

USER'S MANUAL

S3C72Q5/P72Q5
4-Bit CMOS
Microcontroller
Revision 2

S3C72Q5/P72Q5
4-BIT CMOS
MICROCONTROLLER
USER'S MANUAL

Revision 2



ELECTRONICS

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung 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 any consequential or incidental damages.

**S3C72Q5/P72Q5 4-Bit CMOS Microcontroller
User's Manual, Revision 2
Publication Number: 22-S3- C72Q5/P72Q5-032002**

© 2002 Samsung Electronics

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics.

Samsung Electronics' microcontroller business has been awarded full ISO-9001 certification (BSI Certificate No. FM24653). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Ri, Giheung- Eup
Yongin-City, Gyeonggi-Do, Korea
C.P.O. Box #37, Suwon 449-900

TEL: (82)-(031)-209-1934

FAX: (82)-(031)-209-1899

Home Page: <http://www.samsungsemi.com>

Printed in the Republic of Korea

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

Preface

The S3C72Q5/P72Q5 *Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C72Q5/P72Q5 microcontroller for application development. It is organized in two parts:

Part I Programming Model

Part II Hardware Descriptions

Part I contains software-related information to familiarize you with the microcontroller's architecture, programming model, instruction set, and interrupt structure. It has five sections:

Section 1	Product Overview	Section 4	Memory Map
Section 2	Address Spaces	Section 5	SAM48 Instruction Set
Section 3	Addressing Modes		

Section 1, "Product Overview," is a high-level introduction to the S3C72Q5/P72Q5, ranging from a general product description to detailed information about pin characteristics and circuit types.

Section 2, "Address Spaces," introduces you to the S3C72Q5/P72Q5 programming model: the program memory (ROM) and data memory (RAM) structures and how to address them. Section 2 also includes information about stack operations, CPU registers, and the bit sequential carrier (BSC) register.

Section 3, "Addressing Modes," describes types of addressing supported by the SAM48 instruction set (direct, indirect, and bit manipulation) and the addressing modes which are supported (1-bit, 4-bit, and 8-bit). Numerous programming examples make the information practical and usable.

Section 4, "Memory Map," contains a detailed map of the addressable peripheral hardware registers in the memory-mapped area of the RAM (bank 15). Section 4 also contains detailed descriptions in standard format of the most commonly used hardware registers. These easy-to-read register descriptions can be used as a quick-reference source when writing programs.

Section 5, "SAM48 Instruction Set," first introduces the basic features and conventions of the SAM48 instruction set. Then, two summary tables orient you to the individual instructions: One table is a high-level summary of the most important information about each instruction; the other table is designed to give expert programmers a summary of binary code and instruction notation information. The final part of Section 5 contains detailed descriptions of each instruction in a standard format. Each instruction description includes one or more practical examples.

A basic familiarity with the information in Part I will make it easier for you to understand the hardware descriptions in Part II. If you are familiar with the SAM48 product family and are reading this user's manual for the first time, we recommend that you read Sections 1–3 carefully, and just scan the detailed information in Sections 4 and 5 very briefly. Later, you can refer back to Sections 4 and 5 as necessary.

Part II "hardware Descriptions," has detailed information about specific hardware components of the S3C72Q5/P72Q5 microcontroller. Also included in Part II are electrical, mechanical, OTP, and development tools data. Part II has 12 sections:

Section 6	Oscillator Circuit	Section 12	LCD Controller/Driver
Section 7	Interrupts	Section 13	External Memory Interface
Section 8	Power-Down	Section 14	Electrical Data
Section 9	RESET	Section 15	Mechanical Data
Section 10	I/O Ports	Section 16	S3P72Q5 OTP
Section 11	Timers and Timer/Counter 0	Section 17	Development Tools

Two order forms are included at the back of this manual to facilitate customer order for S3C72Q5/P72Q5 microcontrollers: the Mask ROM Order Form, and the Mask Option Selection Form. You can photocopy these forms, fill them out, and then forward them to your local Samsung Sales Representative.

