	2-9
2.2.9	BP0 – BP3, FP0 – FP17.....	2-9
2.2.10	AD0 – AD1, VREFH/VREFL.....	2-9
2.3	Low power modes.....	2-10
2.3.1	STOP.....	2-10
2.3.2	WAIT.....	2-12
2.3.3	Data retention.....	2-12

3 MEMORY AND REGISTERS

3.1	Registers	3-1
3.2	LCD RAM	3-1
3.3	RAM.....	3-1
3.4	Programming registers	3-3
3.4.1	EEPROM programming register (EPROG).....	3-3
3.4.1.1	CPEN — Charge pump enable	3-3
3.4.1.2	ER1, ER0 — Erase select bits	3-3
3.4.1.3	LATCH — EEPROM latch control	3-4
3.4.1.4	EERC — EEPROM RC oscillator control	3-4
3.4.1.5	EEPGM — EEPROM program control	3-4
3.4.2	EPROM programming register (PCR).....	3-4
3.4.2.1	ELAT — EPROM latch control.....	3-5
3.4.2.2	PGM — EPROM program control	3-5

4 INPUT/OUTPUT PORTS

4.1	Input/output programming	4-1
4.2	Ports A and B	4-2
4.3	Port D	4-2
4.4	Port registers	4-3
4.4.1	Port data registers (PORTA, PORTB and PORTD)	4-3
4.4.2	Data direction registers (DDRA, DDRB and DDRD).....	4-3
4.4.3	Port D control register (COND)	4-4
4.4.4	Port D select register (SELD).....	4-4

5 CORE TIMER

5.1	Real time interrupts (RTI)	5-2
5.2	Computer operating properly (COP) watchdog timer	5-3
5.3	Core timer registers	5-3
5.3.1	Core timer control and status register (CTCSR)	5-3
5.3.2	Core timer counter register (CTCR).....	5-5
5.4	Core timer during WAIT	5-5
5.5	Core timer during STOP	5-5

Paragraph Number	TITLE	Page Number
---------------------	-------	----------------

6 16-BIT PROGRAMMABLE TIMER

6.1	Counter	6-3
6.1.1	Counter high register	
	Counter low register	
	Alternate counter high register	
	Alternate counter low register	6-3
6.2	Timer functions	6-4
6.2.1	Timer control registers	6-4
6.2.1.1	Timer control register 1 (TCR1).....	6-4
6.2.1.2	Timer control register 2 (TCR2).....	6-6
6.2.2	Timer status register (TSR).....	6-7
6.2.3	Input capture registers	6-9
6.2.3.1	Input capture register 1	6-9
6.2.3.2	Input capture register 2	6-10
6.2.4	Output compare registers	6-11
6.2.4.1	Output compare register 1.....	6-11
6.2.4.2	Output compare register 2.....	6-12
6.3	Timer during WAIT mode.....	6-13
6.4	Timer during STOP mode.....	6-13
6.5	Timer state diagrams	6-13

7 LIQUID CRYSTAL DISPLAY DRIVER MODULE

7.1	LCD RAM.....	7-2
7.2	LCD operation.....	7-2
7.3	Timing signals and LCD voltage waveforms	7-3
7.4	LCD control register.....	7-8
7.5	LCD during WAIT mode.....	7-8

8 I²C-BUS

8.1	I ² C-bus features.....	8-1
8.2	I ² C-bus system configuration.....	8-2
8.3	I ² C-bus protocol.....	8-2
8.3.1	START signal	8-2
8.3.2	Transmission of the slave address	8-2
8.3.3	Data transfer	8-4
8.3.4	STOP signal	8-4
8.3.5	Repeated START signal.....	8-4
8.3.6	Arbitration procedure	8-4

Paragraph Number	TITLE	Page Number
8.3.7	Clock synchronization	8-5
8.3.8	Handshaking.....	8-5
8.4	Registers	8-6
8.4.1	I ² C-bus address register (MADR)	8-6
8.4.2	I ² C-bus frequency divider register (FDR).....	8-6
8.4.3	I ² C-bus control register (MCR)	8-7
8.4.4	I ² C-bus status register (MSR).....	8-8
8.4.5	I ² C-bus data register (MDR)	8-10
8.5	Programming	8-10
8.5.1	Initialization	8-10
8.5.2	START signal and the first byte of data.....	8-10
8.5.3	Software response	8-11
8.5.4	Generation of a STOP signal	8-12
8.5.5	Generation of a repeated START signal	8-12
8.5.6	Slave mode.....	8-13
8.5.7	Arbitration lost.....	8-13
8.5.8	Operation during STOP and WAIT modes.....	8-13

9 A/D CONVERTER

9.1	A/D converter operation.....	9-1
9.2	A/D registers.....	9-3
9.2.1	A/D status/control register (ADSTAT).....	9-3
9.2.1.1	COCO — Conversion complete flag	9-3
9.2.1.2	ADRC — A/D RC oscillator control	9-3
9.2.1.3	ADON — A/D converter on	9-3
9.2.1.4	CH3 – CH0 — A/D channels 3, 2, 1 and 0.....	9-4
9.2.2	A/D input register (ADIN)	9-5
9.2.3	A/D result data register (ADDATA)	9-5
9.3	ADx analog input	9-6

10 RESETS AND INTERRUPTS

10.1	Resets	10-1
10.1.1	Power-on reset.....	10-1
10.1.2	RESET pin	10-1
10.1.3	Computer operating properly (COP) reset.....	10-2
10.2	Interrupts	10-2
10.2.1	Non-maskable software interrupt (SWI).....	10-3
10.2.2	Maskable hardware interrupts.....	10-3
10.2.2.1	External interrupt (IRQ0, IRQ1, IRQ2)	10-5
10.2.2.2	Real time and core timer (CTIMER) interrupts.....	10-7

Paragraph Number	TITLE	Page Number
10.2.2.3	Programmable 16-bit timer interrupt.....	10-7
10.2.2.4	I ² C interrupts	10-8
10.2.3	Hardware controlled interrupt sequence	10-8

11

CPU CORE AND INSTRUCTION SET

11.1	Registers.....	11-1
11.1.1	Accumulator (A)	11-1
11.1.2	Index register (X).....	11-2
11.1.3	Program counter (PC)	11-2
11.1.4	Stack pointer (SP)	11-2
11.1.5	Condition code register (CCR).....	11-2
11.2	Instruction set	11-3
11.2.1	Register/memory Instructions	11-4
11.2.2	Branch instructions	11-4
11.2.3	Bit manipulation instructions	11-4
11.2.4	Read/modify/write instructions	11-4
11.2.5	Control instructions	11-4
11.2.6	Tables.....	11-4
11.3	Addressing modes.....	11-11
11.3.1	Inherent.....	11-11
11.3.2	Immediate	11-11
11.3.3	Direct.....	11-11
11.3.4	Extended	11-12
11.3.5	Indexed, no offset.....	11-12
11.3.6	Indexed, 8-bit offset.....	11-12
11.3.7	Indexed, 16-bit offset.....	11-12
11.3.8	Relative	11-13
11.3.9	Bit set/clear	11-13
11.3.10	Bit test and branch	11-13

12

ELECTRICAL SPECIFICATIONS

12.1	Maximum ratings	12-1
12.2	Thermal characteristics and power considerations.....	12-2
12.3	DC electrical characteristics for 5V operation	12-3
12.4	AC electrical characteristics for 5V operation	12-4
12.5	A/D converter characteristics.....	12-5

13 MECHANICAL DATA

13.1	Pin configurations — 56-pin SDIP	13-1
13.2	Mechanical dimensions — 56-pin plastic shrink dual-in-line (SDIP)	13-2

14 ORDERING INFORMATION

14.1	EPROM	14-1
14.2	Verification media	14-1
14.3	ROM verification units (RVU).....	14-2

A MC68HC705L28

A.1	Features	A-1
A.2	Modes of operation.....	A-1
A.2.1	Single chip mode	A-2
A.2.2	EPROM bootloader mode.....	A-2
A.2.3	RAM bootloader mode	A-4
A.3	VPP	A-4
A.4	EPROM programming register (PCR)	A-5
A.4.1	ELAT — EPROM latch control	A-5
A.4.2	PGM — EPROM program control.....	A-5
A.5	Pin configurations — 56-pin SDIP	A-6
A.6	Ordering information.....	A-6

LIST OF FIGURES

Figure Number	TITLE	Page Number
1-1	MC68HC05L28/MC68HC705L28 block diagram	1-2
2-1	RAM bootloader circuit	2-3
2-2	MC68HC705L28 EPROM programming circuit	2-5
2-3	Oscillator connections	2-7
2-4	RC connection for external POR	2-8
2-5	STOP flowchart	2-11
2-6	WAIT flowchart	2-13
3-1	Memory map of the MC68HC05L28 and MC68HC705L28	3-2
4-1	Standard I/O port structure	4-2
5-1	Core timer block diagram	5-1
6-1	16-bit programmable timer block diagram	6-2
6-2	Timer state timing diagram for reset	6-14
6-3	Timer state timing diagram for input capture	6-14
6-4	Timer state timing diagram for output compare	6-15
6-5	Timer state timing diagram for timer overflow	6-15
7-1	LCD system block diagram	7-1
7-2	Voltage level selection	7-3
7-3	LCD waveform with 2 backplanes, 1/2 bias	7-4
7-4	LCD waveform with 2 backplanes, 1/3 bias	7-5
7-5	LCD waveform with 3 backplanes	7-6
7-6	LCD waveform with 4 backplanes	7-7
8-1	I ² C bus transmission signal diagrams	8-3
8-2	Clock synchronization	8-5
8-3	Example of a typical I ² C-bus interrupt routine	8-14
9-1	A/D converter block diagram	9-2
9-2	Electrical model of an A/D input pin	9-6
10-1	Interrupt flow chart	10-4
11-1	Programming model	11-1
11-2	Stacking order	11-2
12-1	Equivalent test load	12-2
13-1	56-pin SDIP pinout for the MC68HC05L28/MC68HC705L28	13-1
13-2	Mechanical dimensions for 56-pin SDIP	13-2
A-1	MC68HC705L28 EPROM programming circuit	A-3
A-2	56-pin SDIP pinout for the MC68HC705L28	A-6

THIS PAGE LEFT BLANK INTENTIONALLY

LIST OF TABLES

Table Number	TITLE	Page Number
2-1	MC68HC05L28 operating mode entry conditions.....	2-1
2-2	<i>MC68HC705L28 operating mode entry conditions.....</i>	<i>2-1</i>
2-3	RAM bootloader mode jump vector (MC68HC05L28).....	2-3
2-4	<i>RAM bootloader mode jump vectors (MC68HC705L28)</i>	<i>2-3</i>
3-1	Erase mode select.....	3-3
3-2	Register outline.....	3-6
4-1	I/O pin states	4-1
4-2	I/O configuration functions.....	4-4
5-1	Minimum COP reset times.....	5-3
5-2	Example RTI periods	5-4
7-1	LCD RAM organization	7-2
7-2	Multiplex ratio/backplane selection	7-8
8-1	I ² C-bus prescaler.....	8-7
9-1	A/D clock selection	9-4
9-2	A/D channel assignment.....	9-4
10-1	Interrupt priorities	10-3
10-2	IRQ1 interrupt sensitivity	10-6
10-3	IRQ2 interrupt sensitivity	10-7
11-1	MUL instruction.....	11-5
11-2	Register/memory instructions.....	11-5
11-3	Branch instructions.....	11-6
11-4	Bit manipulation instructions.....	11-6
11-5	Read/modify/write instructions	11-7
11-6	Control instructions.....	11-7
11-7	Instruction set	11-8
11-8	M68HC05 opcode map.....	11-10
12-1	Maximum ratings	12-1
12-2	Package thermal characteristics.....	12-2
12-3	DC electrical characteristics	12-3
12-4	Control timing	12-4
12-5	A/D converter characteristics.....	12-5
14-1	MC order numbers.....	14-1
A-1	MC68HC705L28 operating mode entry conditions.....	A-1
A-2	MC68HC705L28 bootloader mode jump vectors.....	A-4
A-3	MC68HC705L28 order numbers	A-6

THIS PAGE LEFT BLANK INTENTIONALLY

1

INTRODUCTION

The MC68HC05L28 is a flexible general-purpose microcomputer, particularly suited to applications throughout the consumer and automotive industries. It is a member of the highly successful Motorola M68HC05 family of microcomputers and includes many of the standard features of this family. Its hardware includes 8K of ROM, 240 bytes of EEPROM and 256 bytes of RAM, plus a multi-master I²C interface, an A/D converter and an LCD driver subsystem.

The LCD subsystem is particularly flexible and can be programmed to drive a wide range of industry-standard LCD devices — making the MC68HC05L28 particularly suited to many applications in radio, TV and compact disc systems.

The MC68HC05L28's communications, A/D and LCD modules and its two timer modules provide the perfect combination for car dashboard applications, where analog signals from speed and distance sensors have to be converted to digital signals before being processed and displayed.

The MC68HC705L28 is an EPROM equivalent version of the MC68HC05L28, with 8K of EPROM instead of 8K of ROM. All references to the MC68HC05L28 apply equally to the MC68HC705L28, unless otherwise noted. *References specific to the MC68HC705L28 are italicized in the text and also, for quick reference, they are summarised in Appendix A.*

1.1 Features

- Fully static chip design featuring the industry standard M68HC05 core
- Multi-master I²C-bus[†] communication port
- 8176 bytes of user ROM (MC68HC05L28); *8128 bytes of user EPROM (MC68HC705L28)*
- 240 bytes of EEPROM
- 240 bytes of bootstrap ROM
- 256 bytes of RAM

[†] I²C-bus is a proprietary Philips interface bus

- LCD subsystem with 18 frontplanes and 4 backplanes
- 16-bit programmable timer with 2 input capture and 2 output compare functions
- 15-stage, multifunctional core timer, with overflow, real-time interrupt and computer operating properly (COP) watchdog timer (software selectable)
- 3 interrupt request pins with edge or edge-and-level sensitive triggering (software selectable)
- 2-channel, 8-bit analog-to-digital converter
- 16 dedicated I/O lines 6 I/O lines shared with the 16-bit programmable timer and the multi-master I²C-bus interface
- Power saving STOP and WAIT modes
- Available in 56-pin SDIP package

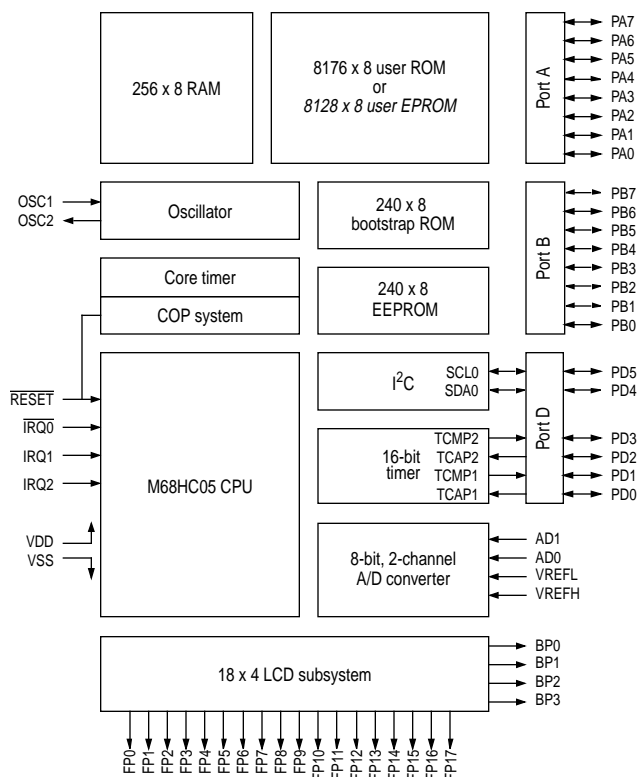


Figure 1-1 MC68HC05L28/MC68HC705L28 block diagram

1.2 Mask options on the MC68HC05L28

There is only one mask option on the MC68HC05L28. This is to enable or disable the STOP instruction. It is programmed during manufacture and must be specified on the order form.

1.2.1 Option register (OPT)

In addition to its mask option, the MC68HC05L28 also has two functions that are programmable via an options register (OPT). This register can be written to once after a reset, but it can be read at any time.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Option register (OPT)	\$001D							IRQED	COPON	??? ?100

IRQED — IRQ edge sensitivity

This bit selects the $\overline{\text{IRQ0}}$ sensitivity.

- 1 (set) — The $\overline{\text{IRQ0}}$ is edge-sensitive.
- 0 (clear) — The $\overline{\text{IRQ0}}$ is edge-and-level-sensitive.

Reset clears this bit.

COPON — COP function enable/disable

- 1 (set) — This bit enables the COP watchdog.
- 0 (clear) — This bit disables the operation of the COP.

Reset clears this bit.

THIS PAGE LEFT BLANK INTENTIONALLY

2

MODES OF OPERATION AND PIN DESCRIPTIONS

2.1 Modes of operation

The MC68HC05L28 has two modes of operation available to the user – single chip and RAM bootloader. *The MC68HC705L28 also has two modes of operation – single chip and EPROM/RAM bootloader.* Table 2-1 and Table 2-2 show the conditions required to enter each mode on the rising edge of RESET.

Table 2-1 MC68HC05L28 operating mode entry conditions

IRQ0/VPP	PB2	PB3	PB6	Mode	
V_{SS} to V_{DD}	x	x	x	Single chip	
$2 \times V_{DD}$	1	0	V_{DD}	RAM bootloader	Jump to RAM (\$81)
$2 \times V_{DD}$	1	1	V_{DD}		Load/execute RAM

Table 2-2 MC68HC705L28 operating mode entry conditions

IRQ0/VPP	PB2	PB3	PB6	Mode	
V_{SS} to V_{DD}	x	x	x	Single chip	
$2 \times V_{DD}$	0	0	V_{DD}	EPROM bootloader	Verify only
$2 \times V_{DD}$	0	1	V_{DD}		Program/verify
$2 \times V_{DD}$	1	0	V_{DD}	RAM bootloader	Jump to RAM (\$81)
$2 \times V_{DD}$	1	1	V_{DD}		Load/execute RAM

2.1.1 MC68HC05L28 modes of operation

2.1.1.1 Single chip mode

This is the normal operating mode of the MC68HC05L28 and the MC68HC705L28. In this mode the device functions as a self-contained microcomputer (MCU) with all on-board peripherals, including two 8-bit I/O ports and one 6-bit I/O port, available to the user. All address and data activity occurs within the MCU.

Single chip mode is entered on the rising edge of $\overline{\text{RESET}}$ if the voltage level on the $\overline{\text{IRQ0}}$ pin is within the normal operating range.

Warning: In the MC68HC705L28, all vectors are fetched from EPROM in single chip mode; therefore, the EPROM must be programmed (via the bootloader mode) before the device is powered up in single chip mode.

2.1.1.2 RAM bootloader mode

The RAM bootloader mode for the MC68HC05L28 and MC68HC705L28 allows the user to perform simple load and execute instructions in RAM. To make use of this feature a circuit board should be constructed as shown in Figure 2-1. Correct configuration of port pins PB2 and PB3 enables loading of a user program into RAM for execution.

The RAM bootloader is selected when the device is put into bootloader mode with PB2 held high. If PB3 is low, the program counter is set to \$0081 and a previously loaded RAM program can be executed. If PB3 is high at reset a program is serially loaded from PA0 into the RAM and executed from \$0081.

The first byte to be loaded is the count byte which should hold the program length plus the count byte. Therefore, for a program length of \$30, the count should equal \$31. The maximum program size, including the count byte is 254 bytes (\$FE), as two bytes must be left for the stack during download. The format of data for the RAM bootloader mode is 9600 baud, 8-bit, no parity, 1 start, 1stop (for 2 MHz bus speed).

In the RAM bootloader mode interrupt vectors are mapped to pseudo-vectors in RAM (for MC68HC05L28 vectors see Table 2-4 and for MC68HC705L28 vectors see Table 2-4). This allows programmers to use their own service-routine addresses. Each pseudo-vector is allowed three bytes of space, rather than the two bytes for normal vectors, because an explicit jump (JMP) opcode is needed to cause the desired jump to the user's service-routine address.

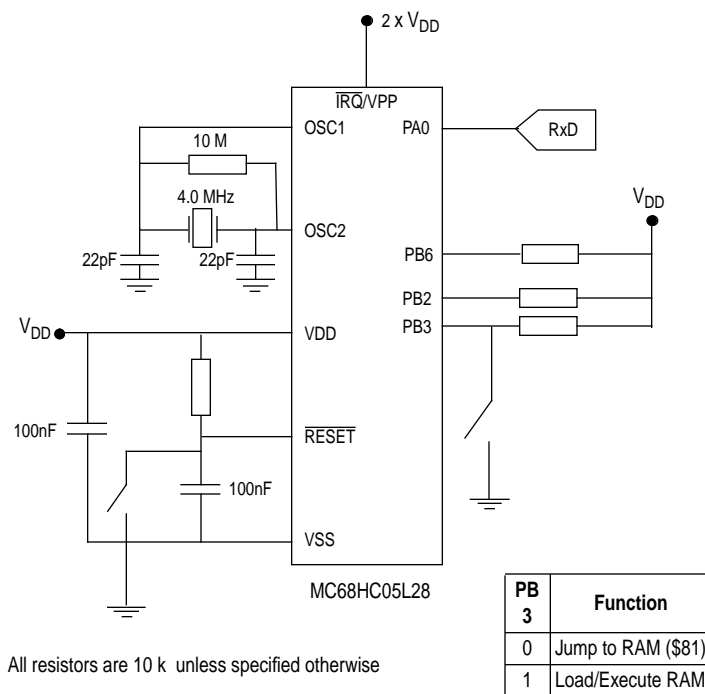


Figure 2-1 RAM bootloader circuit

Table 2-3 RAM bootloader mode jump vector (MC68HC05L28)

Address	Pseudo-vector
0084	Software interrupt

Table 2-4 RAM bootloader mode jump vectors (MC68HC705L28)

Address	Pseudo-vector
0083	Software interrupt
0086	IRQ
0089	Core timer
008C	I ² C-bus
008F	Programmable timer

2.1.2 MC68HC705L28 modes of operation

2.1.2.1 EPROM bootloader mode

This mode is used for programming the on-board EPROM. In bootloader mode the operation of the device is the same as in single chip mode, except that the vectors are fetched from a reserved area of ROM at locations \$3FE4 to \$3FEF, instead of the EPROM. The pin assignments are identical to that of single chip mode shown earlier. A recommended programming circuit is shown in Figure 2-2.

Because the addresses in the MC68HC705L28 and the EPROM containing the user code are incremented independently, it is essential that the data layout in the 27128 EPROM conforms exactly to the MC68HC705L28 memory map. The bootloader uses an external 14-bit counter to address the memory device containing the code to be copied. This counter requires a clock and a reset function which are provided by the MC68HC705L28.

In this mode all interrupt vectors are mapped to pseudo-vectors in RAM (see Table 2-4). This allows programmers to use their own service routine addresses. Each pseudo-vector is allowed three bytes of space, rather than the two bytes for normal vectors, because an explicit jump (JMP) opcode is needed to cause the desired jump to the user's service routine address.

The bootloader code deals with the copying of user code from an external EPROM into the on-chip EPROM. The bootloader function can only be used with an external EPROM. The bootloader performs a programming pass followed by a verify pass.

Pins PB2 and PB3 are used to select various bootloader functions, including the programming mode (see Figure 2-2). Two other pins, PB1 and PB6 are used to drive the PROG and VERF LED outputs. While the EPROM is being programmed the PROG LED lights up; when programming is complete the internal EPROM contents are compared to that of the external EPROM and, if they match exactly, the VERF LED lights up. When finished programming, the PROG LED turns off. If the MC68HC705L28 memory contents are the same as the EPROM the VERF LED lights up, otherwise no LEDs are turned on.

Note: The EPROM must be erased before performing a program cycle.

2.1.2.2 RAM bootloader mode

See Section 2.1.1.2.