Table of Contents

Part I — Programming Model

Section 1 Product Overview

Overview	1-1
OTP	1-1
Features Summary	1-2
Block Diagram	1-3
Pin Assignments	1-4
Pin Descriptions	1-5
Pin Circuit Diagrams	1-7

Section 2 Address Spaces

Program Memory (ROM)	2-1
Overview	2-1
General-Purpose Memory Areas	2-2
Vector Address Area	2-2
Instruction Reference Area	2-4
Data Memory (RAM)	2-5
Overview	2-5
Working Registers	2-9
Stack Operations	2-13
Stack Pointer (SP)	2-13
Push Operations	2-14
Pop Operations	2-15
Bit Sequential Carrier (BSC)	2-16
Program Counter (PC)	2-17
Program Status Word (PSW)	2-17
Interrupt Status Flags (IS0, IS1)	2-18
Emb Flag (EMB)	2-19
Erb Flag (ERB)	2-20
Skip Condition Flags (SC2, SC1, SC0)	2-21
Carry Flag (C)	2-21

Table of Contents (Continued)

Section 3 Addressing Modes

Overview	3-1
EMB and ERB Initialization Values	3-3
Enable Memory Bank Settings.....	3-4
Select Bank Register (SB)	3-5
Direct And Indirect Addressing.....	3-6
1-Bit Addressing	3-6
4-Bit Addressing	3-8
8-Bit Addressing	3-11

Section 4 Memory Map

Overview	4-1
I/O Map For Hardware Registers	4-1
Register Descriptions.....	4-6

Section 5 SAM48 Instruction Set

Overview	5-1
Instruction Set Features	5-1
Instruction Reference Area	5-2
Reducing Instruction Redundancy.....	5-3
Flexible Bit Manipulation	5-4
Instructions Which Have Skip Conditions.....	5-4
Instructions Which Affect The Carry Flag.....	5-4
ADC and SBC Instruction Skip Conditions	5-5
Symbols And Conventions	5-6
Opcode Definitions	5-7
High-Level Summary	5-8
Binary Code Summary.....	5-13
Instruction Descriptions	5-23

Table of Contents (Continued)

Part II — Hardware Descriptions

Section 6 Oscillator Circuits

Overview	6-1
Main-System Oscillator Circuits	6-3
Sub-System Oscillator Circuits	6-3
Power Control Register (PCON)	6-4
Instruction Cycle Times	6-5
System Clock Mode Register (SCMOD)	6-6
Switching The CPU Clock	6-8
Clock Output Mode Register (CLMOD)	6-10
Clock Output Circuit	6-11
Clock Output Procedure	6-11

Section 7 Interrupts

Overview	7-1
Vectored Interrupts	7-2
Multiple Interrupts	7-5
Interrupt Priority Register (IPR)	7-7
External Interrupt 0 and 1 Mode Registers (IMOD0, IMOD1)	7-8
External Interrupt 2 Mode Register (IMOD2)	7-10
Interrupt Flags	7-12

Section 8 Power-Down

Overview	8-1
Idle Mode Timing Diagrams	8-4
Stop Mode Timing Diagrams	8-5
Recommended Connections for Unused Pins	8-7

Table of Contents (Continued)

Section 9 RESET

Overview	9-1
Hardware Register Values after RESET	9-1

Section 10 I/O Ports

Overview	10-1
Port Mode Flags (PM FLAGS)	10-3
Pull-Up Resistor Mode Register (PUMOD0)	10-4
Port 0,1 Circuit Diagram	10-6
Port 4 Circuit Diagram	10-7
Port 5 Circuit Diagram	10-8
Port 6 Circuit Diagram	10-9
Port 7 Circuit Diagram	10-10

Table of Contents (Continued)

Section 11 Timers and Timer/Counter 0

Overview	11-1
Basic Timer (BT)	11-2
Overview	11-2
Basic Timer Mode Register (BMOD).....	11-5
Basic Timer Counter (BCNT).....	11-6
Basic Timer Operation Sequence	11-6
Watchdog Timer Mode Register (WDMOD).....	11-8
Watchdog Timer Counter (WDCNT).....	11-8
Watchdog Timer Counter Clear Flag (WDTCF)	11-8
8-Bit Timer/Counter 0 (TC0)	11-10
Overview	11-10
TC0 Function Summary	11-10
TC0 Component Summary	11-11
TC0 Enable/Disable Procedure.....	11-12
TC0 Programmable Timer/Counter Function	11-13
TC0 Operation Sequence	11-13
TC0 Event Counter Function	11-14
TC0 Clock Frequency Output	11-15
TC0 External Input Signal Divider	11-16
TC0 Mode Register (TMOD0).....	11-17
TC0 Counter Register (TCNT0)	11-19
TC0 Reference Register (TREF0).....	11-20
TC0 Output Enable Flag (TOE0)	11-20
TC0 Output Latch (TOL0).....	11-20
8-Bit Timer/Counter 1 (TC1)	11-22
Overview	11-22
TC1 Function Summary	11-22
TC1 Component Summary	11-23
TC1 Enable/Disable Procedure.....	11-24
TC1 Programmable Timer/Counter Function	11-25
TC1 Operation Sequence	11-25
TC1 Mode Register (TMOD1).....	11-26
TC1 Counter Register (TCNT1)	11-28
TC1 Reference Register (TREF1).....	11-29
Watch Timer.....	11-30
Overview	11-30
Watch Timer Mode Register (WMOD).....	11-32

Table of Contents (Concluded)

Section 12 LCD Controller/Driver

Overview	12-1
LCD Circuit Diagram.....	12-2
LCD Ram Address Area	12-3
LCD Contrast Control Register (LCNST).....	12-4
LCD Output Control Register 0 (LCON0)	12-5
LCD Output Control Register 1 (LCON1)	12-5
LCD Mode Register (LMOD).....	12-6
Key Scan Register (KSR)	12-17

Section 13 External Memory Interface

Overview	13-1
External Memory Control Register (EMCON).....	13-1
How to Access The External Memory	13-3
External Memory Write Cycle Timing Diagram	13-6
External Memory Read Cycle Timing Diagram	13-6

Section 14 Electrical Data

Overview	14-1
Timing Waveforms	14-10

Section 15 Mechanical Data

Overview	15-1
----------------	------

Section 16 S3P72Q5 OTP

Overview	16-1
Operating Mode Characteristics.....	16-3

Section 17 Development Tools

Overview	17-1
SHINE	17-1
SAMA Assembler	17-1
SASM57	17-1
HEX2ROM.....	17-1
Target Boards.....	17-1
OTPs.....	17-1
TB72Q5 Target Board	17-3
Idle LED	17-5
Stop LED.....	17-5

List of Figures

Figure Number	Title	Page Number
1-1	S3C72Q5/P72Q5 Specified Block Diagram	1-3
1-2	S3C72Q5 Pin Assignment Diagram	1-4
1-3	Pin Circuit Type A	1-7
1-4	Pin Circuit Type A-3	1-7
1-5	Pin Circuit Type B	1-7
1-6	Pin Circuit Type C	1-7
1-7	Pin Circuit Type E-2	1-8
1-8	Pin Circuit Type E-3	1-8
1-9	Pin Circuit Type H-4	1-9
1-10	Pin Circuit Type H-5	1-9
1-11	Pin Circuit Type H-6	1-9
1-12	Pin Circuit Type H-7	1-9
1-13	Pin Circuit Type H-9	1-10
1-14	Pin Circuit Type H-10	1-10
1-15	Pin Circuit Type H-11	1-10
1-16	Pin Circuit Type H-12	1-10
2-1	ROM Address Structure.....	2-2
2-2	Vector Address Map	2-2
2-3	S3C72Q5 Data Memory (RAM) Map	2-6
2-4	Working Register Map	2-9
2-5	Register Pair Configuration.....	2-10
2-6	1-Bit, 4-Bit, and 8-Bit Accumulator.....	2-11
2-7	Push-Type Stack Operations	2-14
2-8	Pop-Type Stack Operations.....	2-15
3-1	RAM Address Structure	3-2
3-2	SMB and SRB Values in the SB Register	3-5
4-1	Register Description Format	4-7
6-1	Clock Circuit Diagram.....	6-2
6-2	Crystal/Ceramic Oscillator	6-3
6-3	External Oscillator	6-3
6-4	RC Oscillator	6-3
6-5	Crystal/Ceramic Oscillator	6-3
6-6	External Oscillator	6-3
6-7	CLO Output Pin Circuit Diagram.....	6-11
7-1	Interrupt Execution Flowchart	7-3
7-2	Interrupt Control Circuit Diagram	7-4
7-3	Two-Level Interrupt Handling.....	7-5
7-4	Multi-Level Interrupt Handling.....	7-6
7-5	Circuit Diagram for INT0 and INT1 Pins	7-9
7-6	Circuit Diagram for INT2.....	7-10

List of Figures (Continued)