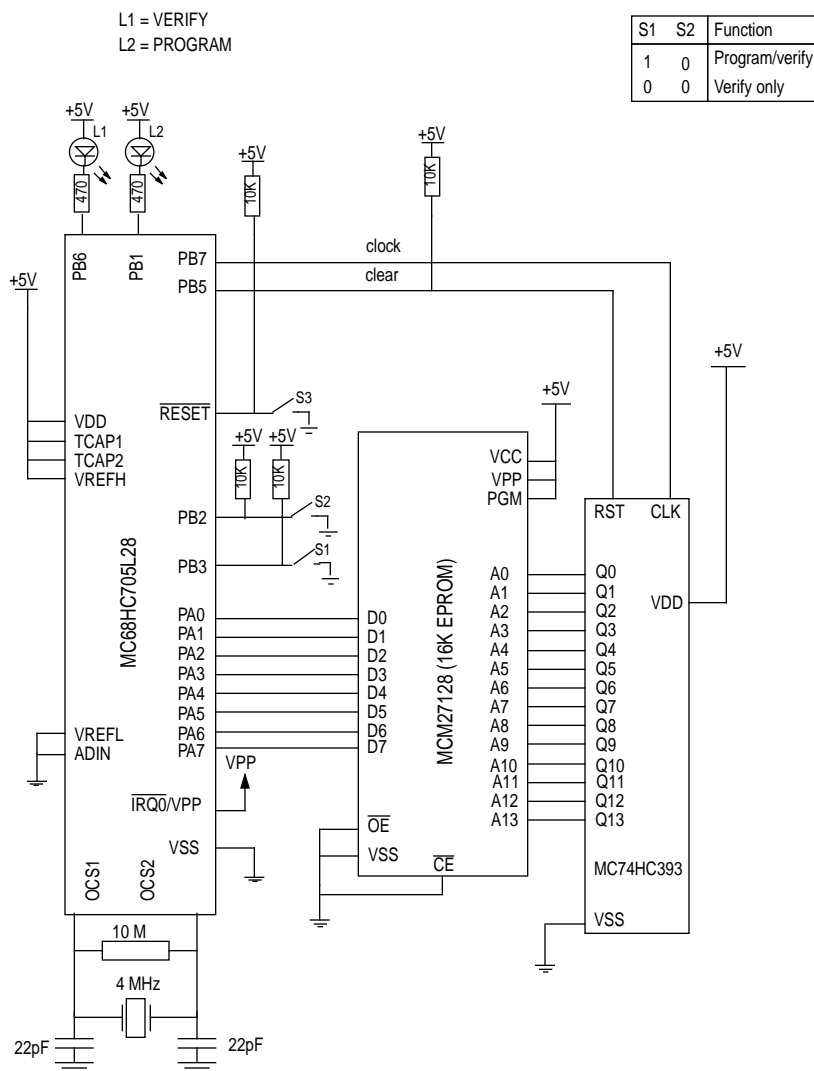


Figure 2-2 MC68HC705L28 EPROM programming circuit

2.2 Pin descriptions

Pin assignments are shown in Section 13.

2.2.1 VDD and VSS

Power is supplied to the microcontroller using these two pins. VDD is the positive supply and VSS is ground.

It is in the nature of CMOS designs that very fast signal transitions occur on the MCU pins. These short rise and fall times place very high short-duration current demands on the power supply. To prevent noise problems, special care must be taken to provide good power supply bypassing at the MCU. Bypass capacitors should have good high-frequency characteristics and be as close to the MCU as possible. Bypassing requirements vary, depending on how heavily the MCU pins are loaded.

2.2.2 $\overline{\text{IRQ0}}$

This interrupt has a software option which offers two types of interrupt triggering sensitivity. It contains an internal Schmitt trigger as part of its input to improve noise immunity.

2.2.3 IRQ1

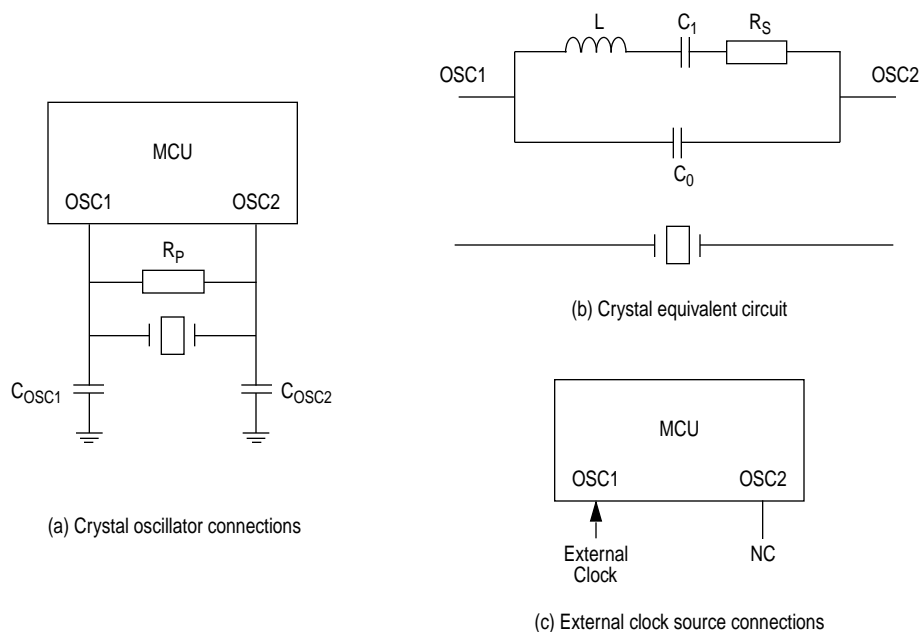
This interrupt has a software option which offers four types of interrupt triggering sensitivity. It contains an internal Schmitt trigger as part of its input to improve noise immunity. This interrupt can be enabled/disabled independently.

2.2.4 IRQ2

This interrupt has a software option which offers four types of interrupt triggering sensitivity. It contains an internal Schmitt trigger as part of its input to improve noise immunity. This interrupt can be enabled/disabled independently.

2.2.5 OSC1, OSC2

These pins provide control input for an on-chip oscillator circuit. A crystal connected to these pins supplies the oscillator clock. The oscillator frequency (f_{OSC}) is divided by two to give the internal bus frequency (f_{OP}).



Crystal			
	2MHz	4MHz	Unit
$R_S(\text{max})$	400	75	
C_0	5	7	pF
C_1	8	12	nF
C_{OSC1}	15 – 40	15 – 30	pF
C_{OSC2}	15 – 30	15 – 25	pF
R_P	10	10	M
Q	30 000	40 000	—

(d) Crystal parameters

Figure 2-3 Oscillator connections

2.2.5.1 Crystal

The circuit shown in Figure 2-3(a) is recommended when using a crystal. Figure 2-3(d) lists the recommended capacitance and feedback resistance values. The internal oscillator is designed to interface with an AT-cut parallel-resonant quartz crystal resonator in the frequency range specified for f_{OSC} (see Section 12.4). Use of an external CMOS oscillator is recommended when crystals outside the specified ranges are to be used. The crystal and associated components should be mounted as close as possible to the input pins to minimize output distortion and start-up stabilization time. The manufacturer of the particular crystal being considered should be consulted for specific information.

2.2.5.2 External clock

An external clock should be applied to the OSC1 input, with the OSC2 pin left unconnected, as shown in Figure 2-3(c). The t_{OXOV} specification (see Section 12.4) does not apply when using an external clock input. The equivalent specification of the external clock source should be used in lieu of t_{OXOV} .

2.2.6 RESET

This active low input pin is used to reset the MCU. Applying a logic zero to this pin forces the device to a known start-up state. This input has an internal Schmitt trigger to improve noise immunity.

The MCU has its own internal power-on reset (POR) circuit. If, however, the user requires an additional POR, then an RC network can be connected to this pin as shown in Figure 2-4. The user must ensure that the RC time constant of this network is greater than the oscillator stabilization period. A time constant of at least 100 ms is recommended.



Figure 2-4 RC connection for external POR

2.2.7 PA0 – PA7, PB0 – PB7

These 16 I/O lines comprise ports A and B. The state of any pin is software programmable, and all the pins are configured as inputs during power-on or reset. Port B pins in output mode can sink 10mA of current to drive the LEDs.

2.2.8 PD0 – PD5

These six I/O lines comprise port D. The state of any pin is software programmable, and all the pins are configured as inputs during power-on or reset. This port shares its pins with the 16-bit timer and I²C subsystems. There are four read/write registers associated with the port to select the different functions. All the port bits can be configured as input/output pins or used by other systems within the MCU.

This 6-bit I/O port shares its pins with other subsystems on the MCU and is controlled using the port D control (COND) and select (SELD) registers. On reset, all registers except the data register are cleared thereby configuring all port pins as normal inputs with pull-up resistors. Writing a '1' to any bit in COND connects the subsystem function to the corresponding port D pin.

2.2.9 BP0 – BP3, FP0 – FP17

These signals comprise the LCD driver subsystem. The subsystem has a maximum of 18 frontplanes and four backplanes.

2.2.10 AD0 – AD1, VREFH/VREFL

These 4 signals comprise the A-to-D interface. The subsystem has a maximum of two A/D channels.

2.3 Low power modes

2.3.1 STOP

The STOP instruction places the MCU in its lowest power consumption mode. In STOP mode, the internal oscillator is turned off, halting all internal processing, including timer (and COP watchdog timer) operation.

During the STOP mode, the core timer interrupt flags (CTOF and RTIF) and interrupt enable bits (CTOFE and RTIE) in the CTCSR are cleared by internal hardware. This removes any pending timer interrupt requests and disables any further timer interrupts. The timer prescaler is also cleared. The I-bit in the CCR is cleared to enable external interrupts. All other registers, the remaining bits in the CTCSR, and memory contents remain unaltered. All input/output lines remain unchanged. The processor can be brought out of the STOP mode only by an external interrupt or a reset (see Figure 2-5).

The STOP instruction can be disabled by a mask option. When disabled, it is executed as a NOP.

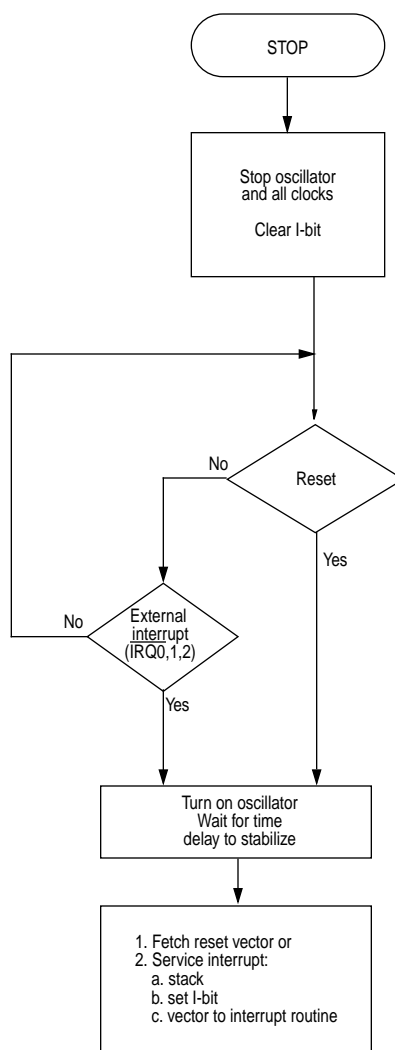


Figure 2-5 STOP flowchart

2.3.2 WAIT

The WAIT instruction places the MCU in a low power consumption mode, but the WAIT mode consumes more power than the STOP mode. All CPU action is suspended, but the core timer, 16-bit timer and I²C remains active. An interrupt from the core timer, 16-bit timer or I²C (if enabled) will cause the MCU to exit WAIT mode.

During WAIT mode, the I-bit in the CCR is cleared to enable interrupts. All other registers, memory and input/output lines remain in their previous state. The core timer may be enabled to allow a periodic exit from the WAIT mode (see Figure 2-6).

2.3.3 Data retention

The contents of the RAM are retained at supply voltages as low as 2.0Vdc. This is called the data retention mode, in which data is maintained but the device is not guaranteed to operate.

For lowest power consumption in data retention mode the device should be put into STOP mode before reducing the supply voltage, to ensure that all the clocks are stopped. If the device is not in STOP mode then it is recommended that RESET be held low while the power supply is outside the normal operating range, to ensure that processing is suspended in an orderly manner.

Recovery from data retention mode, after the power supply has been restored, is by pulling the RESET line high.

To put the MCU into data retention mode:

- Set RESET pin to zero
- Reduce the voltage on VDD. RESET must remain low during data retention mode

To take the MCU out of data retention:

- Return VDD to normal operating level
- Return RESET to logical one.

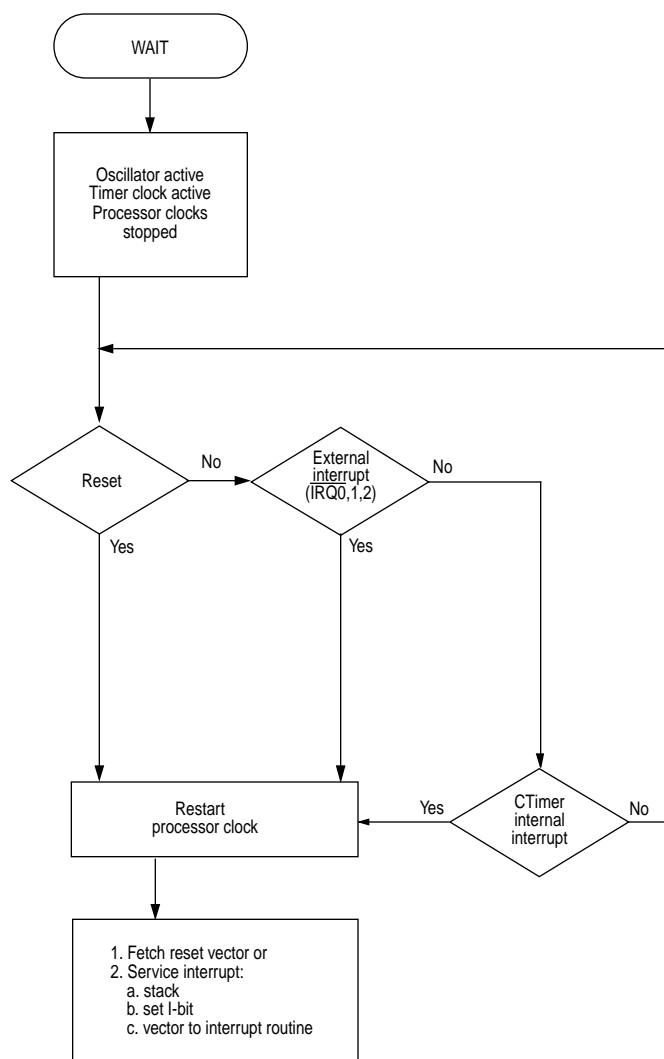


Figure 2-6 WAIT flowchart

THIS PAGE LEFT BLANK INTENTIONALLY

MEMORY AND REGISTERS

The MC68HC05L28 has a 16K byte memory map consisting of registers, user ROM, user RAM, bootstrap ROM, LCD RAM, EEPROM and I/O, as shown in Figure 3-1.

3.1 Registers

All the I/O, control and status registers of the MC68HC05L28 are contained within the first 64-byte block of the memory map (address \$0000 to \$003F).

3.2 LCD RAM

The 12 bytes of LCD RAM are located at address \$0040 to \$004B. This RAM is used to store the data needed for the LCD. See Section 7.1 for further details of the organization of these bits.

3.3 RAM

The user RAM consists of 256 bytes of memory, from \$0080 to \$017F. This is shared with a 64 byte stack area. The stack begins at \$00FF and counts down to \$00C0.

Note: Using the stack area for data storage or temporary work locations requires care to prevent the data from being overwritten due to stacking from an interrupt or subroutine call.

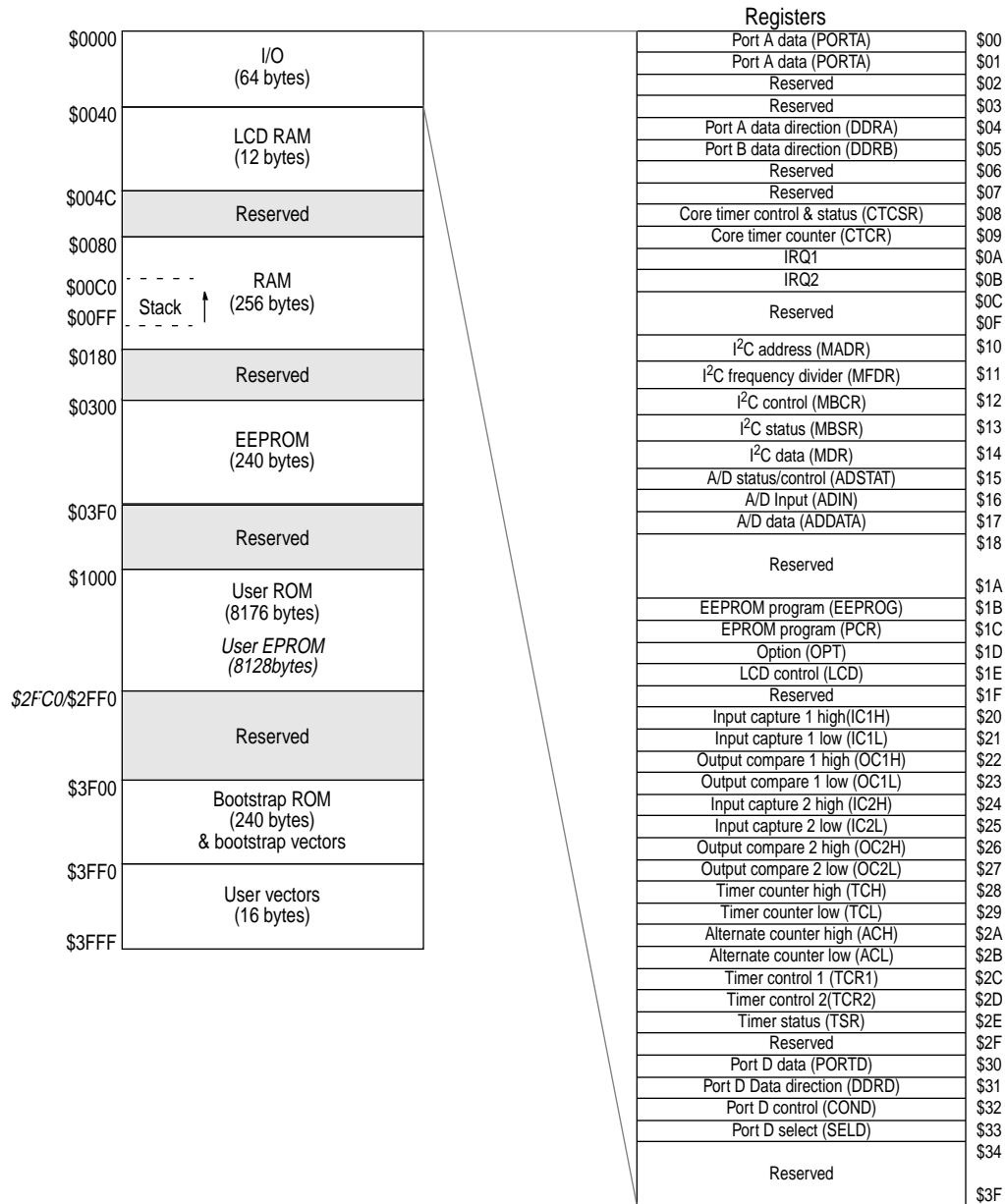


Figure 3-1 Memory map of the MC68HC05L28 and MC68HC705L28

3.4 Programming registers

3.4.1 EEPROM programming register (EEPROM)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
EEPROM program (EEPROM)	\$001B		CPEN	0	ER1	ER0	LATCH	EERC	EEPGM	?000 0000

3.4.1.1 CPEN — Charge pump enable

This bit enables the charge pump which produces the internal programming voltage. It is set with the LATCH bit and should be disabled when not in use. The programming voltage is not available until EEPGM is set.

- 1 (set) — Charge pump is enabled.
- 0 (clear) — Charge pump is disabled.

3.4.1.2 ER1, ER0 — Erase select bits

ER1 and ER0 are used to select either single byte programming or one of three erase modes: byte, block or bulk. Table 3-1 shows the modes selected for each bit configuration. These bits are readable and writeable and are cleared by reset.

Table 3-1 Erase mode select

ER1	ER0	Mode
0	0	Program
0	1	Byte erase
1	0	Block erase
1	1	Bulk erase

- In byte erase mode only the selected byte is erased.
- In block erase mode a 64-byte block of EEPROM is erased. The EEPROM memory space is divided into four 64-byte blocks (\$0300–\$033F, \$0340–\$037F, \$0380–\$03BF, \$03C0–\$03EF) and an erase to any address within a block erases that block.
- In bulk erase mode the entire 240 bytes of EEPROM are erased.

3.4.1.3 LATCH — EEPROM latch control

- 1 (set) – The EEPROM address and data buses are configured for programming. This causes the address and data buses to be latched when a write to EEPROM is carried out. EEPROM cannot be read if LATCH = 1.
- 0 (clear) – The EEPROM address and data buses are configured for normal reads.

3.4.1.4 EERC — EEPROM RC oscillator control

- 1 (set) – The EEPROM uses an internal RC oscillator instead of the CPU clock. After setting, wait for time t_{RCON} to allow the RC oscillator to stabilize. It should be set by the user when the internal bus frequency falls below 1.5 MHz.
- 0 (clear) – The EEPROM uses the CPU clock.

3.4.1.5 EEPGM — EEPROM program control

- 1 (set) – Programming power is connected to the EEPROM array. EEPGM can only be set if LATCH is set and is automatically cleared when LATCH = 0.
- 0 (clear) – Programming power is disconnected from the EEPROM array.

Note: LATCH and EEPGM cannot be set on the same write operation

3.4.2 EPROM programming register (PCR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
EPROM program (PCR)	\$001C							ELAT	PGM	uuuu uu00

3.4.2.1 ELAT — EPROM latch control

- 1 (set) — When this bit is set to 1, the EPROM address and data buses are configured for programming. This causes the address and data buses to be latched when a write to EPROM is executed. The EPROM cannot be read if it is set to 1.
- 0 (clear) — The EPROM address and data buses are configured for normal reads.

3.4.2.2 PGM — EPROM program control

- 1 (set) — Programming power is connected to the EPROM array. The PGM can only be set if ELAT is set and it is automatically cleared when $ELAT = 0$.
- 0 (clear) — EPROM address and data bus are configured for normal reads.

Note: ELAT and PGM cannot be set in the same write operation.

Take the following steps to program a byte of EPROM:

- Apply the programming voltage V_{PP} to the VPP pin.
- Set the ELAT bit.
- Write to the EPROM address.
- Set the PGM bit for a time t_{PROG} to apply the programming voltage.
- Clear the ELAT and PGM bits.

Note: The erased state of the EPROM is \$00.

Table 3-2 Register outline

Register Name	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on Reset
Port A data (PORTA)	\$0000									unaffected
Port B data (PORTB)	\$0001									unaffected
Reserved	\$0002									
Reserved	\$0003									
Port A data direction (DDRA)	\$0004	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	0000 0000
Port B data direction (DDRB)	\$0005	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	0000 0000
Reserved	\$0006									
Reserved	\$0007									
Core timer control/status (CTCSR)	\$0008	CTOF	RTIF	CTOFE	RTIE			RT1	RT0	0000 0011
Core timer counter (CTCR)	\$0009									0000 0000
IRQ1	\$000A			IRQ1INT	IRQ1ENA	IRQ1LV	IRQ1EDG	IRQ1RST	IRQ1VAL	???0 000?
IRQ2	\$000B			IRQ2INT	IRQ2ENA	IRQ2LV	IRQ2EDG	IRQ2RST	IRQ2VAL	???0 000?
Reserved	\$000C									unaffected
Reserved	\$000D									unaffected
Reserved	\$000E									unaffected
Reserved	\$000F									unaffected
I ² C address (MADR)	\$0010	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1		0000 000?
I ² C frequency divide (FDR)	\$0011				MBC4	MBC3	MBC2	MBC1	MBC0	???0 0000
I ² C control (MCR)	\$0012	MEN	MIEN	MSTA	MTX	TXAK	MMUX			0000 00??
I ² C status (MSR)	\$0013	MCF	MAAS	MBB	MAL		SRW	MIF	RXAK	1000 ?001
I ² C data (MDR)	\$0014	TRXD7	TRXD6	TRXD5	TRXD4	TRXD3	TRXD2	TRXD1	TRXD0	uuuu uuuu
A/D status control (ADSTAT)	\$0015	COCO	ADRC	ADON		CH3	CH2	CH1	CH0	0000 0000
A/D input (ADIN)	\$0016							AD1	AD0	uuuu uuuu
A/D data (ADDATA)	\$0017									uuuu uuuu
Reserved	\$0018									
Reserved	\$0019									
Reserved	\$001A									
EEPROM program (EEPROG)	\$001B		CPEN		ER1	ER0	LATCH	EERC	EEPGM	?000 0000
EPROM program (PCR) ⁽¹⁾	\$001C							ELAT	PGM	uuuu uu00
Option (OPT)	\$001D							IRQED	COPON	???? ?100
LCD control (LCD)	\$001E		VLCDON			FDISP	MUX4	MUX3	DISON	?000 0000
Reserved	\$001F									
Input capture 1 high (IC1H)	\$0020									uuuu uuuu
Input capture 1 low (IC1L)	\$0021									uuuu uuuu
Output compare 1 high (OC1H)	\$0022									uuuu uuuu
Output compare 1 low (OC1L)	\$0023									uuuu uuuu
Input capture 2 high (IC2H)	\$0024									uuuu uuuu
Input capture 2 low (IC2L)	\$0025									uuuu uuuu
Output compare 2 high (OC2H)	\$0026									uuuu uuuu
Output compare 2 low (OC2L)	\$0027									uuuu uuuu

Table 3-2 Register outline

Register Name	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on Reset
Timer counter high (TCH)	\$0028									1111 1111
Timer counter low (TCL)	\$0029									1111 1100
Alternate counter high (ACH)	\$002A									1111 1111
Alternate counter low (ACL)	\$002B									1111 0011
Timer control 1 (TCR1)	\$002C	IC1IE	IC2IE	OC1IE	TOIE	CO1E	IEDG1	IEDG2	OLV1	0000 0uu0
Timer control 2 (TCR2)	\$002D			OC2IE		CO2E			OLV2	0000 0000
Timer status (TSR)	\$002E	IC1F	IC2F	OC1F	TOF	TCAP1	TCAP2	OC2F		uuuu 11u0
Reserved	\$002F									
Port D data (PORTD)	\$0030									unaffected
Port D data direction (DDRD)	\$0031			DDR5	DDR4	DDR3	DDR2	DDR1	DDR0	??00 0000
Port D control (COND)	\$0032			COND5	COND4	COND3	COND2	COND1	COND0	??00 0000
Port D select (SELD)	\$0033			PD5/ SCL0	PD4/ SDA0	PD3/ TCMP2	PD2/ TCAP2	PD1/ TCMP1	PD0/ TCAP1	??00 0000
Reserved	\$34-\$3F									

(1) MC68HC705L28 only

THIS PAGE LEFT BLANK INTENTIONALLY

4

INPUT/OUTPUT PORTS

4

In single chip mode, the MC68HC05L28 has a total of 22 I/O lines, arranged as two 8-bit ports (A and B) and one 6-bit port (D). Each I/O line is individually programmable as either input or output, under the software control of the data direction registers. Port D shares various I/O configurations with the timer and I²C subsystems.

To avoid glitches on the output pins, data should be written to the I/O port data register before writing ones to the corresponding data direction register bits to set the pins to output mode.

4.1 Input/output programming

The bidirectional port lines may be programmed as inputs or outputs under software control. The direction of each pin is normally determined by the state of the corresponding bit in the port data direction register (DDR). Each port has an associated DDR. Any standard I/O port pin is configured as an output if its corresponding DDR bit is set to a logic one. A pin is configured as an input if its corresponding DDR bit is cleared.

At power-on or reset, all DDRs are cleared, configuring all port pins as inputs. The data direction registers can be written to or read by the MCU. During the programmed output state, a read of the data register actually reads the value of the output data latch and not the I/O pin. The operation of the standard port hardware is shown schematically in Figure 4-1.

This is summarized in Table 4-1, which shows the effect of reading from or writing to an I/O pin in various circumstances. Note that the read/write signal shown is internal and not available to the user.

Table 4-1 I/O pin states

R/W	DDRn	Action of MCU write to/read of data bit
0	0	The I/O pin is in input mode. Data is written into the output data latch.
0	1	Data is written into the output data latch, and output to the I/O pin.
1	0	The state of the I/O pin is read.
1	1	The I/O pin is in output mode. The output data latch is read.

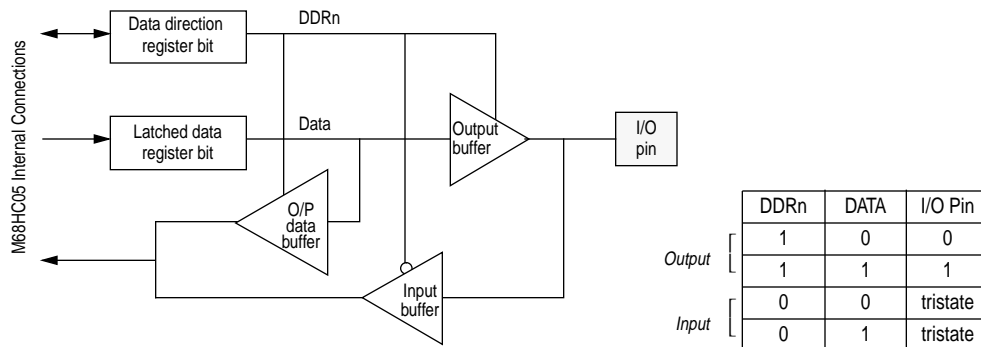


Figure 4-1 Standard I/O port structure

4.2 Ports A and B

These ports are standard M68HC05 bidirectional I/O ports, each comprising a data register and a data direction register.

Reset does not affect the state of the data register, but clears the data direction register, thereby returning all port pins to input mode. Writing a '1' to any DDR bit sets the corresponding port pin to output mode.

4.3 Port D

Port D is a 6-bit non-standard port which shares its pins with the timer and I²C subsystems. There are four read/write registers associated with the port for defining the different functions. All the port D pins can be configured as input/output pins or can be used by other subsystems within the MCU. Setting bits 5-0 in the port D select register to logical '1' configures the pin as dedicated to the timer or I²C subsystems. For details of the alternative function of each port D pin see Section 2.2.8.

4.4 Port registers

The following sections explain in detail the individual bits in the data and control registers associated with the ports.

4.4.1 Port data registers (PORTA, PORTB and PORTD)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port A data (PORTA)	\$0000									Unaffected
Port B data (PORTB)	\$0001									Unaffected
Port D data (PORTD)	\$0030									Unaffected

Each bit can be configured as input or output via the corresponding data direction bit in the port data direction register (DDRx).

Reset does not affect the state of the port data registers.

Each of the port D bits is shared with another MCU subsystem. The configuration of this register is determined by the setting of individual bits in the port D control register. See Section 4.4.3.

4.4.2 Data direction registers (DDRA, DDRB and DDRD)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port A data direction (DDRA)	\$0004									0000 0000
Port B data direction (DDRB)	\$0005									0000 0000
Port D data direction (DDRD)	\$0031									0000 0000

Writing a '1' to any bit configures the corresponding port pin as an output; conversely, writing any bit to '0' configures the corresponding port pin as an input.

Reset clears these registers, thus configuring all ports as inputs.

4.4.3 Port D control register (COND)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port D control (COND)	\$0032									0000 0000

The select register, data direction register and control register determine the function of the I/O port, as shown in Table 4-2.

Table 4-2 I/O configuration functions

DDRD	COND	Function
0	0	Input with pull-up
0	1	Input without pull-up
1	0	Push-pull output
1	1	Open-drain output without pull-up

4.4.4 Port D select register (SELD)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port D select (SELD)	\$0033			PD5/ SCL0	PD4/ SDA0	PD3/ TCMP2	PD2/ TCAP2	PD1/ TCMP1	PD0/ TCAP1	0000 0000

Setting bits 5-0 in the port D select register to logical '1' configures the pin to be dedicated to the timer or the I²C bus subsystems. This select bit overrides the effect that the DDR register has on the port direction. The user must ensure that the DDR and COND register bits are programmed correctly to obtain the desired pin configuration.

PD5/SCL0 — Port D pin 5/SCL0 select

- 1 (set) — This pin is configured as the I²C clock and is always an open-drain I/O. If a pull-up is required then bit 5 in the DDRD and COND registers must be cleared.
- 0 (clear) — This pin is configured as I/O pin PD5.

PD4/SDA0 — Port D pin 4/SCL1 select

- 1 (set) — This pin is configured as the I²C data pin and is always an open-drain I/O. If a pull-up is required then bit 4 in the DDRD and COND registers must be cleared.
- 0 (clear) — This pin is configured as I/O pin PD4.

PD3/TCMP2 — Port D pin 3/TCMP2 select

- 1 (set) — This pin is configured as timer output compare 2 output. Setting bit 3 in the COND register makes it an open drain output
- 0 (clear) — This pin is configured as I/O pin PD3.

PD2/TCAP2 — Port D pin 2/TCAP2 select

- 1 (set) — This pin is configured as timer input capture 2 input. Clearing bit 2 in the COND register enables the pull-up resistor.
- 0 (clear) — This pin is configured as I/O pin PD2.

PD1/TCMP1 — Port D pin 1/TCMP1 select

- 1 (set) — This pin is configured as Timer output compare 1 output. Setting bit 1 in the COND register makes it an open drain output
- 0 (clear) — This pin is configured as I/O pin PD1.

PD0/TCAP1 — Port D pin 0/TCAP1 select

- 1 (set) — This pin is configured as timer input capture 1 input. Clearing bit 0 in the COND register enables the pull-up resistor.
- 0 (clear) — This pin is configured as I/O pin PD0.

THIS PAGE LEFT BLANK INTENTIONALLY

5

CORE TIMER

The MC68HC05L28 has a 15-stage ripple counter called the core timer (CTIMER). Features of this timer are: timer overflow, power-on reset (POR), real time interrupt (RTI) with four selectable interrupt rates, and a computer operating properly (COP) watchdog timer.

5

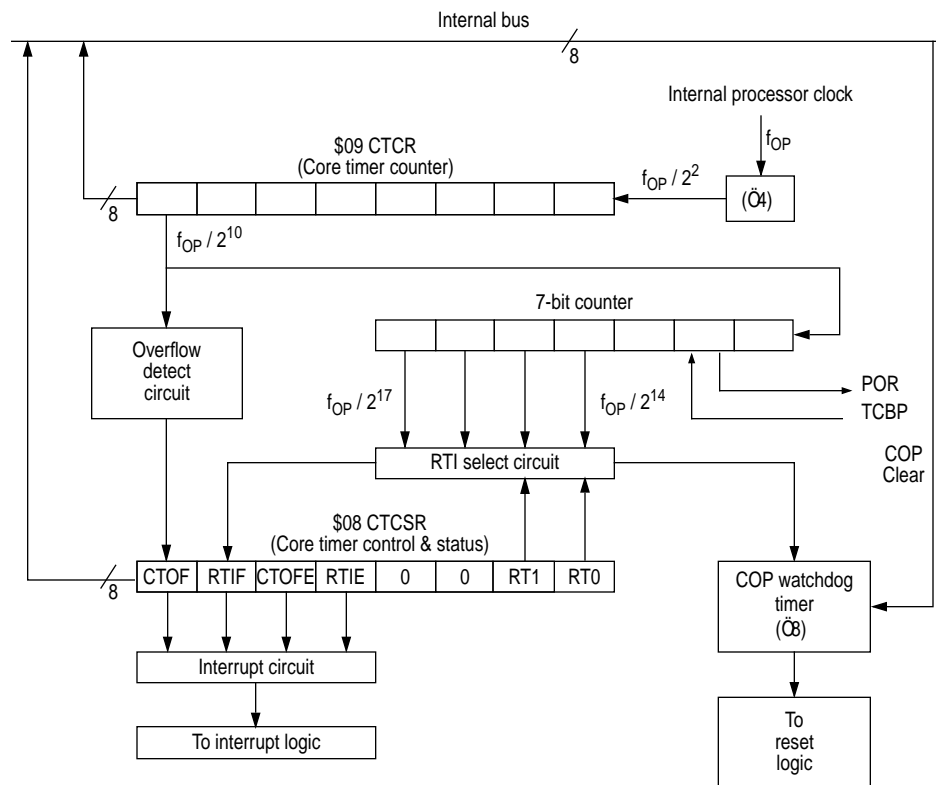


Figure 5-1 Core timer block diagram

As shown in Figure 5-1, the timer is driven by the internal bus clock divided by four with a fixed prescaler. This signal 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 CTIMER counter register (CTCR) at address \$09. A timer overflow function is implemented on the last stage of this counter, giving a possible interrupt at the rate of $f_{OP}/1024$. The POR signal (t_{PORL}) is also derived from this register, at $f_{OP}/4064$. The counter register circuit is followed by two more stages, with the resulting clock ($f_{OP}/16384$) driving the real time interrupt 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 COP watchdog timer circuit. The RTI rate selector bits and the RTI and CTOF enable bits and flags are located in the CTIMER control and status register (CTCSR) at location \$08.

CTOF (core timer overflow flag) is a clearable, read-only status bit set when the 8-bit ripple counter rolls over from \$FF to \$00. A CPU interrupt request will be generated if CTOFE is set. CTOF is cleared by writing a '0' to it. Writing a '1' has no effect. Reset clears this bit.

When CTOFE (core timer overflow flag enable) is set, a CPU interrupt request is generated when the CTOF bit is set. Reset clears CTOFE.

The core timer counter register (CTCR) is a read-only register that contains the current value of the 8-bit ripple counter at the beginning of the timer chain. This counter is clocked at $f_{OP}/4$ and can be used for various functions including a software input capture. Extended time periods can be attained using the CTOF function to increment a temporary RAM storage location simulating a 16-bit (or more) counter.

The power-on cycle clears the entire counter chain and begins clocking the counter. After t_{PORL} 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{RESET} is not asserted, the timer starts counting up from zero and normal device operation begins. When \overline{RESET} is asserted at any time during operation (other than POR), the counter chain is cleared. See Section 5.3 for register details.

5.1 Real time interrupts (RTI)

The real time interrupt circuit consists of a three stage divider and a 1-of-4 selector. The clock frequency that drives the RTI circuit is $f_{OP}/2^{13}$ (or $f_{OP}/8192$), with three additional divider stages, giving a maximum interrupt period of 4 seconds at a crystal frequency (f_{OP}) of 32kHz.

The flag (RTIF) is a clearable, read-only status bit which is set when the output of the chosen (1-of-4 selection) stage becomes active. A CPU interrupt request is generated if RTIE is set. RTIF is cleared by writing a '0' to it. Writing a '1' has no effect. Reset clears this bit. See Section 5.3 for register details.

5.2 Computer operating properly (COP) watchdog timer

The COP watchdog timer is implemented by dividing the output of the RTI circuit by eight, as shown in Figure 5-1. The minimum COP timeout period is seven times the RTI period. This is because the COP will be cleared asynchronously with respect to the value in the core timer counter register/RTI divider, hence the actual COP timeout period will vary between 7x and 8x the RTI period. See Table 5-1.

The COP function is enabled by programming the COPON bit in the option register (OPT). See Section 1.2.1.

If the COP circuit times out, an internal reset is generated and the normal reset vector is fetched. Writing a '0' to bit 0 of address \$0FF0 prevents a COP timeout. When the COP is cleared, only the final divide-by-eight stage is cleared (see Figure 5-1). See Section 5.3 for register details.

Table 5-1 Minimum COP reset times

RT1, RT0	Minimum COP reset at bus frequency:		
	16.384 kHz	4.194 MHz	f_{OP}
00	7 s	53.2 ms	7 x (RTI rate)
01	17s	105.7 ms	7 x (RTI rate)
10	28s	765.8 ms	7 x (RTI rate)
11	56 s	422.8 ms	7 x (RTI rate)

5.3 Core timer registers

5.3.1 Core timer control and status register (CTCSR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Core timer control/status (CTCSR)	\$0008	CTOF	RTIF	CTOFE	RTIE	0	0	RT1	RT0	0000 0011

CTOF — Core timer overflow

- 1 (set) — The core timer has overflowed.
- 0 (clear) — The core timer has not overflowed.

This bit is set when the core timer counter register rolls over from \$FF to \$00; an interrupt request will be generated if CTOFE is set. When set, the bit may be cleared by writing a '0' to it.

RTIF — Real time interrupt flag

- 1 (set) — A real time interrupt has occurred.
- 0 (clear) — No real time interrupt has been generated.

This bit is set when the output of the chosen stage becomes active; an interrupt request will be generated if RTIE is set. When set, the bit may be cleared by writing a '0' to it.

CTOFE — Core timer overflow enable

- 1 (set) — Core timer overflow interrupt is enabled.
- 0 (clear) — Core timer overflow interrupt is disabled.

Setting this bit enables the core timer overflow interrupt. A CPU interrupt request is generated when the CTOF bit is set. Clearing this bit disables the core timer overflow interrupt capability.

RTIE — Real time interrupt enable

- 1 (set) — Real time interrupt is enabled.
- 0 (clear) — Real time interrupt is disabled.

Setting this bit enables the real time interrupt. A CPU interrupt request is generated when the RTIF bit is set. Clearing this bit disables the real time interrupt capability.

RT1, RT0 — Real time interrupt rate select

These two bits select one of four taps from the real time interrupt circuitry. Reset sets both RT0 and RT1 to one, selecting the lowest periodic rate and therefore the maximum time in which to alter them if necessary. The COP reset times are also determined by these two bits. Care should be taken when altering RT0 and RT1 if a timeout is imminent or the timeout period is 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 the RTI taps. See Table 5-2 for some example RTI periods.

Table 5-2 Example RTI periods

			Bus frequency $f_{OP} = 2 \text{ MHz}$	
RT1	RT0	Division ratio	RTI period	Minimum COP period
0	0	2^{14}	8.2ms	57.3ms
0	1	2^{15}	16.4ms	114.7ms
1	0	2^{16}	32.8ms	229.4ms
1	1	2^{17}	65.5ms	458.8ms

5.3.2 Core timer counter register (CTCR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Core timer counter (CTCR)	\$0009									0000 0000

The core timer counter register is a read-only register, which contains the current value of the 8-bit ripple counter at the beginning of the timer chain.

Reset clears this register.

5.4 Core timer during WAIT

The CPU clock halts during the WAIT mode, but the core timer remains active. If the interrupts are enabled, then a CTIMER interrupt will cause the processor to exit the WAIT mode.

5.5 Core timer during STOP

The timer is cleared when going into STOP mode. When STOP is exited by an external interrupt or an external reset, the internal oscillator will restart, followed by an internal processor stabilization delay (t_{PORL}). The timer is then cleared and operation resumes.

THIS PAGE LEFT BLANK INTENTIONALLY

6

16-BIT PROGRAMMABLE TIMER

The MC68HC05L28 has a 16-bit programmable timer. This timer consists of a 16-bit read-only free-running counter, with a fixed divide-by-four prescaler, plus input capture/output compare circuitry.

Selected input edges cause the current counter value to be latched into a 16-bit input capture register so that software can later read this value to determine when the edge occurred. When the free running counter value matches the value in the output compare registers, the programmed pin action takes place.

6

See Figure 6-1 for a block diagram of the timer.

As the timer has a 16-bit architecture, each segment is represented by two 8-bit registers. These registers contain the high and low byte of that functional segment. Accessing the low byte of a specific timer function allows full control of that function, while accessing the high byte inhibits that specific timer function until the low byte is also accessed.

Note: The I-bit in the CCR should be set, while manipulating both the high and low byte register of a specific timer function, to ensure that an interrupt does not occur.

6

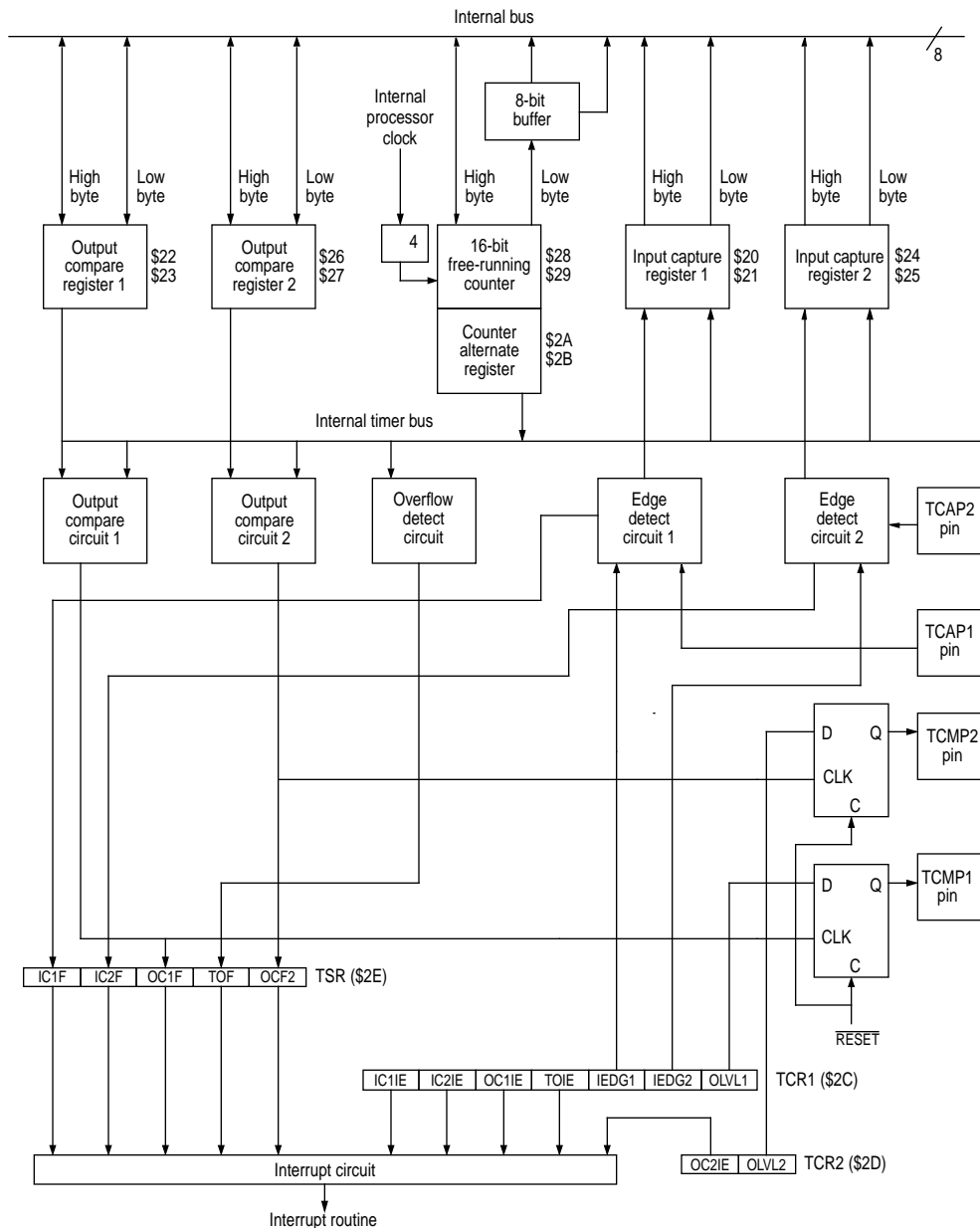


Figure 6-1 16-bit programmable timer block diagram

6.1 Counter

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 μ s if the internal bus clock is 2MHz. 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.

6.1.1 Counter high register Counter low register Alternate counter high register Alternate counter low register

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Timer counter high (TCH)	\$0028	(bit 15)							(bit 8)	\$FF
Timer counter low (TCL)	\$0029									\$FC
Alternate counter high (ACH)	\$002A	(bit 15)							(bit 8)	\$FF
Alternate counter low (ACL)	\$002B									\$FC

The double-byte, free-running counter can be read from either of two locations: the counter register at \$28-\$29 or the alternate counter register at \$2A-\$2B. A read from only the less significant byte (LSB) of the free-running counter, \$29 or \$2B, receives the count value at the time of the read. If a read of the free-running counter or alternate counter register first addresses the more significant byte (MSB), \$28 or \$2A, the LSB 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 alternate counter register LSB and thus completes a read sequence of the total counter value. In reading either the free-running counter or alternate counter register, if the MSB is read, the LSB must also be read to complete the sequence.

The alternate counter register differs from the counter register only in that a read of the MSB does not clear TOF. Therefore the counter alternate register can be read at any time without the possibility of missing timer overflow interrupts due to clearing of TOF.

If the timer overflow flag (TOF) is set when the counter register LSB is read, then a read of the TSR will clear the flag.

The free-running counter is set to \$FFFC during reset and is always a read-only register. During a power-on reset, the counter is also preset to \$FFFC and begins running after the oscillator start-up delay. Because the free-running counter is 16 bits preceded by a fixed divide-by-four prescaler, the value in the free-running counter repeats every 262 144 internal bus clock cycles. TOF is set when the counter overflows (from \$FFFF to \$0000); this will cause an interrupt if TOIE is set.

Bits 8–15 — MSB of counter/alternate counter register

A read of only the more significant byte (MSB) transfers the LSB to a buffer, which remains fixed after the first MSB read, until the LSB is also read.

Bits 0–7 — LSB of counter/alternate counter register

A read of only the less significant byte (LSB) receives the count value at the time of reading.

6.2 Timer functions

The 16-bit programmable timer is monitored and controlled by a group of fifteen registers, full details of which are contained in the following paragraphs. An explanation of the timer functions is also given.

6.2.1 Timer control registers

The timer control registers at locations \$2C and \$2D are read/write registers. Five bits control interrupts associated with the timer status register flags IC1F, IC2F, OC1F, OC2F and TOF. Two bits control which edge is significant to the input capture 1 and 2 edge detectors.

6.2.1.1 Timer control register 1 (TCR1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Timer control 1 (TCR1)	\$002C	IC1IE	IC2IE	OC1IE	TOIE	CO1E	IEDG1	IEDG2	OLVL1	0000 0uu0

IC1IE — Input capture 1 interrupt enable

- 1 (set) — Input capture 1 interrupt enabled.
- 0 (clear) — Input capture 1 interrupt disabled.