Figure Number	Title	Page Number
8-1	Timing When Idle Mode is Released by RESET	8-4
8-2	Timing When Idle Mode is Released by an Interrupt.....	8-4
8-3	Timing When Stop Mode is Released by RESET.....	8-5
8-4	Timing When Stop Mode is Released by an Interrupt	8-5
9-1	Timing for Oscillation Stabilization after RESET	9-1
10-1	Port 0,1 Circuit Diagram	10-6
10-3	Port 4 Circuit Diagram	10-7
10-4	Port 5 Circuit Diagram	10-8
10-5	Port 6 Circuit Diagram	10-9
10-6	Port 7 Circuit Diagram	10-10
11-1	Basic Timer Circuit Diagram	11-4
11-2	TC0 Circuit Diagram	11-12
11-3	TC0 Timing Diagram	11-19
11-4	TC1 Circuit Diagram	11-24
11-5	TC1 Timing Diagram	11-28
11-6	Watch Timer Circuit Diagram	11-31
12-1	LCD Circuit Diagram.....	12-2
12-2	LCD Clock Circuit Diagram.....	12-2
12-3	Display RAM Organization.....	12-3
12-4	LCD Voltage Dividing Resistors Connection	12-8
12-5	RE, LE and Inputs Signal Waveform (1/9 Duty)	12-9
12-6	LCD Signal Waveform for 1/9 Duty and 1/4 Bias	12-10
12-7	RE, LE and Inputs Signal Waveform (1/10 Duty)	12-11
12-8	LCD Signal Waveform for 1/10 Duty and 1/4 Bias	12-12
12-9	RE, LE and Inputs Signal Waveform (1/11 Duty)	12-13
12-10	LCD Signal Waveform for 1/11 Duty and 1/4 Bias	12-14
12-11	RE, LE and Inputs Signal Waveform (1/12 Duty)	12-15
12-12	LCD Signal Waveform for 1/12 Duty and 1/4 Bias	12-16
12-13	Segment Pin Output Signal When LCON1.3 = 1.....	12-17
13-1	External Memory Write Cycle Timing Diagram	13-6
13-2	External Memory Read Cycle Timing Diagram	13-6
13-3	External Interface Function Diagram (S3C72Q5, SRAM, EPROM, EEPROM).....	13-7

List of Figures (Continued)

Figure Number	Title	Page Number
14-1	Standard Operating Voltage Range	14-9
14-2	Stop Mode Release Timing When Initiated By RESET	14-10
14-3	Stop Mode Release Timing When Initiated By Interrupt Request.....	14-10
14-4	A.C. Timing Measurement Points (Except for X _{IN} and XT _{IN})	14-11
14-5	Input Timing for External Interrupts and Quasi-Interrupts	14-11
14-6	Clock Timing Measurement at X _{IN}	14-12
14-7	Clock Timing Measurement at XT _{IN}	14-12
14-6	TCL0 Timing	14-13
14-7	Input Timing for RESET Signal.....	14-13
15-1	100-QFP-1420 Package Dimensions.....	15-1
16-1	S3P72Q5 Pin Assignments (100-QFP Package).....	16-2
16-2	Standard Operating Voltage Range	16-5
17-1	SMDS Product Configuration (SMDS2+)	17-2
17-2	TB72Q5 Target Board Configuration	17-3
17-3	50-Pin Connectors for TB72Q5.....	17-6
17-4	TB72Q5 Adapter Cable for 100-QFP Package (S3C72Q5/P72Q5)	17-6