IC2IE — Input capture 2 interrupt enable

- 1 (set) — Input capture 2 interrupt enabled.
- 0 (clear) — Input capture 2 interrupt disabled.

OC1IE — Output compare 1 interrupt enable

- 1 (set) — Output compare 1 interrupt enabled.
- 0 (clear) — Output compare 1 interrupt disabled.

TOIE — Timer overflow interrupt enable

- 1 (set) — Timer overflow interrupt enabled.
- 0 (clear) — Timer overflow interrupt disabled.

CO1E — Timer compare 1 output enable

- 1 (set) — Output of timer compare 1 is enabled.
- 0 (clear) — Output of timer compare 1 is disabled.

Reset clears this bit.

IEDG1 — Input edge 1

This bit determines which level transition on TCAP1 pin will trigger the transfer of the free-running counter to input capture register.

- 1 (set) — TCAP1 is rising edge sensitive.
- 0 (clear) — TCAP1 is falling edge sensitive.

When IEDG1 is set, a rising edge on the TCAP1 pin will trigger a transfer of the free-running counter value to the input capture register. When clear, a falling edge triggers the transfer.

Reset does not affect the IEDG1 bit.

IEDG2 — Input edge 2

This bit determines which level transition on TCAP2 pin will trigger the transfer of the free-running counter to input capture register 2.

- 1 (set) — TCAP2 is rising edge sensitive.
- 0 (clear) — TCAP2 is falling edge sensitive.

When IEDG2 is set, a rising edge on the TCAP2 pin will trigger a transfer of the free-running counter value to the input capture register. When clear, a falling edge triggers the transfer.

Reset does not affect the IEDG2 bit.

OLVL1 — Output level 1

This bit determines the level that is clocked into the output level register by the next successful output compare 1 and which will appear on the TCMP1 pin.

- 1 (set) — A high output level will appear on the TCMP1 pin.
- 0 (clear) — A low output level will appear on the TCMP1 pin.

When OLVL1 is set, a high output level will be clocked into the output level register by the next successful output compare, and will appear on the TCMP1 pin. When clear, it will be a low level that will appear on the TCMP1 pin.

6.2.1.2 Timer control register 2 (TCR2)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Timer control 2 (TCR2)	\$002D	0	0	OC2IE	0	CO2E	0	0	OLVL2	0000 0000

OC2IE — Output compare 2 interrupt enable

- 1 (set) — Output compare 2 interrupt enabled.
- 0 (clear) — Output compare 2 interrupt disabled.

CO2E — Timer compare 2 output enable

- 1 (set) — Output of timer compare 2 is enabled.
- 0 (clear) — Output of timer compare 2 is disabled.

Reset clears this bit.

OLVL2 — Output level 2

This bit determines the level that is clocked into the output level register by the next successful output compare 2 and which will appear on the TCMP2 pin.

- 1 (set) — A high output level will appear on the TCMP2 pin.
- 0 (clear) — A low output level will appear on the TCMP2 pin.

When OLVL2 is set, a high output level will be clocked into the output level register by the next successful output compare, and will appear on the TCMP2 pin. When clear, it will be a low level that will appear on the TCMP2 pin.

Bits 1, 2 4, 6 and 7 — unused; always read 0.

6.2.2 Timer status register (TSR)

The timer status register (\$2E) contains the status bits for the interrupt conditions — ICF, OCF, TOF.

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

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Timer status (TSR)	\$002E	IC1F	IC2F	OC1F	TOF	TCAP1	TCAP2	OC2F	0	uuuu 11u0

IC1F — Input capture 1 flag

1 (set) — Valid input capture has occurred.

0 (clear) — No input capture has occurred.

This bit is set when the selected polarity of edge is detected by the input capture 1 edge detector; an input capture interrupt will be generated, if IC1IE is set. IC1F is cleared by reading the TSR and then the input capture low 1 register at \$21.

6

IC2F — Input capture 2 flag

1 (set) — Valid input capture has occurred.

0 (clear) — No input capture has occurred.

This bit is set when the selected polarity of edge is detected by the input capture 2 edge detector; an input capture interrupt will be generated, if IC2IE is set. IC2F is cleared by reading the TSR and then the input capture low 2 register at \$25.

OC1F — Output compare 1 flag

1 (set) — A valid output compare has occurred.

0 (clear) — No output compare has occurred.

This bit is set when the output compare register contents match those of the free-running counter; an output compare interrupt will be generated, if OC1IE is set. OC1F is cleared by reading the TSR and then the output compare low 1 register at \$23.

TOF — Timer overflow flag

1 (set) — Timer overflow has occurred.

0 (clear) — No timer overflow has occurred.

This bit is set when the free-running counter overflows from \$FFFF to \$0000; a timer overflow interrupt will occur, if TOIE is set. TOF is cleared by reading the TSR and the counter low register at \$29.

When using the timer overflow function and reading the free-running counter at random times to measure an elapsed time, a problem may occur whereby the timer overflow flag is unintentionally cleared if:

- the timer status register is read or written when TOF is set and
- the LSB of the free-running counter is read, but not for the purpose of servicing the flag.

Reading the alternate counter register instead of the counter register will avoid this potential problem.

TCAP1 — Timer capture 1

This bit reflects the current status of the timer capture 1 input.

6

Note: On the MC68HC05L28, TCAP1 is connected directly to PD0 which defaults to an input with pull-up on reset. TCAP1 will be '1' unless PD0 is externally driven low.

TCAP2 — Timer capture 2

This bit reflects the current state of the timer capture 2 Input.

Note: On the MC68HC05L28, TCAP2 is connected directly to PD2 which defaults to an input with pull-up on reset. TCAP2 will be '1' unless PD2 is externally driven low.

OC2F — Output compare 2 flag

- 1 (set) — A valid output compare has occurred.
- 0 (clear) — No output compare has occurred.

This bit is set when the output compare register contents match those of the free-running counter; an output compare interrupt will be generated, if OC2IE is set. OC2F is cleared by reading the TSR and then the output compare low 2 register at \$27.

Bit 0 — always reads zero.

6.2.3 Input capture registers

'Input capture' is a technique whereby an external signal (connected to the TCAP1 or TCAP2 pin) is used to trigger a read of the free-running counter. In this way it is possible to relate the timing of an external signal to the internal counter value, and hence to elapsed time. There are two identical input capture registers.

6.2.3.1 Input capture register 1

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Input capture high 1 (ICH1)	\$0020									Unaffected
Input capture low 1 (ICL1)	\$0021									Unaffected

The two 8-bit registers that make up the 16-bit input capture register 1 are read-only, and are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a valid transition. The level transition that triggers the counter transfer is defined by the input edge bit (IEDG1). The most significant 8 bits are stored in the input capture high 1 register at \$20 and the least significant in the input capture low 1 register at \$21.

An interrupt can accompany a capture if the corresponding interrupt enable bit (IC1IE in timer control register 1 at \$2C) is set.

The result obtained from an input capture will be one greater 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 synchronisation. 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 valid signal transition whether the input capture 1 flag (IC1F) 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 (\$20), the counter transfer is inhibited until the LSB (\$21) is also read. This 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 (\$21) does not inhibit the free-running counter transfer since the two actions occur on opposite edges of the internal bus clock.

Reset does not affect the contents of the input capture register, except when exiting STOP mode.

6.2.3.2 Input capture register 2

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Input capture high 2 (ICH2)	\$0024									Unaffected
Input capture low 2 (ICL2)	\$0025									Unaffected

The two 8-bit registers that make up the 16-bit input capture register 2 are read-only, and are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a valid transition. The level transition that triggers the counter transfer is defined by the input edge bit (IEDG2). The most significant 8 bits are stored in the input capture high 2 register at \$24 and the least significant in the input capture low 2 register at \$25.

An interrupt can accompany a capture if the corresponding interrupt enable bit (IC2IE in timer control register 1 at \$2C) is set.

6

The result obtained from an input capture will be one greater 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 synchronisation. 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 valid signal transition whether the input capture 2 flag (IC2F) 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 (\$24), the counter transfer is inhibited until the LSB (\$25) is also read. This 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 (\$25) does not inhibit the free-running counter transfer since the two actions occur on opposite edges of the internal bus clock.

Reset does not affect the contents of the input capture register, except when exiting STOP mode.

6.2.4 Output compare registers

'Output compare' is a technique that may be used, for example, to generate an output waveform, or to signal when a specific time period has elapsed, by presetting the output compare register to the appropriate value. There are two output compare registers – OC1 and OC2. All the 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.

6.2.4.1 Output compare register 1

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Output compare high 1 (OCH1)	\$0022									Unaffected
Output compare low 1 (OCL1)	\$0023									Unaffected

6

The 16-bit output compare register 1 is made up of two 8-bit registers at locations \$22 (MSB) and \$23 (LSB). The contents of the output compare 1 register are compared with the contents of the free-running counter once every four internal processor clock cycles. If a match is found, the output compare 1 flag (OC1F) in the timer status register is set and the output level 1 (OLVL1) bit is clocked to the TCMP1 pin. The output compare 1 register values and the output level 1 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 (OC1IE) is set.

After a processor write cycle to the output compare register 1 containing the MSB (\$22), the output compare function is inhibited until the LSB (\$23) is also written. The user must write both bytes (locations) if the MSB is written first. A write made only to the LSB will not inhibit the compare function.

The processor can write to either byte of the output compare register 1 without affecting the other byte. The output level 1 bit (OLVL1) is clocked to the output level 1 register whether the output compare 1 flag (OC1F) is set or clear. The minimum time required to update the output compare 1 register is a function of the program rather than the internal hardware. Because the output compare 1 flag and the output compare 1 register are not defined at power on, and not affected by reset, care must be taken when initialising output compare functions with software. The following procedure is recommended:

- 1) write to output compare high 1 to inhibit further compares;
- 2) read the timer status register to clear OC1F (if set);
- 3) write to output compare low 1 to enable the output compare 1 function.

6.2.4.2 Output compare register 2

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Output compare high 2 (OCH2)	\$0026									Unaffected
Output compare low 2 (OCL2)	\$0027									Unaffected

The 16-bit output compare register 2 is made up of two 8-bit registers at locations \$26 (MSB) and \$27 (LSB). The contents of the output compare 2 register are compared with the contents of the free-running counter once every four internal processor clock cycles. If a match is found, the output compare 2 flag (OC2F) in the timer status register is set and the output level 2 (OLVL2) bit is clocked to the TCMP2 pin. The output compare 2 register values and the output level 2 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 (OC2IE) is set.

6

After a processor write cycle to the output compare register 1 containing the MSB (\$26), the output compare function is inhibited until the LSB (\$27) is also written. The user must write both bytes (locations) if the MSB is written first. A write made only to the LSB will not inhibit the compare function.

The processor can write to either byte of the output compare 2 register without affecting the other byte. The output level 2 bit (OLVL2) is clocked to the output level 2 register whether the output compare 2 flag (OC2F) is set or clear. The minimum time required to update the output compare 2 register is a function of the program rather than the internal hardware. Because the output compare 2 flag and the output compare 2 register are not defined at power on, and not affected by reset, care must be taken when initialising output compare functions with software. The following procedure is recommended:

- write to output compare high 2 to inhibit further compares;
- read the timer status register to clear OC2F (if set);
- write to output compare low 2 to enable the output compare 2 function.

Note: As the TCMP1 and TCMP2 pins are shared with port D, output compares 1 and 2 cannot be used for compare when the pins are selected to be input. However, the data register can still be used as a temporary store.

6.3 Timer during WAIT mode

In WAIT mode all CPU action is suspended, whereas the timer continues to run.

6.4 Timer during STOP mode

In the STOP mode all MCU clocks are stopped, so the timer stops counting. If STOP is exited by an interrupt the counter retains the last count value. If the device is reset, the counter is forced to \$FFFC. During STOP, if at least one valid input capture edge occurs at the TCAP pins, the input capture detect circuit is armed. This does not set any timer flags nor wake up the MCU. When the MCU does wake up, however, there is an active input capture flag and data from the first valid edge that occurred during the STOP period. If the device is reset to exit STOP mode, no input capture flag or data remains, even if a valid input capture edge occurred.

6

6.5 Timer state diagrams

The relationships between the internal clock signals, the counter contents and the status of the flag bits are shown in the following diagrams. It should be noted that the signals labelled 'internal' (processor clock, timer clocks and reset) are not available to the user.

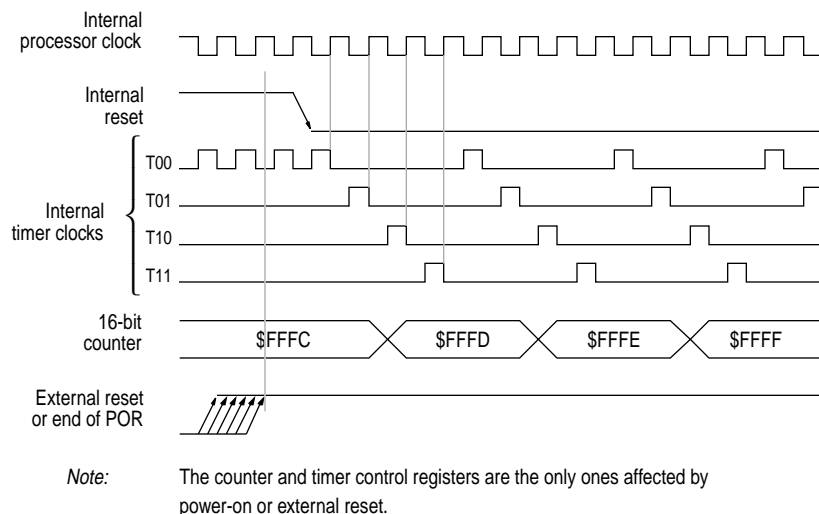
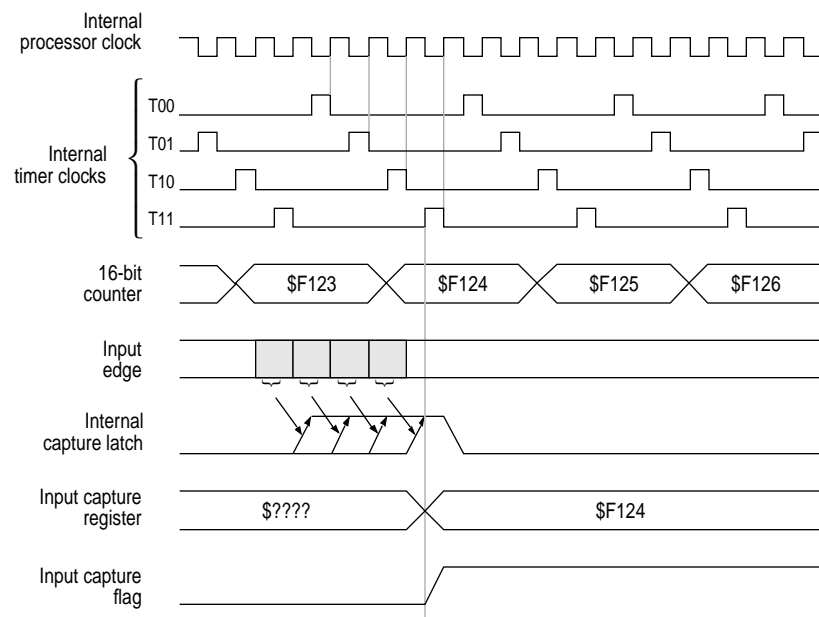


Figure 6-2 Timer state timing diagram for reset



Note: If the input edge occurs in the shaded area from one timer state T10 to the next timer state T10, then the input capture flag will be set during the next T11 state.

Figure 6-3 Timer state timing diagram for input capture

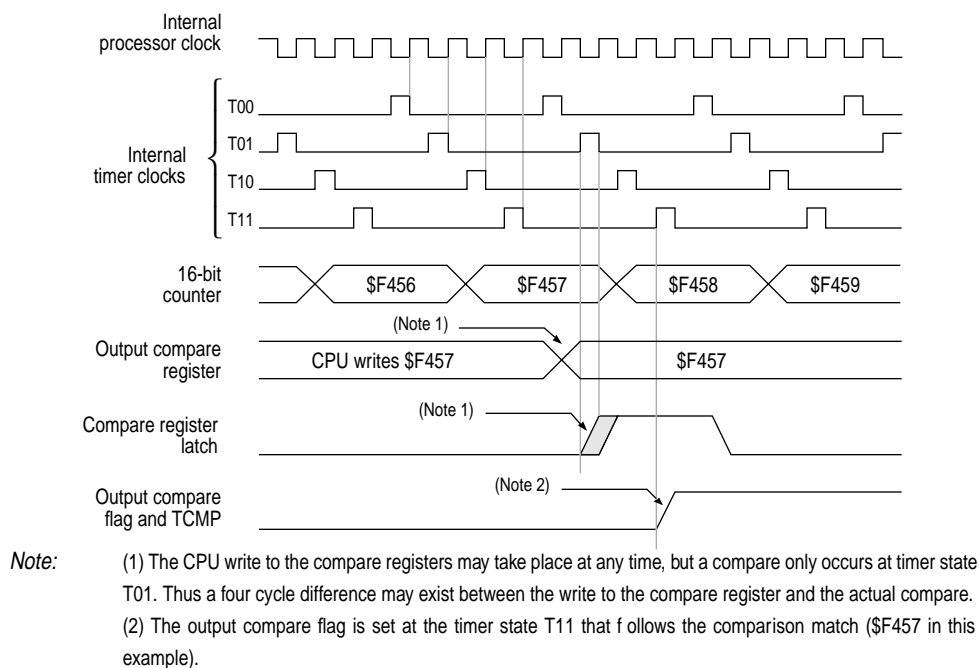


Figure 6-4 Timer state timing diagram for output compare

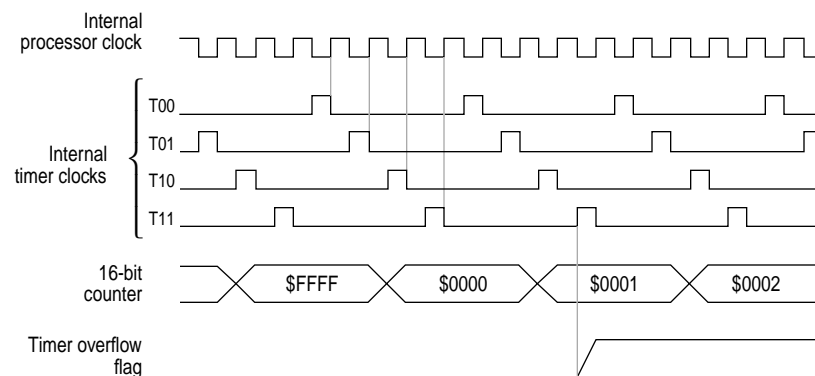


Figure 6-5 Timer state timing diagram for timer overflow

THIS PAGE LEFT BLANK INTENTIONALLY

7

LIQUID CRYSTAL DISPLAY DRIVER MODULE

This chapter describes the generic M68HC05 family LCD driver module. Any differences in the module specific to the MC68HC05L28 are indicated along with the generic description.

The M68HC05 family LCD driver module can be configured with up to 24 frontplane drivers and a maximum of 4 backplane drivers. This allows a maximum of 96 LCD segments.

The LCD driver module on the MC68HC05L28 supports 18 frontplanes and 4 backplanes, allowing a maximum of 72 LCD segments. Each segment is controlled by a corresponding bit in the LCD RAM. The mode of operation is determined by the values set in the LCD control register at \$1E.

At reset or on power-up, the drivers are configured in the default duplex mode, 1/2 bias with 2 backplanes and 18 frontplanes. Also at power-up or reset the ON/OFF control for the display, the DISON bit in the LCD control (LCD) register, is cleared disabling the LCD drivers. Figure 7-1 shows a block diagram of the LCD system.

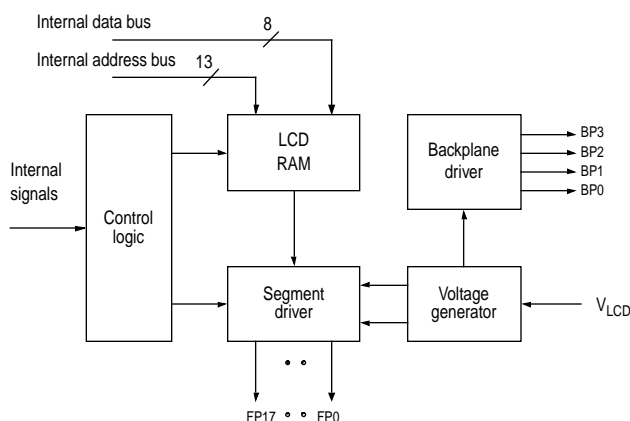


Figure 7-1 LCD system block diagram

7.1 LCD RAM

Data to be displayed on the LCD must be written into the LCD RAM. The LCD RAM is comprised of 12 bytes of RAM (in the MC68HC05L28's memory map) at \$0040 – \$004B. The 96 bits in the LCD RAM correspond to the 96 segments that can be driven by the frontplane/backplane drivers. Table 7-1 shows how the LCD RAM is organized. Writing a '1' to a given location will result in the corresponding display segment being activated when the DISON bit is set. The LCD RAM is a dual port RAM that interfaces with the internal address and data buses of the MCU. It is possible to read from LCD RAM locations for scrolling purposes. When DISON = 0, the LCD RAM can be used as main on-chip RAM.

Table 7-1 LCD RAM organization

LCD RAM			Data					
Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
\$40	FP1-BP3	FP1-BP2	FP1-BP1	FP1-BP0	FP0-BP3	FP0-BP2	FP0-BP1	FP0-BP0
\$41	FP3-BP3	FP3-BP2	FP3-BP1	FP3-BP0	FP2-BP3	FP2-BP2	FP2-BP1	FP2-BP0
\$42	FP5-BP3	FP5-BP2	FP5-BP1	FP5-BP0	FP4-BP3	FP4-BP2	FP4-BP1	FP4-BP0
\$43	FP7-BP3	FP7-BP2	FP7-BP1	FP7-BP0	FP6-BP3	FP6-BP2	FP6-BP1	FP6-BP0
\$44	FP9-BP3	FP9-BP2	FP9-BP1	FP9-BP0	FP8-BP3	FP8-BP2	FP8-BP1	FP8-BP0
\$45	FP11-BP3	FP11-BP2	FP11-BP1	FP11-BP0	FP10-BP3	FP10-BP2	FP10-BP1	FP10-BP0
\$46	FP13-BP3	FP13-BP2	FP13-BP1	FP13-BP0	FP12-BP3	FP12-BP2	FP12-BP1	FP12-BP0
\$47	FP15-BP3	FP15-BP2	FP15-BP1	FP15-BP0	FP14-BP3	FP14-BP2	FP14-BP1	FP14-BP0
\$48	FP17-BP3	FP17-BP2	FP17-BP1	FP17-BP0	FP16-BP3	FP16-BP2	FP16-BP1	FP16-BP0
\$49	FP19-BP3	FP19-BP2	FP19-BP1	FP19-BP0	FP18-BP3	FP18-BP2	FP18-BP1	FP18-BP0
\$4A	FP21-BP3	FP21-BP2	FP21-BP1	FP21-BP0	FP20-BP3	FP20-BP2	FP20-BP1	FP20-BP0
\$4B	FP23-BP3	FP23-BP2	FP23-BP1	FP23-BP0	FP22-BP3	FP22-BP2	FP22-BP1	FP22-BP0

These LCD pins are not available on the MC68HC05L28/MC68HC705L28. The corresponding RAM bytes (\$49 to \$4B) can continue to be used as main on-chip RAM.

7.2 LCD operation

The LCD driver module can operate in four modes providing different multiplex ratios and number of backplanes as follows:

- 1/2 bias, 2 backplanes
- 1/3 bias, 2 backplanes
- 1/3 bias, 3 backplanes
- 1/4 bias, 4 backplanes

The operating mode is selected at power on using the multiplex ratio bits (MUX3 and MUX4) in the LCD control register as shown in Table 7-2.

It is recommended that the DISON bit in the LCD register is not set (display is disabled) until the multiplex rate is selected. The voltage levels required for the different multiplex rates are generated internally by a resistive divider chain between V_{LCD} , V_{DD} and V_{SS} .

The 2-way multiplex with 1/3 bias and the three and four-way multiplex options require four voltage levels, whereas the two-way multiplex with 1/2 bias needs only three levels. Resistors R1, R2 and R3 are valued at $25k \pm 40\%$. Figure 7-2 shows the resistive divider chain network that is used to produce the various LCD waveforms outlined in Section 7.3.

Note: V_{LCD} may not exceed the positive power supply voltage V_{DD} .

Note: The V_{LCD} option is not available on the MC68HC05L28 or MC68HC705L28, but is included here for completeness of the generic module description. Bit 6 of the LCD control register must be cleared.

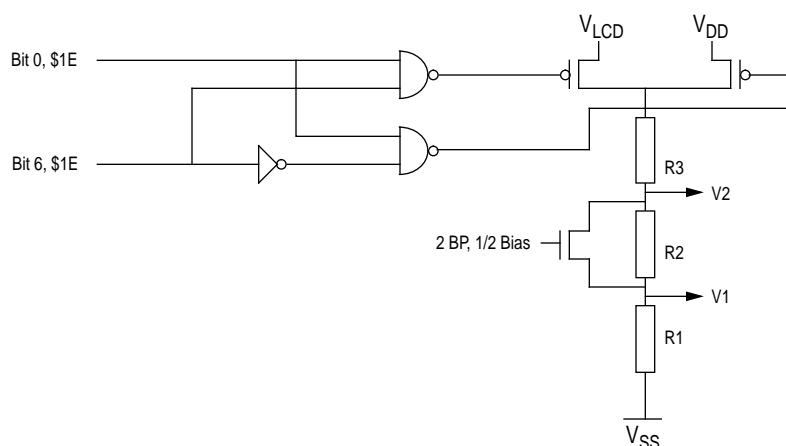


Figure 7-2 Voltage level selection

7.3 Timing signals and LCD voltage waveforms

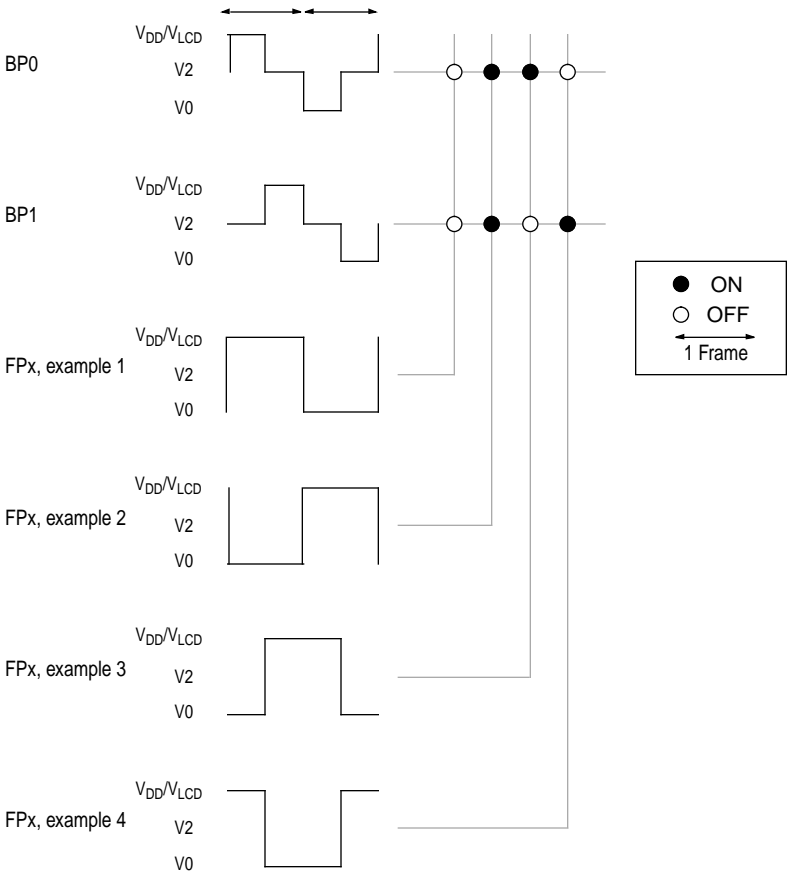
The LCD timing signals are all derived from the main system clock; with a bus frequency of 2 MHz ($f_{osc} = 4$ MHz) the frame rate will be 61 Hz for 2 and 4-way multiplexing and 91 Hz for 3-way multiplexing (see Table 7-2). An extra divide-by-two stage can be included in the LCD clock generator by setting FDISP in the LCD register. This will result in the frame rate being halved. For

example, when 3-way multiplexing is used, a frame rate of 45.5 Hz instead of 91 Hz can be obtained. See Section 7.4.

Figure 7-3 to Figure 7-6 show the backplane waveforms and some examples of frontplane waveforms for each of the operating modes.

The backplane waveforms are continuous and repetitive (every 2 frames); they are fixed within each operating mode and are not affected by the data in the LCD RAM.

The frontplane waveforms are dependent on the LCD segments to be driven as defined in the LCD RAM. Each "on" segment must have a differential driving voltage (BP-FP) applied to it once in each frame; the LCD driver module hardware uses the data in the LCD RAM to construct the frontplane waveform to meet this criterion.



Note: In this mode $V1=V2$

Figure 7-3 LCD waveform with 2 backplanes, 1/2 bias

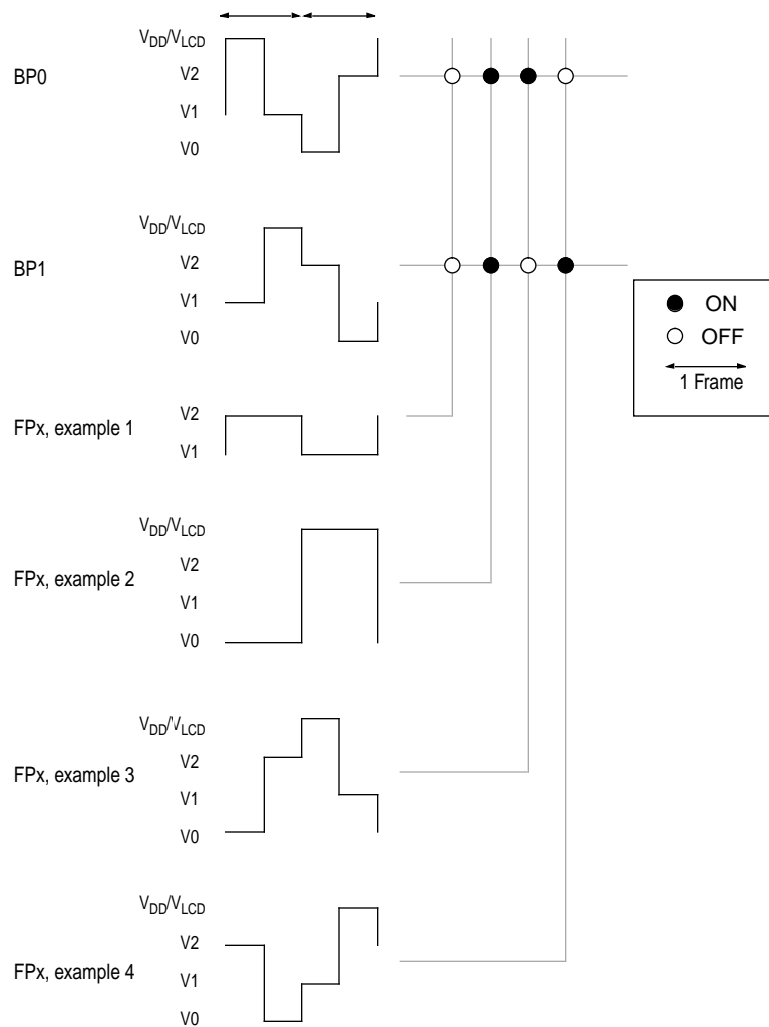


Figure 7-4 LCD waveform with 2 backplanes, 1/3 bias

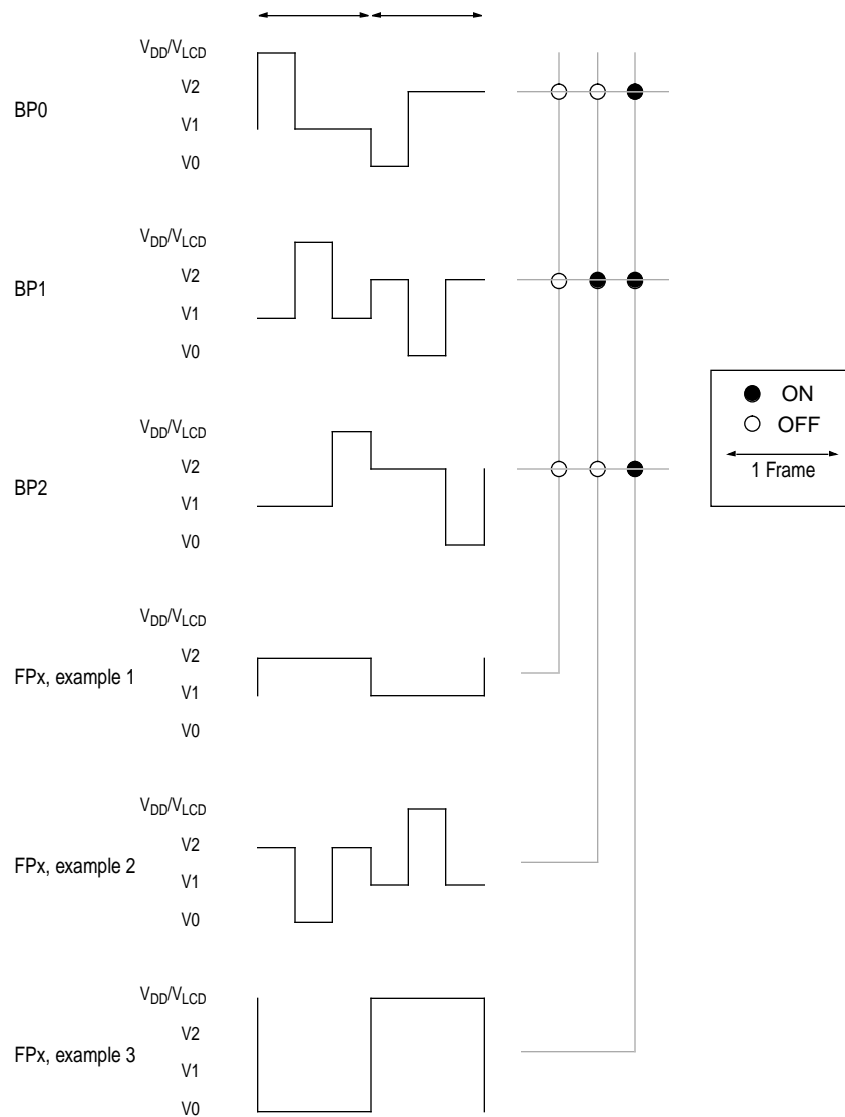


Figure 7-5 LCD waveform with 3 backplanes

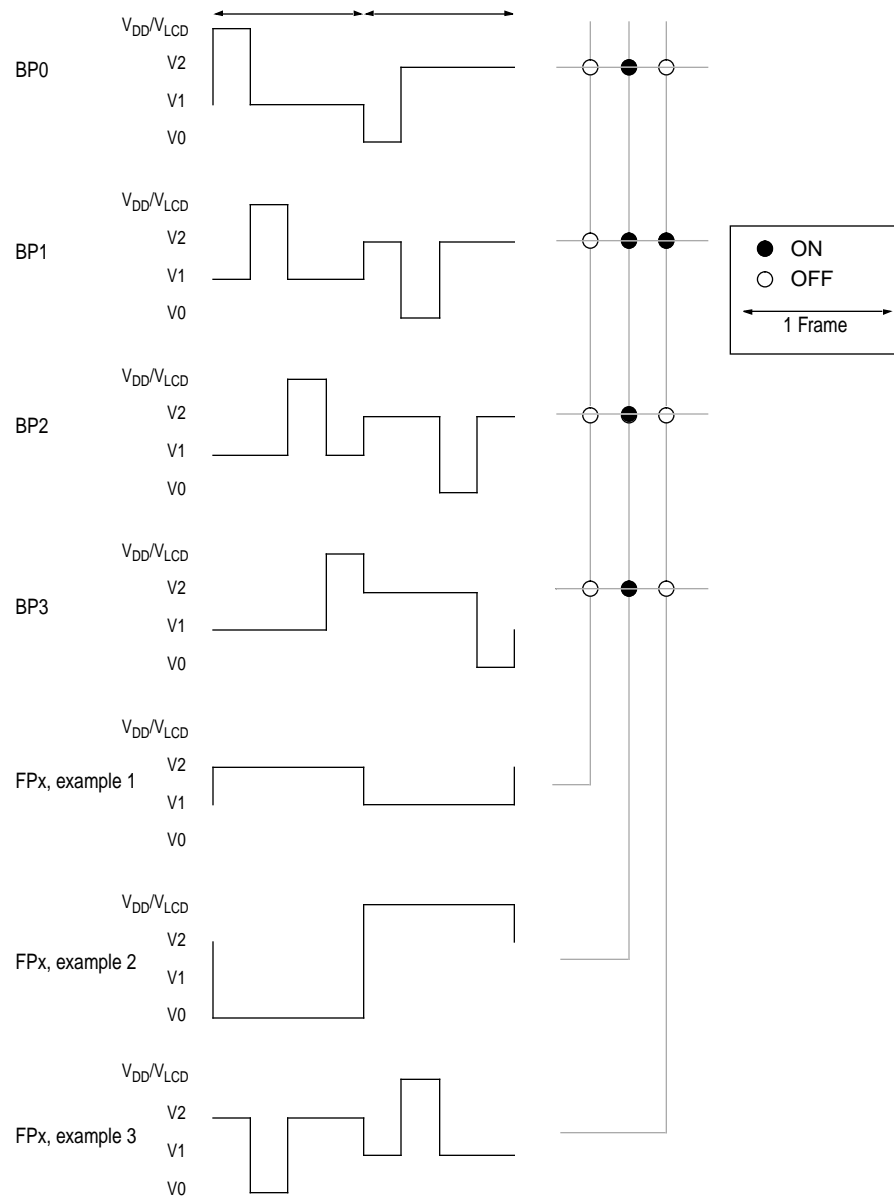


Figure 7-6 LCD waveform with 4 backplanes

7.4 LCD control register

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
LCD control register (LCD)	\$001E		VLCDON			FDISP	MUX4	MUX3	DISON	?000 0000

VLCDON — LCD voltage select

The V_{LCD} option is not available on the MC68HC05L28 or MC68HC705L28, therefore, this bit must be cleared.

FDISP — Display frequency

- 1 (set) — An extra divide by two stage is included in the LCD clock generator to give a reduced frame rate. For example, in the 3-way multiplexing mode, a frame rate of 45.5 Hz instead of 91 Hz can be achieved.
- 0 (clear) — Default frame rate is used.

7

MUX4, MUX3 — Multiplex ratio

These two bits select the multiplex ratio to be 2, 3 or 4 backplanes.

Table 7-2 Multiplex ratio/backplane selection

MUX4	MUX3	Backplanes	Bias	Frequency
0	0	2	1/2	61 Hz
0	1	3	1/3	91 Hz
1	0	4	1/3	61 Hz
1	1	2	1/3	61 Hz

DISON — Display ON/OFF

- 1 (set) — Display is ON.
- 0 (clear) — Display is OFF

Reserved bits

Bits 4, 5 and 7 are reserved for future use and must be set to 0 when writing to this register.

7.5 LCD during WAIT mode

The LCD does not function during WAIT mode.

8

I²C-BUS

I²C-bus is a two-wire, bidirectional serial bus that provides a simple, efficient way to exchange data between devices. Being a two-wire device, the I²C-bus minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

The bus is suitable for applications involving frequent communications between a number of devices over short distances. The number of devices connected to the I²C-bus is limited only by a maximum bus capacitance of 400pF; it has a maximum data rate of 100 kbits per second.

The I²C-bus system is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters attempt to control the bus simultaneously. This feature provides the capability for complex applications with multiprocessor control. It may also be used for rapid testing and alignment of end products via external connections to an assembly line computer.

The I²C-bus function is enabled by the MEN bit in the I²C-bus control register (MCR).

8

8.1 I²C-bus features

- Multi-master operation
- Software-programmable for one of 32 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost-driven interrupt with automatic switching from master to slave mode
- Calling address identification interrupt
- Generates/detects the START or STOP signal
- Repeated START signal generation
- Generates/recognizes the acknowledge bit
- Bus busy detection

8.2 I²C-bus system configuration

The I²C-bus system uses a serial data line and a serial clock line for data transfer. All the devices connected to it must have open drain or open collector outputs. A logic 'AND' function is used on both lines with two pull-up resistors.

8.3 I²C-bus protocol

A standard communication is normally composed of four parts: START signal, slave address transmission, data transfer, and STOP signal. These signals are described in the following sections and illustrated in Figure 8-1.

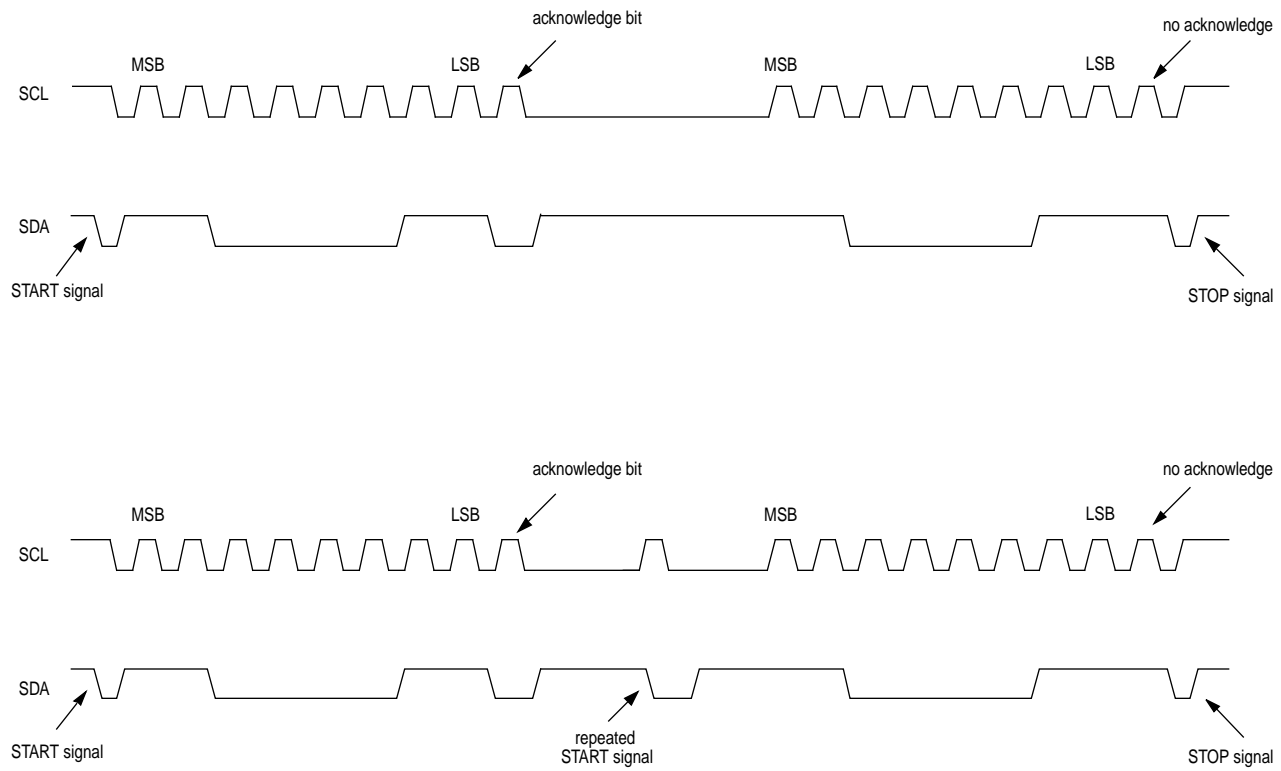
8.3.1 START signal

When the bus is free (no master device engaging the bus; SCL and SDA lines at a logic high), a master may initiate communication by sending a START signal, which is defined as being a high to low transition of SDA with SCL high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and wakes up all slaves.

8.3.2 Transmission of the slave address

The first byte of data transferred after the START signal is the slave address transmitted by the master. This address is seven bits long, followed by a $\overline{R/W}$ bit which tells the slave the desired direction of transfer of all the following bytes (until a STOP or repeated start).

Only the slave with the calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 8-1).

Figure 8-1 I²C bus transmission signal diagrams

8.3.3 Data transfer

Once successful slave addressing has been achieved, the data transfer can proceed byte by byte, in the direction specified by the R/\overline{W} bit.

Data can be changed only when SCL is low and must be held stable while SCL is high. The MSB is transmitted first. Each data byte is eight bits long, and there is one clock pulse on SCL for each data bit. Every byte of data must be followed by an acknowledge bit, which the receiving device signals by pulling SDA low at the ninth clock. Therefore, one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, then the SDA line is left high by the slave. The master can then generate a STOP signal to abort the data transfer or a START signal to commence a new calling (called a repeated start).

If the master receiver does not acknowledge the slave transmitter after one byte of transmission, it means 'end of data' to the slave, which then releases the SDA line so that the master can generate the STOP or START signal.

8.3.4 STOP signal

8

The master can terminate the communication by generating a STOP signal to free the bus. A STOP signal is defined as a low to high transition of SDA while SCL is high (see Figure 8-1).

8.3.5 Repeated START signal

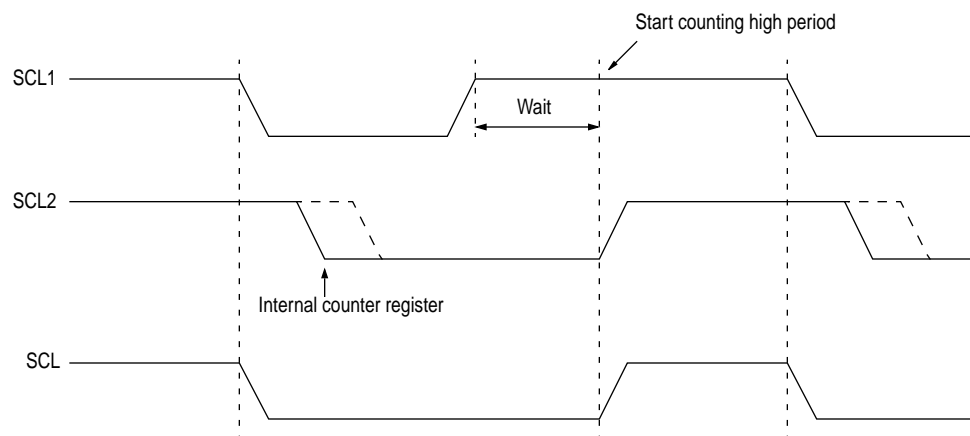
A repeated START signal generates a START signal without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave, or with the same slave in a different mode (transmit/receive mode), without releasing the bus.

8.3.6 Arbitration procedure

The I²C-bus is a true multi-master system that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high period is equal to the shortest clock high period among the masters. A data arbitration procedure determines the relative priority of the contending masters; a master loses arbitration if it transmits logic 1 while another transmits logic 0. The losing masters then immediately switch to slave receive mode and stop driving SDA outputs. The transition from master to slave mode does not generate a STOP condition in this case. At this point, the MAL bit in the I²C-bus status register (MSR) is set by hardware to indicate loss of arbitration.

8.3.7 Clock synchronization

Since wired-AND logic is performed on the SCL line, a high to low transition on SCL affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 8-2). When all devices concerned have counted off their low period, the SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line, and all of them start counting their high periods. The first device to complete its high period pulls the SCL line low again.



8

Figure 8-2 Clock synchronization

8.3.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. The slave device may hold SCL low after the completion of one byte of data transfer (nine bits). In such cases, it halts the bus clock and forces the master clock into a wait state until the slave releases the SCL line.

8.4 Registers

8.4.1 I²C-bus address register (MADR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
I ² C-bus address register (MADR)	\$0010	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1		0000 000u

ADR7 – ADR1 — Slave address bits

These bits define the slave address of the I²C-bus, and are used in slave mode in conjunction with the MAAS bit in the MSR register (see Section 8.4.4). These bits can be read and written at any time.

Bit 0 — reserved by Motorola.

8.4.2 I²C-bus frequency divider register (FDR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
I ² C-bus frequency divider register (FDR)	\$0011				MBC4	MBC3	MBC2	MBC1	MBC0	uuu0 0000

8

MBC4 – MBC0 — Clock rate select bits

These bits can be read and written at any time.

These bits are used to prescale the clock for bit rate selection. Due to the potential slow rise and fall times of the SCL and SDA signals the bus signals are sampled at the prescaler frequency. This sampling incurs an overhead of six clocks per SCL pulse. The serial bit clock frequency is equal to the CPU clock divided by the divider shown in Table 8-1, plus the sampling overhead of six clocks per cycle.

For a 4 MHz external crystal operation, the serial bit clock frequency of the I²C-bus ranges from 460 Hz to 90909 kHz.