List of Tables

Table Number	Title	Page Number
1-1	Pin Descriptions	1-5
2-1	Program Memory Address Ranges	2-1
2-2	Data Memory Organization and Addressing.....	2-7
2-3	Working Register Organization and Addressing.....	2-10
2-4	BSC Register Organization.....	2-16
2-5	Program Status Word Bit Descriptions.....	2-17
2-6	Interrupt Status Flag Bit Settings	2-18
2-7	Valid Carry Flag Manipulation Instructions.....	2-21
3-1	RAM Addressing Not Affected by the EMB Value	3-4
3-2	1-Bit Direct and Indirect RAM Addressing.....	3-6
3-3	4-Bit Direct and Indirect RAM Addressing.....	3-8
3-4	8-Bit Direct and Indirect RAM Addressing.....	3-11
4-1	I/O Map for Memory Bank 15	4-2
4-2	I/O Map for Memory Bank 15	4-3
5-1	Valid 1-Byte Instruction Combinations for REF Look-Ups	5-2
5-2	Bit Addressing Modes and Parameters	5-4
5-3	Skip Conditions for ADC and SBC Instructions	5-5
5-4	Data Type Symbols	5-6
5-5	Register Identifiers	5-6
5-6	Instruction Operand Notation	5-6
5-7	Opcode Definitions (Direct)	5-7
5-8	Opcode Definitions (Indirect)	5-7
5-9	CPU Control Instructions — High-Level Summary.....	5-9
5-10	Program Control Instructions — High-Level Summary.....	5-9
5-11	Data Transfer Instructions — High-Level Summary	5-10
5-12	Logic Instructions — High-Level Summary	5-11
5-13	Arithmetic Instructions — High-Level Summary.....	5-11
5-14	Bit Manipulation Instructions — High-Level Summary	5-12
5-15	CPU Control Instructions — Binary Code Summary	5-14
5-16	Program Control Instructions — Binary Code Summary	5-15
5-17	Data Transfer Instructions — Binary Code Summary.....	5-16
5-18	Logic Instructions — Binary Code Summary.....	5-18
5-19	Arithmetic Instructions — Binary Code Summary	5-19
5-20	Bit Manipulation Instructions — Binary Code Summary	5-20
6-1	Power Control Register (PCON) Organization	6-4
6-2	Instruction Cycle Times for CPU Clock Rates.....	6-5
6-3	System Clock Mode Register (SCMOD) Organization	6-6
6-4	Main Oscillation Stop Mode.....	6-7
6-5	Elapsed Machine Cycles During CPU Clock Switch.....	6-8
6-6	Clock Output Mode Register (CLMOD) Organization.....	6-10

List of Tables (Continued)

Table Number	Title	Page Number
7-1	Interrupt Types and Corresponding Port Pin(s)	7-1
7-2	IS1 and IS0 Bit Manipulation for Multi-Level Interrupt Handling	7-6
7-3	Standard Interrupt Priorities	7-7
7-4	Interrupt Priority Register Settings	7-7
7-5	IMOD0 and IMOD1 Register Organization	7-8
7-6	IMOD2 Register Bit Settings	7-10
7-7	Interrupt Enable and Interrupt Request Flag Addresses	7-12
7-8	Interrupt Request Flag Conditions and Priorities	7-13
8-1	Hardware Operation During Power-Down Modes	8-2
8-2	System Operating Mode Comparison	8-3
8-3	Unused Pin Connections for Reducing Power Consumption	8-7
9-1	Hardware Register Values After RESET	9-2
10-1	I/O Port Overview	2
10-2	Port Pin Status During Instruction Execution	10-2
10-3	Port Mode Group Flags	10-3
10-4	Pull-Up Resistor Mode Register (PUMOD0) Organization	10-4
11-1	Basic Timer Register Overview	11-3
11-2	Basic Timer Mode Register (BMOD) Organization	11-5
11-3	Watchdog Timer Interval Time	11-8
11-4	TC0 Register Overview	11-11
11-5	TMOD0 Settings for TCL0 Edge Detection	11-14
11-6	TC0 Mode Register (TMOD0) Organization	11-17
11-7	TMOD0.6, TMOD0.5, and TMOD0.4 Bit Settings	11-18
11-8	TC1 Register Overview	11-23
11-9	TC1 Mode Register (TMOD1) Organization	11-26
11-10	TMOD1.6, TMOD1.5, and TMOD1.4 Bit Settings	11-27
11-11	Watch Timer Mode Register (WMOD) Organization	11-32
12-1	LCD Contrast Control Register (LCNST) Organization	12-4
12-2	LCD Output Control Register (LCON0) Organization	12-5
12-3	LCD Output Control Register (LCON1) Organization	12-5
12-4	LCD Mode Control Register (LMOD) Organization	12-7
12-5	KSR Organization	12-17
13-1	External Memory Control Register (EMCON) Organization	13-2

List of Tables (Continued)

Table Number	Title	Page Number
14-1	Absolute Maximum Ratings	14-2
14-2	D.C Characteristics.....	14-2
14-3	Main System Clock Oscillator Characteristics.....	14-5
14-4	Recommended Oscillator Constants.....	14-6
14-5	Subsystem Clock Oscillator Characteristics.....	14-7
14-6	Input/Output Capacitance	14-7
14-7	A.C. Electrical Characteristics	14-8
14-8	RAM Data Retention Supply Voltage in Stop Mode	14-9
16-1	Descriptions of Pins Used to Read/Write the EPROM	16-3
16-2	Comparison of S3P72Q5 and S3C72Q5 Features	16-3
16-3	Operating Mode Selection Criteria.....	16-3
16-4	D.C Characteristics.....	16-4
17-1	Power Selection Settings for TB72Q5.....	17-4
17-2	Main-clock Selection Settings for TB72Q5.....	17-4
17-3	Sub-clock Selection Settings for TB72Q5.....	17-5