Table 8-1 I²C-bus prescaler

MCB4-0	Divider	MCB4-0	Divider	MCB4-0	Divider	MCB4-0	Divider
0 0 0 0 0	22	0 1 0 0 0	88	1 0 0 0 0	352	1 1 0 0 0	1408
0 0 0 0 1	24	0 1 0 0 1	96	1 0 0 0 1	384	1 1 0 0 1	1536
0 0 0 1 0	28	0 1 0 1 0	112	1 0 0 1 0	448	1 1 0 1 0	1792
0 0 0 1 1	34	0 1 0 1 1	136	1 0 0 1 1	544	1 1 0 1 1	2176
0 0 1 0 0	44	0 1 1 0 0	176	1 0 1 0 0	704	1 1 1 0 0	2816
0 0 1 0 1	48	0 1 1 0 1	192	1 0 1 0 1	768	1 1 1 0 1	3072
0 0 1 1 0	56	0 1 1 1 0	224	1 0 1 1 0	896	1 1 1 1 0	3584
0 0 1 1 1	68	0 1 1 1 1	272	1 0 1 1 1	1088	1 1 1 1 1	4352

8.4.3 I²C-bus control register (MCR)

These bits can be read and written at any time.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
I ² C-bus control register (MCR)	\$0012	MEN	MIEN	MSTA	MTX	TXAK				0000 0uuu

MEN — I²C-bus enable

- 1 (set) — I²C-bus interface system is enabled. This bit must be set before any other MCR bits can be set.
- 0 (clear) — I²C-bus interface system is disabled and reset. This is the power-on reset case. When low, the interface is held in reset, but registers can be accessed.

If the module is enabled in the middle of a byte transfer, the interface behaves as follows:

slave mode ignores the current transfer on the bus and starts operating when a subsequent start condition is detected.

Master mode is not aware that the bus is busy, so if a start cycle is initiated the current bus cycle may become corrupt. This results in either the current bus master or the I²C-bus losing arbitration, after which bus operation returns to normal.

MIEN — I²C-bus interrupt enable

- 1 (set) — I²C-bus interrupt is requested when MIF is set.
- 0 (clear) — I²C-bus interrupt is disabled.

MSTA — Master/slave mode select

- 1 (set) — Master mode; send START signal when set.
- 0 (clear) — Slave mode; send STOP signal when cleared.

This bit is cleared on reset. When MSTA is changed from 0 to a 1, a START signal is generated on the bus and the master mode is selected. When this bit changes from a 1 to a 0, a STOP signal is generated and the slave mode is selected. In master mode, clearing MSTA and then immediately setting it generates a repeated START signal without generating a STOP signal (see Figure 8-1).

MTX — Transmit/receive mode select

- 1 (set) — Transmit mode.
- 0 (clear) — Receive mode.

TXAK — Transmit acknowledge bit

- 1 (set) — No acknowledge signal response.
- 0 (clear) — An acknowledge signal will be sent to the bus at the ninth clock bit after receiving one byte of data.

This bit only has meaning in master receive mode.

Bits 2–0 — not implemented; always read zero.

8.4.4 I²C-bus status register (MSR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
I ² C-bus status register (MSR)	\$0013	MCF	MAAS	MBB	MAL		SRW	MIF	RXAK	1000 u001

Bits in this register can be read at any time; Bits 4 and 1 can be cleared at any time.

MCF — Data transferring

- 1 (set) — Data transmit complete.
- 0 (clear) — Data is being transferred.

MAAS — I²C-bus addressed as a slave

- 1 (set) — I²C-bus is addressed as a slave.
- 0 (clear) — I²C-bus is not addressed.

This bit is set when the address of the I²C-bus (specified in MADR) matches the calling address. An interrupt is generated providing the MIEN bit in the MCR register is set; the CPU then selects its transmit/receive mode according to the state of the SRW bit. Writing to the MCR register clears this bit.

MBB — Bus busy

- 1 (set) — Bus is busy.
- 0 (clear) — Bus is idle.

This bit indicates the status of the bus. When a START signal is detected, MBB is set. When a STOP signal is detected, MBB is cleared.

MAL — Arbitration lost

- 1 (set) — Arbitration lost.
- 0 (clear) — Default state.

MAL is set by hardware when the arbitration procedure is lost during a master transmission mode. This bit must be cleared by software.

Bit 3 — Not implemented; always reads zero.

SRW — Read/write command

- 1 (set) — R/W command bit is set (read).
- 0 (clear) — R/W command bit is clear (write).

When MAAS is set, the R/W command bit of the calling address sent from a master is latched into this bit. On checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.

MIF — I²C-bus interrupt flag

- 1 (set) — An I²C-bus interrupt is pending.
- 0 (clear) — No I²C-bus interrupt is pending.

When this bit is set, an I²C-bus interrupt is generated provided the MIEN bit in the MCR register is set. MIF is set when one of the following events occurs:

- 1) The transfer of one byte of data is complete; MIF is set at the falling edge of the ninth clock after the byte has been received.
- 2) A calling address is received which matches the address of the I²C-bus in slave receive mode.
- 3) Arbitration is lost.

MIF must be cleared by software in the interrupt routine.

RXAK — Received acknowledge bit

- 1 (set) — No acknowledge signal has been detected at the ninth clock after the transmission of a byte of data.
- 0 (clear) — An acknowledge bit has been received at the ninth clock after the transmission of a byte of data.

8.4.5 I²C-bus data register (MDR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
I ² C-bus data register (MDR)	\$0014	TRXD7	TRXD6	TRXD5	TRXD4	TRXD3	TRXD2	TRXD1	TRXD0	Undefined

These bits can be read and written at any time.

In master transmit mode, a write to this register will cause the data in it to be sent to the bus automatically, MSB first. In master receive mode, a read of this register initiates the transfer of the next incoming byte of data into the register. See Figure 8-3.

In slave transmit mode, the SCL line is forced low until data is written into this register, to prevent transmission. Similarly, in slave receive mode, the data bus must be read before a transmission can occur.

8.5 Programming

8.5.1 Initialization

8

After a reset, the I²C-bus control register (MCR) is in a default state. Before the I²C-bus can be used, it must be initialized as follows:

- 1) Configure the frequency divider register for the desired SCL frequency.
- 2) Configure the I²C-bus address register (MADR) to define the slave address of the I²C-bus.
- 3) Set the MEN bit in the I²C-bus control register (MCR) to enable the I²C-bus system.
- 4) Configure the other bits in the MCR register.

8.5.2 START signal and the first byte of data

After the initialization procedure has been completed, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I²C-bus busy bit (MBB) must be tested to check whether the serial bus is free. If the bus is free (MBB = 0), the START condition and the first byte (the slave address) can be sent. An example of a program that does this is shown below:

```
SEI                                ;DISABLE INTERRUPT
CHFLAG    BRSET      5,MSR,CHFLAG ;CHECK THE MBB BIT OF THE STATUS
                                           ;REGISTER. IF IT IS SET, WAIT
```


			;UNTIL IT IS CLEAR
TXSTART	BSET	4,MCR	;SET TRANSMIT MODE
	BSET	5,MCR	;SET MASTER MODE
			;i.e. GENERATE START CONDITION
	LDA	CALLING	;GET THE CALLING ADDRESS
	STA	MDR	;TRANSMIT THE CALLING ADDRESS
	CLI		;ENABLE INTERRUPT

8.5.3 Software response

The transmission or reception of a byte sets the data transferring bit, MCF, which indicates that one byte of communication is finished. Also, the I²C-bus interrupt bit, MIF, is set to generate an I²C-bus interrupt (if MIEN is set). Figure 8-3 shows an example of a typical I²C-bus interrupt routine. In the interrupt routine, the first step is for software to clear the MIF bit. The MCF bit can be cleared by reading from the I²C-bus data I/O register (MDR) in receive mode, or by writing to MDR in transmit mode. Software may service the I²C-bus I/O in the main program by monitoring the MIF bit if the interrupt function is disabled. The following is an example of a software response by a 'master transmitter' in the interrupt routine:

ISR	BCLR	1,MSR	;CLEAR THE MIF FLAG
	BRCLR	5,MCR,SLAVE	;CHECK THE MSTA FLAG
			;BRANCH IF SLAVE MODE
	BRCLR	4,MCR,RECIEVE	;CHECK THE MODE FLAG
	BRSET	0,MSR,END	;CHECK ACKNOWLEDGE FROM
			;RECEIVER
			;IF NO ACKNOWLEDGE, END
			:TRANSMISSION
TRANSMIT	LDA	DATABUF	;GET THE NEXT BYTE OF DATA

8.5.4 Generation of a STOP signal

A data transfer ends with a STOP signal generated by the master device. A master transmitter can simply generate a STOP signal after all the data has been transmitted; for example:

```
MASTX      BRSET      0,MSR,END      ;IF NO ACKNOWLEDGEMENT,
                                         ;BRANCH TO END
                                         LDAA      TXCNT      ;GET VALUE FROM THE
                                         ;TRANSMITTING COUNTER
                                         BEQ      END      ;IF NO MORE DATA, BRANCH TO END
                                         LDAA      DATABUF    ;GET NEXT BYTE OF DATA
                                         STAA      MDR      ;TRANSMIT THE DATA
                                         DEC      TXCNT      ;DECREASE THE TXCNT
                                         BRA      EMASTX     ;EXIT
END         BCLR      5,MCR          ;GENERATE A STOP CONDITION
EMASTX     RTI
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data. This can be done by setting the transmit acknowledge bit (TXAK) before reading the second last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

8

```
MASR      DEC      RXCNT
          BEQ      ENMASR      ;LAST BYTE TO BE READ
          LDA      RXCNT
          DECA
          ;CHECK SECOND LAST BYTE TO BE
          ;READ
          BNE      NXMAR      ;NOT LAST ONE OR SECOND LAST
LAMAR     BSET      3,MCR      ;SECOND LAST, DISABLE
          ;ACKNOWLEDGEMENT TRANSMITTING
          BRA      NXMAR      ;NXMAR
ENMASR     BCLR      5,MCR      ;LAST ONE, GENERATE STOP
SIGNAL
NXMAR     LDA      MDR      ;READ DATA AND STORE
          STA      RXBUF
          RTI
```

8.5.5 Generation of a repeated START signal

At the end of the data transfer, if the master still wants to communicate on the bus, it can generate another START signal, followed by another slave address, without first generating a STOP signal. For example:

RESTART	BCLR	5,MCR	;ANOTHER START (RESTART) IS
			;GENERATED BY
	BSET	5,MCR	;THESE TWO CONSECUTIVE
			;INSTRUCTIONS
	LDAA	CALLING	;GET THE CALLING ADDRESS
	STAA	MDR	;TRANSMIT THE CALLING ADDRESS

8.5.6 Slave mode

In the slave interrupt service routine, the MAAS bit should be tested to check if a calling of its own address has just been received. If MAAS is set, software should set the transmit/receive mode select bit (MTX) according to the R/W command bit, SRW. Writing to the MCR clears the MAAS bit automatically. A data transfer may then be initiated by writing to MDR or by performing a dummy read from MDR.

In the slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. If RXAK is set, this means an 'end of data' signal from the master receiver, which must then switch from transmitter mode to receiver mode by software. This is followed by a dummy read, which releases the SCL line so that the master can generate a STOP signal.

8.5.7 Arbitration lost

8

Only one master can engage the device at one time. Those devices wishing to engage the bus, but having lost arbitration, are immediately switched to slave receive mode by hardware. Their data output to the SDA line is stopped, but the internal transmitting clock is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with MAL = 1 and MSTA = 0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware inhibits the transmission; the MSTA bit is cleared without generating a STOP condition, an interrupt is generated, and MAL is set to indicate that the attempt to engage the bus has failed. In these cases, the slave interrupt service routine should test MAL first; if MAL is set, it should be cleared by software.

8.5.8 Operation during STOP and WAIT modes

During STOP mode, the I²C-bus is disabled.

During WAIT mode, the I²C-bus is idle, but 'wakes up' when it receives a valid start condition in slave mode. If the interrupt is enabled, the CPU comes out of WAIT mode after the end of a byte of transmission.

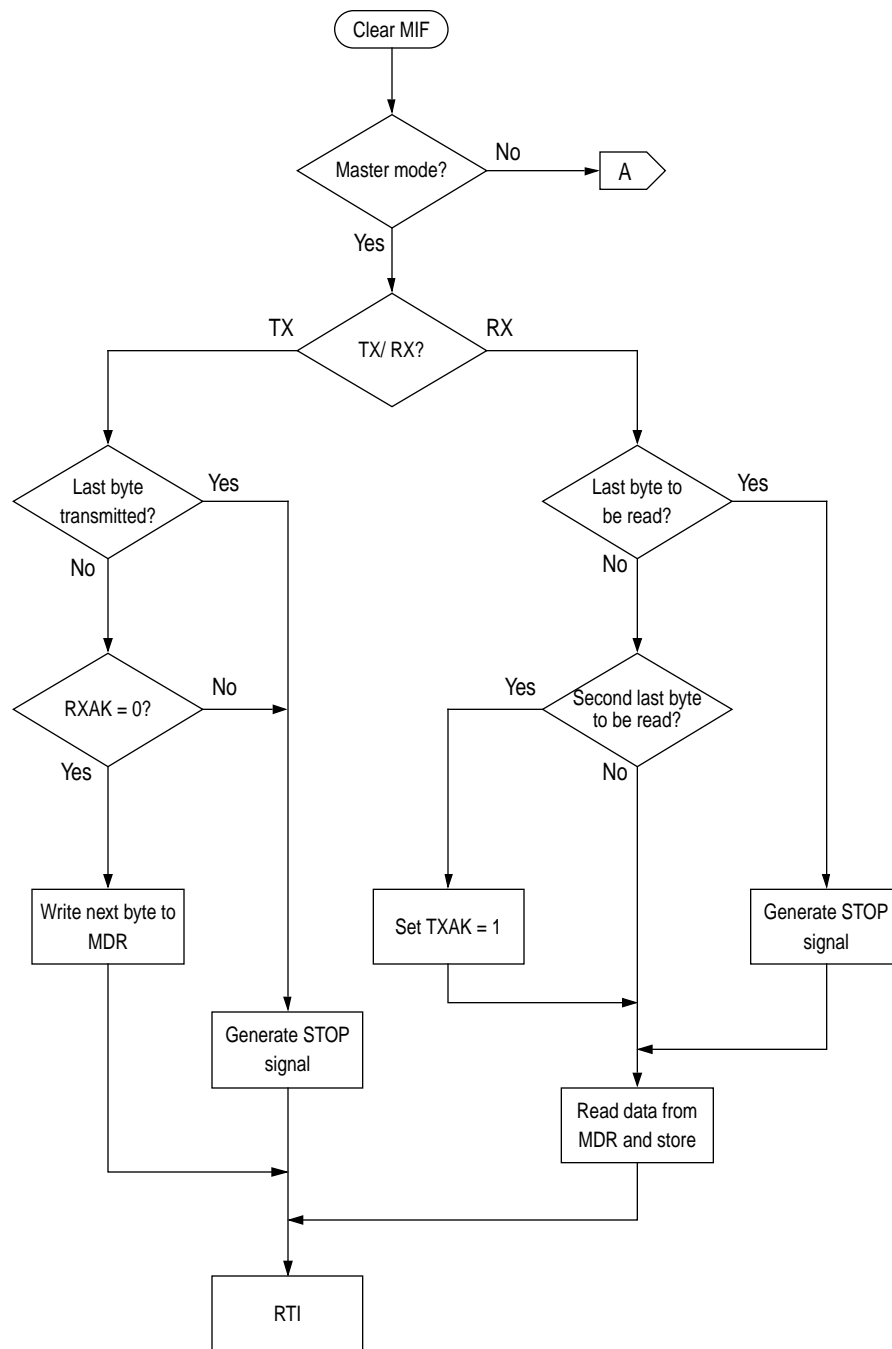


Figure 8-3 Example of a typical I²C-bus interrupt routine

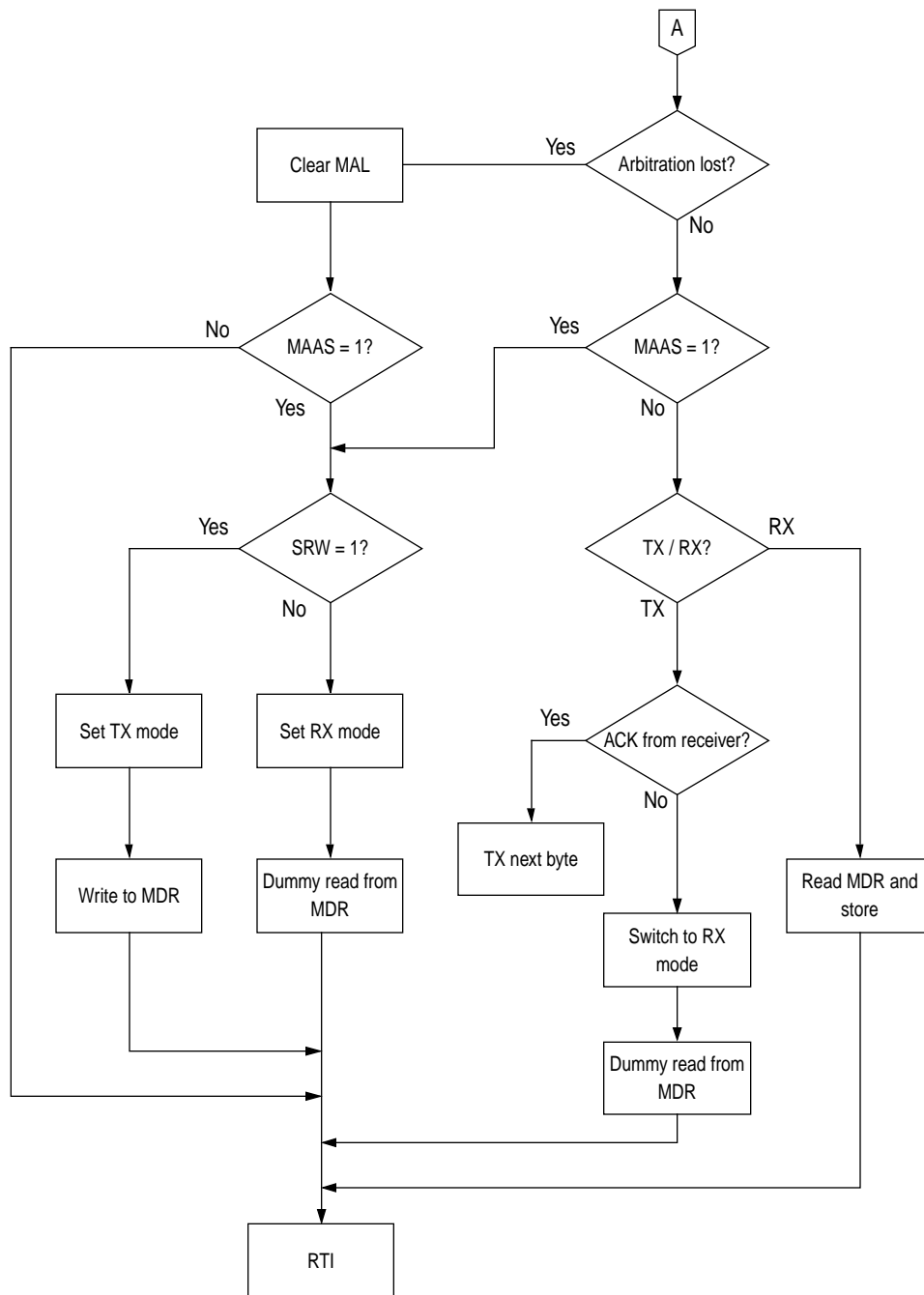


Figure 8-3 Example of a typical I²C-bus interrupt routine (Continued)

THIS PAGE LEFT BLANK INTENTIONALLY

9

A/D CONVERTER

The analog to digital converter system consists of a single 8-bit successive approximation converter and a 16-channel multiplexer. There are only two A/D channels available on the MC68HC05L28. These are connected to the ADx pins of the MC68HC05L28 and the other channels are dedicated to internal reference points for test functions. The ADx pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signal due to output buffer leakage current.

There is one 8-bit result data register, ADDATA and one 8-bit status/control register, ADSTAT.

The A/D converter is ratiometric and two dedicated pins, VREFH and VREFL, are used to supply the reference voltage levels of each analog input. These pins are used in preference to the system power supply lines because any voltage drops in the bonding wires of the heavily loaded supply pins could degrade the accuracy of the A/D conversion. An input voltage equal to or greater than V_{REFH} converts to \$FF (full scale) with no overflow indication and an input voltage equal to V_{REFL} converts to \$00.

The A/D converter can operate from either the bus clock or an internal RC type oscillator. The internal RC type oscillator is activated by the ADRC bit in ADSTAT and can be used to give a sufficiently high clock rate to the A/D converter when the bus speed is too low to provide accurate results (see Section 9.2.1.2). When the A/D converter is not being used it can be disconnected, using the ADON bit in the ADSTAT register, in order to save power (see Section 9.2.1.3).

9

9.1 A/D converter operation

The A/D converter consists of an analog multiplexer, an 8-bit digital to analog capacitor array, a comparator and a successive approximation register (SAR) (see Figure 9-1).

There are four options that can be selected by the multiplexer; the ADx input pin, VRH, $(VRH+VRL)/2$ or VRL. Selection is made via the CHx bits in the ADSTAT register (see Section 9.2.1.4). ADx are the only input points for A/D conversion operations; the others are reference points which can be used for test purposes.

The A/D reference input (ADx) is applied to a precision internal digital to analog converter. Control logic drives this D/A converter and the analog output is successively compared with the analog input (ADx) sampled at the beginning of the conversion. The conversion is monotonic with no missing codes.

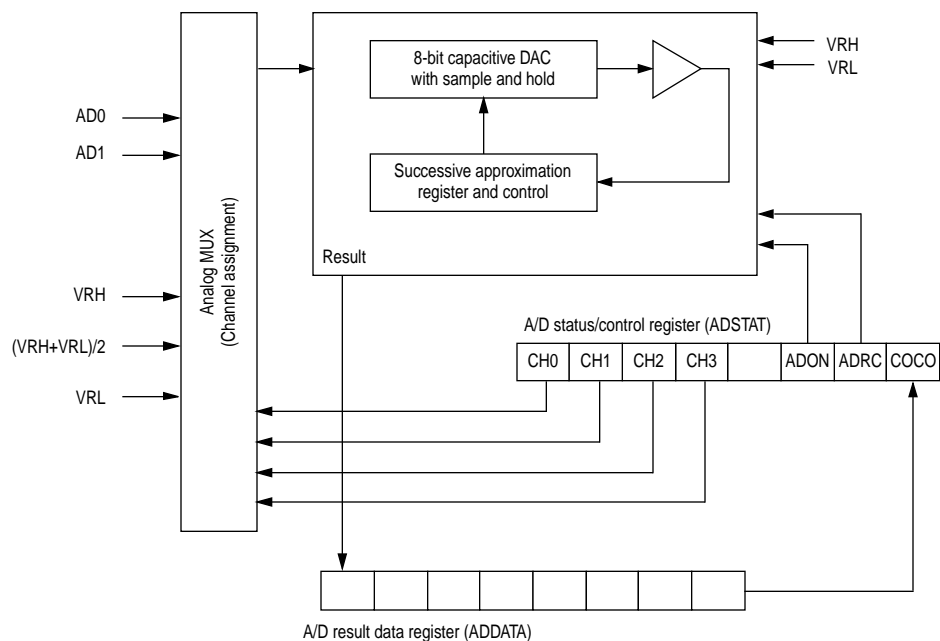


Figure 9-1 A/D converter block diagram

9

The result of each successive comparison is stored in the SAR and, when the conversion is complete, the contents of the SAR are transferred to the read-only result data register (\$17), and the conversion complete flag, COCO, is set in the A/D status/control register (\$15).

Note: Any write to the A/D status/control register will abort the current conversion, reset the conversion complete flag and start a new conversion on the selected channel.

At power-on or external reset, both the ADRC and ADON bits are cleared, thus the A/D is disabled.

9.2 A/D registers

9.2.1 A/D status/control register (ADSTAT)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
A/D status/control register (ADSTAT)	\$0015	COCO	ADRC	ADON		CH3	CH2	CH1	CH0	0000 0000

9.2.1.1 COCO — Conversion complete flag

- 1 (set) — COCO flag is set each time a conversion is complete, allowing the new result to be read from the A/D result data register (\$16). The converter then starts a new conversion.
- 0 (clear) — COCO is cleared by reading the result data register or writing to the status/control register.

Reset clears the COCO flag.

9.2.1.2 ADRC — A/D RC oscillator control

The ADRC bit allows the user to control the A/D RC oscillator, which is used to provide a sufficiently high clock rate to the A/D to ensure accuracy when the chip is running at low speeds.

- 1 (set) — When the ADRC bit is set, the A/D RC oscillator is turned on and, if ADON is set, the A/D runs from the RC oscillator clock. See Table 9-1.
- 0 (clear) — When the ADRC bit is cleared, the A/D RC oscillator is turned-off and, if ADON is set, the A/D runs from the CPU clock.

When the A/D RC oscillator is turned on, it takes time t_{ADRC} to stabilize (see Section 12.4). During this time A/D conversion results may be inaccurate.

Power-on or external reset clears the ADRC bit.

9.2.1.3 ADON — A/D converter on

The ADON bit allows the user to enable/disable the A/D converter.

- 1 (set) — A/D converter is switched on.
- 0 (clear) — A/D converter is switched off.

When the A/D converter is switched on, it takes time t_{ADON} for the current sources to stabilize (see Section 12.4). During this time A/D conversion results may be inaccurate.

Power-on or external reset will clear the ADON bit, disabling the A/D converter.

Table 9-1 A/D clock selection

ADRC	ADON	RC oscillator	A/D converter	Comments
0	0	OFF	OFF	A/D switched off.
0	1	OFF	ON	A/D using CPU clock.
1	0	ON	OFF	Allows the RC oscillator to stabilise.
1	1	ON	ON	A/D using RC oscillator clock.

9.2.1.4 CH3 – CH0 — A/D channels 3, 2, 1 and 0

The CH3–CH0 bits allow the user to determine which channel of the A/D converter multiplexer is selected. See Table 9-2 for channel selection.

Reset clears these bits.

Table 9-2 A/D channel assignment

CH3	CH2	CH1	CH0	Channel selected
0	0	0	0	AD0
0	0	0	1	AD1
0	0	1	0	reserved
0	0	1	1	reserved
0	1	0	0	reserved
0	1	0	1	reserved
0	1	1	0	reserved
0	1	1	1	reserved
1	0	0	0	VRH pin (high)
1	0	0	1	(VRH + VRL) / 2
1	0	1	0	VRL pin (low)
1	0	1	1	VRL pin (low)
1	1	0	0	VRL pin (low)
1	1	0	1	VRL pin (low)
1	1	1	0	VRL pin (low)
1	1	1	1	VRL pin (low)

9.2.2 A/D input register (ADIN)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
A/D input register (ADIN)	\$0016							AD1	AD0	Undefined

The ADIN register allows the A/D input to be read as a static input. Reading this register during an A/D conversion sequence may inject noise into the analog input and reduce the accuracy of the A/D result.

Note: Performing a digital read of the A/D input with levels other than V_{DD} or V_{SS} on the ADIN pin will result in greater power dissipation during the read cycle. This will also give unpredictable results on the ADIN input.

Reset does not affect the ADIN bit.

9.2.3 A/D result data register (ADDATA)

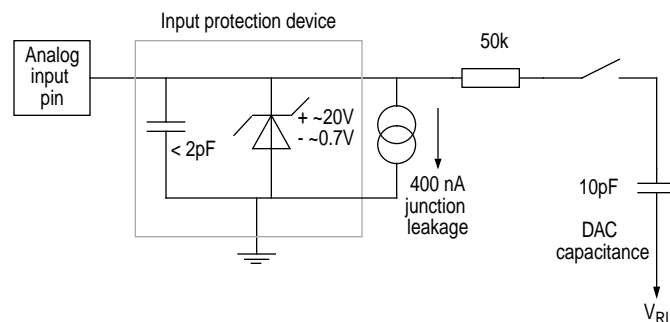
	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
A/D result data register (ADDATA)	\$0017									Undefined

ADDATA is a read-only register which is used to store the result of an A/D conversion. The result is loaded into the register from the SAR and the conversion complete flag in the ADSTAT register, COCO, is set.

9.3 ADx analog input

The external analog voltage value to be processed by the A/D converter is sampled on an internal capacitor through a resistive path, provided by input-selection switches and a sampling aperture time switch, as shown in Figure 9-2. Sampling time is limited to 12 bus clock cycles. After sampling, the analog value is stored on the capacitor and held until the end of conversion. During this hold time, the analog input is disconnected from the internal A/D system and the external voltage source sees a high impedance input.

The equivalent analog input during sampling is an RC low-pass filter with a minimum resistance of 50 k and a capacitance of at least 10pF (these are typical values measured at room temperature).



9

Note: The analog switch is closed during the 12 cycle sample time only.

Figure 9-2 Electrical model of an A/D input pin

10

RESETS AND INTERRUPTS

10.1 Resets

The MC68HC05L28 can be reset in three ways: by the initial power-on reset function, by an active low input to the $\overline{\text{RESET}}$ pin and by a COP watchdog timer reset, if the watchdog timer is enabled.

10.1.1 Power-on reset

A power-on reset occurs when a positive transition is detected on VDD. The power-on reset function is strictly for power turn-on conditions and should not be used to detect drops in the power supply voltage. The power-on circuitry provides a stabilization delay (t_{PORL}) from when the oscillator becomes active. If the external $\overline{\text{RESET}}$ pin is low at the end of this delay then the processor remains in the reset state until $\overline{\text{RESET}}$ goes high. The user must ensure that the voltage on VDD has risen to a point where the MCU can operate properly by the time t_{PORL} has elapsed. If there is doubt, the external $\overline{\text{RESET}}$ pin should remain low until the voltage on VDD has reached the specified minimum operating voltage. This may be accomplished by connecting an external RC-circuit to this pin to generate a power-on reset (POR). In this case, the time constant must be great enough (at least 100ms) to allow the oscillator circuit to stabilise.

10

10.1.2 $\overline{\text{RESET}}$ pin

When the oscillator is running in a stable state, the MCU is reset when a logic zero is applied to the $\overline{\text{RESET}}$ input for a minimum period of 1.5 machine cycles (t_{CYC}). This pin contains an internal Schmitt trigger as part of its input to improve noise immunity.

10.1.3 Computer operating properly (COP) reset

The MCU contains a watchdog timer that automatically times out if not reset (cleared) within a specific time by a program reset sequence.

Note: COP timeout is prevented by periodically writing a '0' to bit 0 of address \$3FF0.

If the COP watchdog timer is allowed to timeout, an internal reset is generated to reset the MCU. Because the internal reset signal is used, the MCU comes out of a COP reset in the same operating mode it was in when the COP timeout was generated.

The COP reset function is enabled or disabled by a bit in the option register.

See Section 5.2 for more information on the COP watchdog timer.

10.2 Interrupts

The MCU can be interrupted by different sources – six maskable hardware interrupts and one non-maskable software interrupt:

- External signal on the IRQ pins ($\overline{\text{IRQ0}}$, IRQ1, IRQ2)
- Core timer
- 16-bit programmable timer
- I²C
- Software Interrupt Instruction (SWI)

10

Interrupts cause the processor to save the register contents on the stack and to set the interrupt mask (I-bit) to prevent additional interrupts. The RTI instruction (return from interrupt) causes the register contents to be recovered from the stack and normal processing to resume.

Unlike reset, hardware interrupts do not cause the current instruction execution to be halted, but are considered pending until the current instruction is complete. The current instruction is the one already fetched and being operated on. When the current instruction is complete, the processor checks all pending hardware interrupts. If interrupts are not masked (CCR I-bit clear) and the corresponding interrupt enable bit is set, the processor proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

Table 10-1 shows the relative priority of all the possible interrupt sources. Figure 10-1 shows the interrupt processing flow.

Table 10-1 Interrupt priorities

Source	Register	Flags	Vector Address	Priority
Reset	—	—	\$3FFE, \$3FFF	highest ↑ lowest
Software interrupt (SWI)	—	—	\$3FFC, \$3FFD	
External interrupt (IRQ)	IRQx	IRQxINT	\$3FFA, \$3FFB	
Core timer	CTCSR	CTOF, RTIF	\$3FF8, \$3FF9	
I ² C	MSR	MIF	\$3FF6, \$3FF7	
Programmable timer	TSR	ICF, OCF, TOF	\$3FF4, \$3FF5	

10.2.1 Non-maskable software interrupt (SWI)

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

10.2.2 Maskable hardware interrupts

If the interrupt mask bit (I-bit) of the CCR is set, all maskable interrupts (internal and external) are masked. Clearing the I-bit allows interrupt processing to occur.

Note: The internal interrupt latch is cleared in the first part of the interrupt service routine; therefore, one external interrupt pulse could be latched and serviced as soon as the I-bit is cleared.

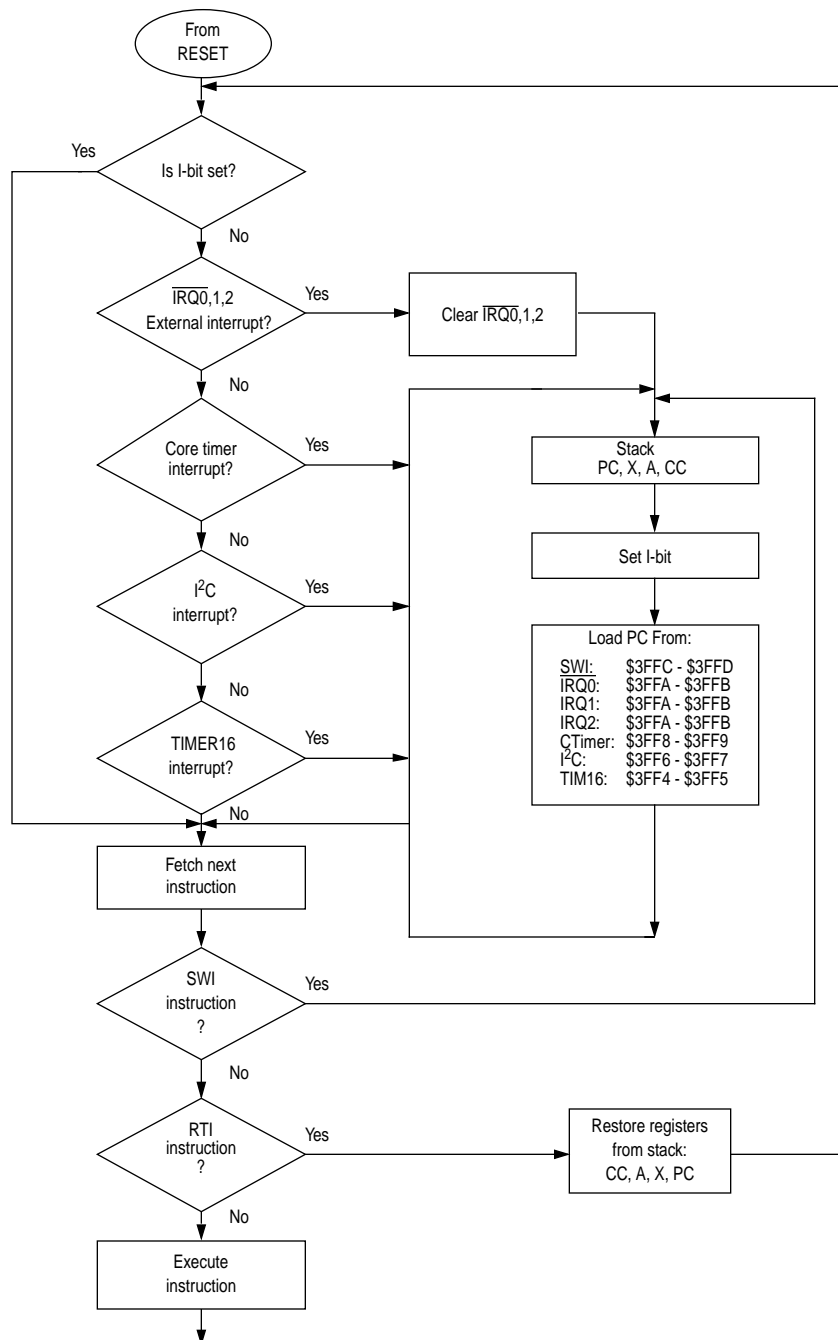


Figure 10-1 Interrupt flow chart

10.2.2.1 External interrupt ($\overline{\text{IRQ0}}$, IRQ1, IRQ2)

These external interrupt sources use the same interrupt vector (\$3FFA, \$3FFB)

$\overline{\text{IRQ0}}$

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 $\overline{\text{IRQ0}}$. It is then synchronized internally and serviced by the interrupt service routine located at the address specified by the contents of \$3FFA and \$3FFB.

Either a level-sensitive and edge-sensitive trigger, or an edge-sensitive-only trigger can be selected by bit-1 (IRQED) in the option register (\$1D). When IRQED is cleared, the interrupt is edge-and-level sensitive and when set the interrupt is edge sensitive. IRQED can be written to once only after a power-on-reset or external reset. This bit is cleared after reset.

IRQ1

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
IRQ1 status/control register (IRQ1)	\$000A			IRQ1INT	IRQ1ENA	IRQ1LV	IRQ1EDG	IRQ1RST	IRQ1VAL	??00 0000

This interrupt can be enabled independently and different sensitivities can be defined: falling edge, falling edge and low level, rising edge, rising edge and high level. The interrupt is enabled by setting bit 4 (IRQ1ENA) of register \$0A and is disabled by clearing it. The interrupt vector, \$3FFA and \$3FFB, is shared with the other IRQ interrupts. Bit 5 of register \$0A is an interrupt flag (IRQ1INT) which distinguishes between the interrupts and is set when an interrupt occurs. The interrupt is cleared by writing 1 to the IRQ1RST bit which always reads 0. The status of IRQ1 can be monitored by reading the IRQ1VAL bit (bit 0 on the IRQ1 register).

IRQ1INT — IRQ1 interrupt flag

- 1 (set) — A valid IRQ1 interrupt has been generated.
- 0 (clear) — No valid IRQ1 interrupt has been generated.

IRQ1ENA — IRQ1 interrupt enable

- 1 (set) — IRQ1 interrupts are enabled.
- 0 (clear) — IRQ1 interrupts are disabled.

IRQ1LV, IRQ1EDG — IRQ1 interrupt sensitivity bits

These two bits are used to select the sensitivity of the IRQ1 interrupt trigger according to Table 10-2.

Table 10-2 IRQ1 interrupt sensitivity

IRQ1LV	IRQ1EDG	Interrupt sensitivity
0	0	Falling edge
0	1	Rising edge
1	0	Falling edge and low level
1	1	Rising edge and high level

IRQ1RST — IRQ1 reset

The IRQ1 interrupt is cleared by writing a '1' to this bit. This bit is write-only and always returns zero.

IRQ1VAL — IRQ1 pin status

The IRQ1VAL bit reflects current status of the IRQ1 pin.

IRQ2

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
IRQ2 status/control register (IRQ2)	\$000B			IRQ2INT	IRQ2ENA	IRQ2LV	IRQ2EDG	IRQ2RST	IRQ2VAL	??00 0000

This interrupt can be enabled independently and different sensitivities can be defined: falling edge, falling edge and low level, rising edge, rising edge and high level. The interrupt is enabled by setting bit 4 (IRQ2ENA) of register \$0B and is disabled by clearing it. The interrupt vector, \$3FFA and \$3FFB, is shared with the other IRQ interrupts. Bit 5 of register \$0B is an interrupt flag (IRQ2INT) which distinguishes between the interrupts and is set when an interrupt occurs. The interrupt is cleared by writing 1 to the IRQ2RST bit which always reads 0. The status of IRQ2 can be monitored by reading the IRQ2VAL bit (bit 0 on the IRQ2 register).

10**IRQ2INT — IRQ2 interrupt flag**

- 1 (set) — A valid IRQ2 interrupt has been generated.
- 0 (clear) — No valid IRQ2 interrupt has been generated.

IRQ2ENA — IRQ2 interrupt enable

- 1 (set) — IRQ2 interrupts are enabled.
- 0 (clear) — IRQ2 interrupts are disabled.

IRQ2LV, IRQ2EDG — IRQ2 interrupt sensitivity bits

These two bits are used to select the sensitivity of the IRQ2 interrupt trigger according to Table 10-2.

Table 10-3 IRQ2 interrupt sensitivity

IRQ2LV	IRQ2EDG	Interrupt sensitivity
0	0	Falling edge
0	1	Rising edge
1	0	Falling edge and low level
1	1	Rising edge and high level

IRQ2RST — IRQ2 reset

The IRQ2 interrupt is cleared by writing a '1' to this bit. This bit is write-only and always returns zero.

IRQ2VAL — IRQ2 pin status

The IRQ2VAL bit reflects current status of the IRQ2 pin.

10.2.2.2 Real time and core timer (CTIMER) interrupts

There are two core timer interrupt flags that cause a CTIMER interrupt whenever an interrupt is enabled and its flag becomes set (RTIF and CTOF). The interrupt flags and enable bits are located in the CTIMER control and status register (CTCSR). These interrupts vector to the same interrupt service routine, whose start address is contained in memory locations \$3FF8 and \$3FF9 (see Section 5.3.1 and Figure 5-1).

To make use of the real time interrupt the RTIE bit must first be set. The RTIF bit will then be set after the specified number of counts.

To make use of the core timer overflow interrupt, the CTOFE bit must first be set. The CTOF bit will then be set when the core timer counter register overflows from \$FF to \$00.

10

10.2.2.3 Programmable 16-bit timer interrupt

There are five interrupt flags that cause a timer interrupt when set and enabled. The timer interrupt enable bits are located in the timer control registers (TCR) and the timer interrupt flags are located in the timer status registers (TSR). All interrupts vector to the same service routine, whose start address is contained in memory locations \$3FF4 and \$3FF5. In WAIT mode the CPU clock halts but the timer continues to run.

10.2.2.4 I²C interrupts

There is an interrupt flag and three status flags for the I²C that cause an I²C interrupt when set and enabled. These interrupts will vector to the service routine located at the address specified by the contents of memory locations \$3FF6 and \$3FF7.

10.2.3 Hardware controlled interrupt sequence

The following three functions (RESET, STOP, and WAIT) are not in the strictest sense interrupts. However, they are acted upon in a similar manner. Flowcharts for STOP and WAIT are shown in Section 2-5 and Figure 2-6.

RESET: A reset condition causes the program to vector to its starting address, which is contained in memory locations \$3FFE (MSB) and \$3FFF (LSB). The I-bit in the condition code register is also set, to disable maskable interrupts.

STOP: The STOP instruction causes the oscillator to be turned off and the processor to 'sleep' until an external interrupt ($\overline{\text{IRQ}}$) interrupt occurs, or the device is reset.

WAIT: The WAIT instruction causes all processor clocks to stop, but leaves the core timer clock running. This 'rest' state of the processor can be cleared by reset, an external interrupt ($\overline{\text{IRQ}}$), or a timer interrupt. There are no special WAIT vectors for these interrupts.

11

CPU CORE AND INSTRUCTION SET

This section provides a description of the CPU core registers, the instruction set and the addressing modes of the MC68HC05L28.

11.1 Registers

The MCU contains five registers, as shown in the programming model of Figure 11-1. The interrupt stacking order is shown in Figure 11-2.

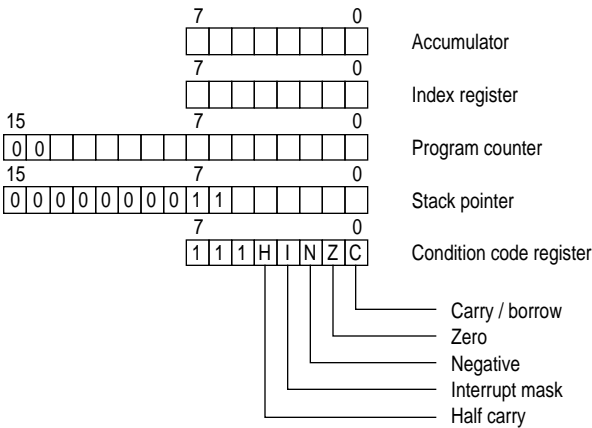


Figure 11-1 Programming model

11.1.1 Accumulator (A)

The accumulator is a general purpose 8-bit register used to hold operands and results of arithmetic calculations or data manipulations.

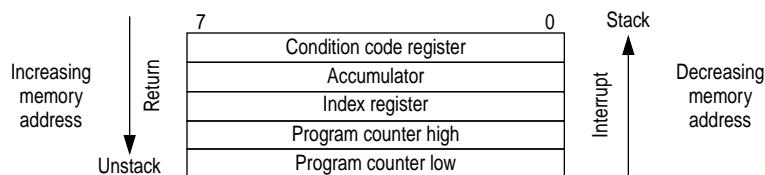


Figure 11-2 Stacking order

11.1.2 Index register (X)

The index register is an 8-bit register, which can contain the indexed addressing value used to create an effective address. The index register may also be used as a temporary storage area.

11.1.3 Program counter (PC)

The program counter is a 16-bit register, which contains the address of the next byte to be fetched. Although the M68HC05 CPU core can address 64K bytes of memory, the actual address range of the MC68HC05L28 is limited to 16K bytes. The two most significant bits of the program counter are therefore not used and are permanently set to zero.

11.1.4 Stack pointer (SP)

The stack pointer is a 16-bit register, which contains the address of the next free location on the stack. During an MCU reset or the reset stack pointer (RSP) instruction, the stack pointer is set to location \$00FF. The stack pointer is then decremented as data is pushed onto the stack and incremented as data is pulled from the stack.

11

When accessing memory, the ten most significant bits are permanently set to 0000000011. These ten bits are appended to the six least significant register bits to produce an address within the range of \$00C0 to \$00FF. Subroutines and interrupts may use up to 64 (decimal) locations. If 64 locations are exceeded, the stack pointer wraps around and overwrites the previously stored information. A subroutine call occupies two locations on the stack; an interrupt uses five locations.

11.1.5 Condition code register (CCR)

The CCR is a 5-bit register in which four bits are used to indicate the results of the instruction just executed, and the fifth bit indicates whether interrupts are masked. These bits can be individually tested by a program, and specific actions can be taken as a result of their state. Each bit is explained in the following paragraphs.

Half carry (H)

This bit is set during ADD and ADC operations to indicate that a carry occurred between bits 3 and 4.

Interrupt (I)

When this bit is set, all maskable interrupts are masked. If an interrupt occurs while this bit is set, the interrupt is latched and remains pending until the interrupt bit is cleared.

Negative (N)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation was negative.

Zero (Z)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation was zero.

Carry/borrow (C)

When set, this bit indicates that a carry or borrow out of the arithmetic logical unit (ALU) occurred during the last arithmetic operation. This bit is also affected during bit test and branch instructions and during shifts and rotates.

11.2 Instruction set

The MCU has a set of 62 basic instructions. They can be grouped into five different types as follows:

- Register/memory
- Read/modify/write
- Branch
- Bit manipulation
- Control

11

The following paragraphs briefly explain each type. All the instructions within a given type are presented in individual tables.

This MCU uses all the instructions available in the M146805 CMOS family plus one more: the unsigned multiply (MUL) instruction. This instruction allows unsigned multiplication of the contents of the accumulator (A) and the index register (X). The high-order product is then stored in the index register and the low-order product is stored in the accumulator. A detailed definition of the MUL instruction is shown in Table 11-1.

11.2.1 Register/memory Instructions

Most of these instructions use two operands. The first operand is either the accumulator or the index register. The second operand is obtained from memory using one of the addressing modes. The jump unconditional (JMP) and jump to subroutine (JSR) instructions have no register operand. Refer to Table 11-2 for a complete list of register/memory instructions.

11.2.2 Branch instructions

These instructions cause the program to branch if a particular condition is met; otherwise, no operation is performed. Branch instructions are two-byte instructions. Refer to Table 11-3.

11.2.3 Bit manipulation instructions

The MCU can set or clear any writable bit that resides in the first 256 bytes of the memory space (page 0). All port data and data direction registers, timer and serial interface registers, control/status registers and a portion of the on-chip RAM reside in page 0. An additional feature allows the software to test and branch on the state of any bit within these locations. The bit set, bit clear, bit test and branch functions are all implemented with single instructions. For the test and branch instructions, the value of the bit tested is also placed in the carry bit of the condition code register. Refer to Table 11-4.

11.2.4 Read/modify/write instructions

These instructions read a memory location or a register, modify or test its contents, and write the modified value back to memory or to the register. The test for negative or zero (TST) instruction is an exception to this sequence of reading, modifying and writing, since it does not modify the value. Refer to Table 11-5 for a complete list of read/modify/write instructions.

11.2.5 Control instructions

These instructions are register reference instructions and are used to control processor operation during program execution. Refer to Table 11-6 for a complete list of control instructions.

11.2.6 Tables

Tables for all the instruction types listed above follow. In addition there is a complete alphabetical listing of all the instructions (see Table 11-7), and an opcode map for the instruction set of the M68HC05 MCU family (see Table 11-8).

Table 11-1 MUL instruction

Operation	$X:A \leftarrow X \cdot A$			
Description	Multiplies the eight bits in the index register by the eight bits in the accumulator and places the 16-bit result in the concatenated accumulator and index register.			
Condition codes	H : Cleared I : Not affected N : Not affected Z : Not affected C : Cleared			
Source	MUL			
Form	Addressing mode Inherent	Cycles 11	Bytes 1	Opcode \$42

Table 11-2 Register/memory instructions

Function	Mnemonic	Addressing modes																	
		Immediate			Direct			Extended			Indexed (no offset)			Indexed (8-bit offset)			Indexed (16-bit offset)		
		Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles
Load A from memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5
Load X from memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5
Store A in memory	STA				B7	2	4	C7	3	5	F7	1	4	E7	2	5	D7	3	6
Store X in memory	STX				BF	2	4	CF	3	5	FF	1	4	EF	2	5	DF	3	6
Add memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5
Add memory and carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5
Subtract memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5
Subtract memory from A with borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5
AND memory with A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5
OR memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5
Exclusive OR memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5
Arithmetic compare A with memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5
Arithmetic compare X with memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5
Bit test memory with A (logical compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5
Jump unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4
Jump to subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	6	DD	3	7

Table 11-3 Branch instructions

Function	Mnemonic	Relative addressing mode		
		Opcode	# Bytes	# Cycles
Branch always	BRA	20	2	3
Branch never	BRN	21	2	3
Branch if higher	BHI	22	2	3
Branch if lower or same	BLS	23	2	3
Branch if carry clear	BCC	24	2	3
(Branch if higher or same)	(BHS)	24	2	3
Branch if carry set	BCS	25	2	3
(Branch if lower)	(BLO)	25	2	3
Branch if not equal	BNE	26	2	3
Branch if equal	BEQ	27	2	3
Branch if half carry clear	BHCC	28	2	3
Branch if half carry set	BHCS	29	2	3
Branch if plus	BPL	2A	2	3
Branch if minus	BMI	2B	2	3
Branch if interrupt mask bit is clear	BMC	2C	2	3
Branch if interrupt mask bit is set	BMS	2D	2	3
Branch if interrupt line is low	BIL	2E	2	3
Branch if interrupt line is high	BIH	2F	2	3
Branch to subroutine	BSR	AD	2	6

Table 11-4 Bit manipulation instructions

Function	Mnemonic	Addressing modes					
		Bit set/clear			Bit test and branch		
		Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles
Branch if bit n is set	BRSET n (n=0–7)				2•n	3	5
Branch if bit n is clear	BRCLR n (n=0–7)				01+2•n	3	5
Set bit n	BSET n (n=0–7)	10+2•n	2	5			
Clear bit n	BCLR n (n=0–7)	11+2•n	2	5			

Table 11-5 Read/modify/write instructions

Function	Mnemonic	Addressing modes														
		Inherent (A)			Inherent (X)			Direct			Indexed (no offset)			Indexed (8-bit offset)		
		Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles
Increment	INC	4C	1	3	5C	1	3	3C	2	5	7C	1	5	6C	2	6
Decrement	DEC	4A	1	3	5A	1	3	3A	2	5	7A	1	5	6A	2	6
Clear	CLR	4F	1	3	5F	1	3	3F	2	5	7F	1	5	6F	2	6
Complement	COM	43	1	3	53	1	3	33	2	5	73	1	5	63	2	6
Negate (two's complement)	NEG	40	1	3	50	1	3	30	2	5	70	1	5	60	2	6
Rotate left through carry	ROL	49	1	3	59	1	3	39	2	5	79	1	5	69	2	6
Rotate right through carry	ROR	46	1	3	56	1	3	36	2	5	76	1	5	66	2	6
Logical shift left	LSL	48	1	3	58	1	3	38	2	5	78	1	5	68	2	6
Logical shift right	LSR	44	1	3	54	1	3	34	2	5	74	1	5	64	2	6
Arithmetic shift right	ASR	47	1	3	57	1	3	37	2	5	77	1	5	67	2	6
Test for negative or zero	TST	4D	1	3	5D	1	3	3D	2	4	7D	1	4	6D	2	5
Multiply	MUL	42	1	11												

Table 11-6 Control instructions


Function	Mnemonic	Inherent addressing mode		
		Opcode	# Bytes	# Cycles
Transfer A to X	TAX	97	1	2
Transfer X to A	TXA	9F	1	2
Set carry bit	SEC	99	1	2
Clear carry bit	CLC	98	1	2
Set interrupt mask bit	SEI	9B	1	2
Clear interrupt mask bit	CLI	9A	1	2
Software interrupt	SWI	83	1	10
Return from subroutine	RTS	81	1	6
Return from interrupt	RTI	80	1	9
Reset stack pointer	RSP	9C	1	2
No-operation	NOP	9D	1	2
Stop	STOP	8E	1	2
Wait	WAIT	8F	1	2

Table 11-7 Instruction set

Mnemonic	Addressing modes										Condition codes				
	INH	IMM	DIR	EXT	REL	IX	IX1	IX2	BSC	BTB	H	I	N	Z	C
ADC												•			
ADD												•			
AND											•	•			•
ASL											•	•			
ASR											•	•			
BCC											•	•	•	•	•
BCLR											•	•	•	•	•
BCS											•	•	•	•	•
BEQ											•	•	•	•	•
BHCC											•	•	•	•	•
BHCS											•	•	•	•	•
BHI											•	•	•	•	•
BHS											•	•	•	•	•
BIH											•	•	•	•	•
BIL											•	•	•	•	•
BIT											•	•			•
BLO											•	•	•	•	•
BLS											•	•	•	•	•
BMC											•	•	•	•	•
BMI											•	•	•	•	•
BMS											•	•	•	•	•
BNE											•	•	•	•	•
BPL											•	•	•	•	•
BRA											•	•	•	•	•
BRN											•	•	•	•	•
BRCLR											•	•	•	•	
BRSET											•	•	•	•	
BSET											•	•	•	•	•
BSR											•	•	•	•	•
CLC											•	•	•	•	0
CLI											•	0	•	•	•
CLR											•	•	0	1	•
CMP											•	•			

Address mode abbreviations

BSC	Bit set/clear	IMM	Immediate
BTB	Bit test & branch	IX	Indexed (no offset)
DIR	Direct	IX1	Indexed, 1 byte offset
EXT	Extended	IX2	Indexed, 2 byte offset
INH	Inherent	REL	Relative

 Not implemented

Condition code symbols


H	Half carry (from bit 3)	Tested and set if true, cleared otherwise
I	Interrupt mask	• Not affected
N	Negate (sign bit)	? Load CCR from stack
Z	Zero	0 Cleared
C	Carry/borrow	1 Set

Table 11-7 Instruction set (Continued)

Mnemonic	Addressing modes										Condition codes				
	INH	IMM	DIR	EXT	REL	IX	IX1	IX2	BSC	BTB	H	I	N	Z	C
COM											•	•			1
CPX											•	•			
DEC											•	•			•
EOR											•	•			•
INC											•	•			•
JMP											•	•	•	•	•
JSR											•	•	•	•	•
LDA											•	•			•
LDX											•	•			•
LSL											•	•			
LSR											•	•	0		
MUL											0	•	•	•	0
NEG											•	•			
NOP											•	•	•	•	•
ORA											•	•			•
ROL											•	•			
ROR											•	•			
RSP											•	•	•	•	•
RTI											?	?	?	?	?
RTS											•	•	•	•	•
SBC											•	•			
SEC											•	•	•	•	1
SEI											•	1	•	•	•
STA											•	•			•
STOP											•	0	•	•	•
STX											•	•			•
SUB											•	•			
SWI											•	1	•	•	•
TAX											•	•	•	•	•
TST											•	•			•
TXA											•	•	•	•	•
WAIT											•	0	•	•	•

Address mode abbreviations

BSC	Bit set/clear	IMM	Immediate
BTB	Bit test & branch	IX	Indexed (no offset)
DIR	Direct	IX1	Indexed, 1 byte offset
EXT	Extended	IX2	Indexed, 2 byte offset
INH	Inherent	REL	Relative

 Not implemented

Condition code symbols

H	Half carry (from bit 3)	Tested and set if true, cleared otherwise
I	Interrupt mask	• Not affected
N	Negate (sign bit)	? Load CCR from stack
Z	Zero	0 Cleared
C	Carry/borrow	1 Set

Table 11-8 M68HC05 opcode map

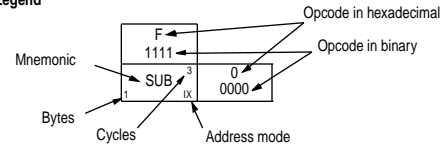
	Bit manipulation		Branch	Read/modify/write					Control		Register/memory							
	BTB	BSC	REL	DIR	INH	INH	IX1	IX	INH	INH	IMM	DIR	EXT	IX2	IX1	IX		
High	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	High	
Low	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	Low	
0	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BRA ³ _{REL} ²	NEG ⁵ _{DIR} ¹	NEGA ³ _{INH} ¹	NEGX ³ _{INH} ²	NEG ⁶ _{IX1} ¹	NEG ⁵ _{IX} ¹	RTI ⁹ _{INH} ⁶		SUB ² _{IMM} ²	SUB ³ _{DIR} ³	SUB ⁴ _{EXT} ³	SUB ⁵ _{IX2} ²	SUB ⁴ _{IX1} ¹	SUB ³ _{IX} ¹	0	
0000																		0000
1	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BRN ³ _{REL} ²						RTS ⁶ _{INH} ¹		CMP ² _{IMM} ²	CMP ³ _{DIR} ³	CMP ⁴ _{EXT} ³	CMP ⁵ _{IX2} ²	CMP ⁴ _{IX1} ¹	CMP ³ _{IX} ¹	1	
0001																		0001
2	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BHI ³ _{REL} ²		MUL ¹¹ _{INH} ¹						SBC ² _{IMM} ²	SBC ³ _{DIR} ³	SBC ⁴ _{EXT} ³	SBC ⁵ _{IX2} ²	SBC ⁴ _{IX1} ¹	SBC ³ _{IX} ¹	2	
0010																		0010
3	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BLS ³ _{REL} ²	COM ³ _{DIR} ¹	COMA ³ _{INH} ¹	COMX ³ _{INH} ²	COM ⁶ _{IX1} ¹	COM ⁵ _{IX} ¹	SWI ¹⁰ _{INH} ¹		CPX ² _{IMM} ²	CPX ³ _{DIR} ³	CPX ⁴ _{EXT} ³	CPX ⁵ _{IX2} ²	CPX ⁴ _{IX1} ¹	CPX ³ _{IX} ¹	3	
0011																		0011
4	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BCC ³ _{REL} ²	LSR ⁵ _{DIR} ¹	LSRA ³ _{INH} ¹	LSRX ³ _{INH} ²	LSR ⁶ _{IX1} ¹	LSR ⁵ _{IX} ¹			AND ² _{IMM} ²	AND ³ _{DIR} ³	AND ⁴ _{EXT} ³	AND ⁵ _{IX2} ²	AND ⁴ _{IX1} ¹	AND ³ _{IX} ¹	4	
0100																		0100
5	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BCS ³ _{REL} ²								BIT ² _{IMM} ²	BIT ³ _{DIR} ³	BIT ⁴ _{EXT} ³	BIT ⁵ _{IX2} ²	BIT ⁴ _{IX1} ¹	BIT ³ _{IX} ¹	5	
0101																		0101
6	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BNE ³ _{REL} ²	ROR ⁵ _{DIR} ¹	RORA ³ _{INH} ¹	RORX ³ _{INH} ²	ROR ⁶ _{IX1} ¹	ROR ⁵ _{IX} ¹			LDA ² _{IMM} ²	LDA ³ _{DIR} ³	LDA ⁴ _{EXT} ³	LDA ⁵ _{IX2} ²	LDA ⁴ _{IX1} ¹	LDA ³ _{IX} ¹	6	
0110																		0110
7	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BEQ ³ _{REL} ²	ASR ⁵ _{DIR} ¹	ASRA ³ _{INH} ¹	ASRX ³ _{INH} ²	ASR ⁶ _{IX1} ¹	ASR ⁵ _{IX} ¹	TAX ² _{INH} ¹			STA ⁴ _{DIR} ³	STA ⁵ _{EXT} ³	STA ⁶ _{IX2} ²	STA ⁵ _{IX1} ¹	STA ⁴ _{IX} ¹	7	
0111																		0111
8	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BHCC ³ _{REL} ²	LSL ⁵ _{DIR} ¹	LSLA ³ _{INH} ¹	LSLX ³ _{INH} ²	LSL ⁶ _{IX1} ¹	LSL ⁵ _{IX} ¹	CLC ² _{INH} ¹		EOR ² _{IMM} ²	EOR ³ _{DIR} ³	EOR ⁴ _{EXT} ³	EOR ⁵ _{IX2} ²	EOR ⁴ _{IX1} ¹	EOR ³ _{IX} ¹	8	
1000																		1000
9	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BHCS ³ _{REL} ²	ROL ⁵ _{DIR} ¹	ROLA ³ _{INH} ¹	ROLX ³ _{INH} ²	ROL ⁶ _{IX1} ¹	ROL ⁵ _{IX} ¹	SEC ² _{INH} ¹		ADC ² _{IMM} ²	ADC ³ _{DIR} ³	ADC ⁴ _{EXT} ³	ADC ⁵ _{IX2} ²	ADC ⁴ _{IX1} ¹	ADC ³ _{IX} ¹	9	
1001																		1001
A	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BPL ³ _{REL} ²	DEC ⁵ _{DIR} ¹	DECA ³ _{INH} ¹	DECX ³ _{INH} ²	DEC ⁶ _{IX1} ¹	DEC ⁵ _{IX} ¹	CLI ² _{INH} ¹		ORA ² _{IMM} ²	ORA ³ _{DIR} ³	ORA ⁴ _{EXT} ³	ORA ⁵ _{IX2} ²	ORA ⁴ _{IX1} ¹	ORA ³ _{IX} ¹	A	
1010																		1010
B	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BMI ³ _{REL} ²						SEI ² _{INH} ¹		ADD ² _{IMM} ²	ADD ³ _{DIR} ³	ADD ⁴ _{EXT} ³	ADD ⁵ _{IX2} ²	ADD ⁴ _{IX1} ¹	ADD ³ _{IX} ¹	B	
1011																		1011
C	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BMC ³ _{REL} ²	INC ⁵ _{DIR} ¹	INCA ³ _{INH} ¹	INCX ³ _{INH} ²	INC ⁶ _{IX1} ¹	INC ⁵ _{IX} ¹	RSP ² _{INH} ¹		JMP ² _{DIR} ³	JMP ³ _{EXT} ³	JMP ⁴ _{IX2} ²	JMP ⁵ _{IX1} ¹	JMP ⁴ _{IX} ¹	JMP ³ _{IX} ¹	C	
1100																		1100
D	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BMS ³ _{REL} ²	TST ⁴ _{DIR} ¹	TSTA ³ _{INH} ¹	TSTX ³ _{INH} ²	TST ⁵ _{IX1} ¹	TST ⁴ _{IX} ¹	NOP ² _{INH} ¹		BSR ⁶ _{REL} ²	JSR ⁶ _{DIR} ³	JSR ⁷ _{EXT} ³	JSR ⁷ _{IX2} ²	JSR ⁶ _{IX1} ¹	JSR ⁵ _{IX} ¹	D	
1101																		1101
E	BRSET ⁵ _{BTB} ²	BSET ⁵ _{BSC} ²	BIL ³ _{REL} ²						STOP ² _{INH} ¹		LDX ² _{IMM} ²	LDX ³ _{DIR} ³	LDX ⁴ _{EXT} ³	LDX ⁵ _{IX2} ²	LDX ⁴ _{IX1} ¹	LDX ³ _{IX} ¹	E	
1110																		1110
F	BRCLR ⁵ _{BTB} ²	BCLR ⁵ _{BSC} ²	BIH ³ _{REL} ²	CLR ⁵ _{DIR} ¹	CLRA ³ _{INH} ¹	CLR ³ _{INH} ²	CLR ⁶ _{IX1} ¹	CLR ⁵ _{IX} ¹	WAIT ² _{INH} ¹		TXA ² _{INH} ¹		STX ⁴ _{DIR} ³	STX ⁵ _{EXT} ³	STX ⁶ _{IX2} ²	STX ⁵ _{IX1} ¹	STX ⁴ _{IX} ¹	F
1111																		1111

Abbreviations for address modes and registers

BSC	Bit set/clear	IX	Indexed (no offset)
BTB	Bit test and branch	IX1	Indexed, 1 byte (8-bit) offset
DIR	Direct	IX2	Indexed, 2 byte (16-bit) offset
EXT	Extended	REL	Relative
INH	Inherent	A	Accumulator
IMM	Immediate	X	Index register

 Not implemented

Legend



11.3 Addressing modes

Ten different addressing modes provide programmers with the flexibility to optimize their code for all situations. The various indexed addressing modes make it possible to locate data tables, code conversion tables and scaling tables anywhere in the memory space. Short indexed accesses are single byte instructions; the longest instructions (three bytes) enable access to tables throughout memory. Short absolute (direct) and long absolute (extended) addressing are also included. One or two byte direct addressing instructions access all data bytes in most applications. Extended addressing permits jump instructions to reach all memory locations.

The term 'effective address' (EA) is used in describing the various addressing modes. The effective address is defined as the address from which the argument for an instruction is fetched or stored. The ten addressing modes of the processor are described below. Parentheses are used to indicate 'contents of' the location or register referred to. For example, (PC) indicates the contents of the location pointed to by the PC (program counter). An arrow indicates 'is replaced by' and a colon indicates concatenation of two bytes. For additional details and graphical illustrations, refer to the **M6805 HMOS/M146805 CMOS Family Microcomputer/ Microprocessor User's Manual** or to the **M68HC05 Applications Guide**.

11.3.1 Inherent

In the inherent addressing mode, all the information necessary to execute the instruction is contained in the opcode. Operations specifying only the index register or accumulator, as well as the control instruction, with no other arguments are included in this mode. These instructions are one byte long.

11.3.2 Immediate

In the immediate addressing mode, the operand is contained in the byte immediately following the opcode. The immediate addressing mode is used to access constants that do not change during program execution (e.g. a constant used to initialize a loop counter).

$$EA = PC+1; PC \leftarrow PC+2$$

11

11.3.3 Direct

In the direct addressing mode, the effective address of the argument is contained in a single byte following the opcode byte. Direct addressing allows the user to directly address the lowest 256 bytes in memory with a single two-byte instruction.

$$EA = (PC+1); PC \leftarrow PC+2$$
$$\text{Address bus high} \leftarrow 0; \text{Address bus low} \leftarrow (PC+1)$$

11.3.4 Extended

In the extended addressing mode, the effective address of the argument is contained in the two bytes following the opcode byte. Instructions with extended addressing mode are capable of referencing arguments anywhere in memory with a single three-byte instruction. When using the Motorola assembler, the user need not specify whether an instruction uses direct or extended addressing. The assembler automatically selects the short form of the instruction.

$$\begin{aligned} \text{EA} &= (\text{PC}+1):(\text{PC}+2); \text{PC} \leftarrow \text{PC}+3 \\ \text{Address bus high} &\leftarrow (\text{PC}+1); \text{Address bus low} \leftarrow (\text{PC}+2) \end{aligned}$$

11.3.5 Indexed, no offset

In the indexed, no offset addressing mode, the effective address of the argument is contained in the 8-bit index register. This addressing mode can access the first 256 memory locations. These instructions are only one byte long. This mode is often used to move a pointer through a table or to hold the address of a frequently referenced RAM or I/O location.

$$\begin{aligned} \text{EA} &= \text{X}; \text{PC} \leftarrow \text{PC}+1 \\ \text{Address bus high} &\leftarrow 0; \text{Address bus low} \leftarrow \text{X} \end{aligned}$$

11.3.6 Indexed, 8-bit offset

In the indexed, 8-bit offset addressing mode, the effective address is the sum of the contents of the unsigned 8-bit index register and the unsigned byte following the opcode. Therefore the operand can be located anywhere within the lowest 511 memory locations. This addressing mode is useful for selecting the *m*th element in an *n* element table.

$$\begin{aligned} \text{EA} &= \text{X} + (\text{PC}+1); \text{PC} \leftarrow \text{PC}+2 \\ \text{Address bus high} &\leftarrow \text{K}; \text{Address bus low} \leftarrow \text{X} + (\text{PC}+1) \\ \text{where K} &= \text{the carry from the addition of X and (PC+1)} \end{aligned}$$

11

11.3.7 Indexed, 16-bit offset

In the indexed, 16-bit offset addressing mode, the effective address is the sum of the contents of the unsigned 8-bit index register and the two unsigned bytes following the opcode. This address mode can be used in a manner similar to indexed, 8-bit offset except that this three-byte instruction allows tables to be anywhere in memory. As with direct and extended addressing, the Motorola assembler determines the shortest form of indexed addressing.

$$\begin{aligned} \text{EA} &= \text{X} + [(\text{PC}+1):(\text{PC}+2)]; \text{PC} \leftarrow \text{PC}+3 \\ \text{Address bus high} &\leftarrow (\text{PC}+1) + \text{K}; \text{Address bus low} \leftarrow \text{X} + (\text{PC}+2) \\ \text{where K} &= \text{the carry from the addition of X and (PC+2)} \end{aligned}$$

11.3.8 Relative

The relative addressing mode is only used in branch instructions. In relative addressing, the contents of the 8-bit signed byte (the offset) following the opcode are added to the PC if, and only if, the branch conditions are true. Otherwise, control proceeds to the next instruction. The span of relative addressing is from -126 to +129 from the opcode address. The programmer need not calculate the offset when using the Motorola assembler, since it calculates the proper offset and checks to see that it is within the span of the branch.

$$\begin{aligned} \text{EA} &= \text{PC}+2+(\text{PC}+1); \text{PC} \leftarrow \text{EA} \text{ if branch taken;} \\ &\text{otherwise EA} = \text{PC} \leftarrow \text{PC}+2 \end{aligned}$$

11.3.9 Bit set/clear

In the bit set/clear addressing mode, the bit to be set or cleared is part of the opcode. The byte following the opcode specifies the address of the byte in which the specified bit is to be set or cleared. Any read/write bit in the first 256 locations of memory, including I/O, can be selectively set or cleared with a single two-byte instruction.

$$\begin{aligned} \text{EA} &= (\text{PC}+1); \text{PC} \leftarrow \text{PC}+2 \\ \text{Address bus high} &\leftarrow 0; \text{Address bus low} \leftarrow (\text{PC}+1) \end{aligned}$$

11.3.10 Bit test and branch

The bit test and branch addressing mode is a combination of direct addressing and relative addressing. The bit to be tested and its condition (set or clear) is included in the opcode. The address of the byte to be tested is in the single byte immediately following the opcode byte (EA1). The signed relative 8-bit offset in the third byte (EA2) is added to the PC if the specified bit is set or cleared in the specified memory location. This single three-byte instruction allows the program to branch based on the condition of any readable bit in the first 256 locations of memory. The span of branch is from -125 to +130 from the opcode address. The state of the tested bit is also transferred to the carry bit of the condition code register.

$$\begin{aligned} \text{EA1} &= (\text{PC}+1); \text{PC} \leftarrow \text{PC}+2 \\ \text{Address bus high} &\leftarrow 0; \text{Address bus low} \leftarrow (\text{PC}+1) \\ \text{EA2} &= \text{PC}+3+(\text{PC}+2); \text{PC} \leftarrow \text{EA2} \text{ if branch taken;} \\ &\text{otherwise PC} \leftarrow \text{PC}+3 \end{aligned}$$

THIS PAGE LEFT BLANK INTENTIONALLY

12

ELECTRICAL SPECIFICATIONS

This section contains the electrical specifications and associated timing information for the MC68HC05L28.

12.1 Maximum ratings

Table 12-1 Maximum ratings

Rating	Symbol	Value	Unit
Supply voltage ⁽¹⁾	V_{DD}	- 0.3 to +7.0	V
Input voltage: Normal operations Bootloader mode (IRQ0 pin only)	V_{IN}	$V_{SS} - 0.3$ to $V_{DD} + 0.3$ $V_{SS} - 0.3$ to $2 \times V_{DD} + 0.3$	V
Current sink into port B	I_B	80	mA
Operating temperature range MC68HC05L28 (standard)	T_A	T_L to T_H -40 to +85	C
Storage temperature range	T_{STG}	- 65 to +150	C
Current drain per pin ⁽²⁾ - excluding VDD and VSS	I_D	25	mA

(1) All voltages are with respect to V_{SS} .

(2) Maximum current drain per pin is for one pin at a time, limited by an external resistor.

Note: This device contains circuitry designed to protect against damage due to high electrostatic voltages or electric fields. However, it is recommended that normal precautions be taken to avoid the application of any voltages higher than those given in the maximum ratings table to this high impedance circuit. For maximum reliability all unused inputs should be tied to either V_{SS} or V_{DD} .

12

12.2 Thermal characteristics and power considerations

Table 12-2

The average chip junction temperature, T_J , in degrees Celcius can be obtained from the following equation:

[1]

where:

T_A = Ambient Temperature (C)

θ_{JA} = Package Thermal Resistance, Junction-to-ambient (C/W)

$P_D = P_{INT} + P_{I/O}$ (W)

P_{INT} = Internal Chip Power = $I_{DD} \cdot V_{DD}$ (W)

$P_{I/O}$ = Power Dissipation on Input and Output pins (User determined)

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

[2]

Solving equations [1] and [2] for K gives:

[3]

where K is a constant for a particular part. K can be determined 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 for any value of T_A by solving the above equations. The package thermal characteristics are shown in Table 12-2.