List of Programming Tips

Description	Page Number
Section 2: Address Spaces	
Defining Vectored Interrupts	2-3
Using the REF Look-Up Table	2-4
Clearing Data Memory Bank 0 ,and the page 0 in Bank 1	2-8
Selecting the Working Register Area	2-12
Initializing the Stack Pointer.....	2-13
Using the BSC Register to Output 16-Bit Data	2-16
Setting ISx Flags for Interrupt Processing	2-18
Using the EMB Flag to Select Memory Banks.....	2-19
Using the ERB Flag to Select Register Banks.....	2-20
Using the Carry Flag as a 1-Bit Accumulator.....	2-22
Section 3: Addressing Modes	
Initializing the EMB and ERB Flags	3-3
1-Bit Addressing Modes	3-7
4-Bit Addressing Modes	3-8
8-Bit Addressing Modes	3-12
Section 5: SAM48 Instruction Set	
Example of the Instruction Redundancy Effect.....	5-3
Section 6: Oscillator Circuits	
Setting the CPU Clock	6-4
Switching Between Main-system and Sub-system Clock	6-9
CPU Clock Output to the CLO Pin	6-11
Section 7: Interrupts	
Setting the INT Interrupt Priority	7-8
Using INT2 as a Key Input Interrupt	7-11
Section 8: Power-Down	
Reducing Power Consumption for Key Input Interrupt Processing.....	8-6

List of Programming Tips (Continued)

Description	Page Number
Section 10: I/O Ports	
Configuring I/O Ports to Input or Output	10-3
Enabling and Disabling I/O Port Pull-Up Resistors	10-4
Section 11: Timers and Timer/Counter 0	
Using the Basic Timer	11-7
Using the Watchdog Timer	11-9
TC0 Signal Output to the TCLO0 Pin	11-15
External TCL0 Clock Output to the TCLO0 Pin	11-16
Restarting TC0 Counting Operation	11-18
Setting a TC0 Timer Interval.....	11-21
Restarting TC1 Counting Operation	11-27
Setting a TC1 Timer Interval.....	11-29
Using the Watch Timer	11-33
Section 13: External Memory Interface	
External Memory Interface.....	13-4

List of Register Descriptions

Register Identifier	Full Register Name	Page Number
BMOD	Basic Timer Mode Register	4-8
CLMOD	Clock Output Mode Register	4-9
EMCON	External Memory Control Register	4-10
IE0, 1, IRQ0, 1	INT0, 1 Interrupt Enable/Request Flags	4-11
IE2, IRQ2	INT2 Interrupt Enable/Request Flags	4-12
IEB, IRQB	INTB Interrupt Enable/Request Flags	4-13
IEP0, IRQP0	INTP0 Interrupt Enable/Request Flags	4-14
IET0, IRQT0	INTT0 Interrupt Enable/Request Flags	4-15
IET1, IRQT1	INTT1 Interrupt Enable/Request Flags	4-16
IEW, IRQW	INTW Interrupt Enable/Request Flags	4-17
IMOD0	External Interrupt 0 (INT0) Mode Register	4-18
IMOD1	External Interrupt 1 (INT1) Mode Register	4-19
IMOD2	External Interrupt 2 (INT2) Mode Register	4-20
IPR	Interrupt Priority Register	4-21
LCNST	LCD Contrast Control Register	4-22
LCON0	LCD Output Control Register 0	4-23
LCON1	LCD Output Control Register 1	4-24
LMOD	LCD Mode Register	4-25
PASR	Page Selection Register	4-26
PCON	Power Control Register	4-27
PMG0	Port I/O Mode Register 0 (Group 0: Port 0, 1)	4-28
PMG1	Port I/O Mode Register 1 (Group 1: Port 4, 5)	4-29
PMG2	Port I/O Mode Register 2 (Group 2: Port 6, 7)	4-30
PNE0	N-channel Open-drain Mode Register 0	4-31
PSW	Program Status Word	4-32
PUMOD0	Pull-Up Resistor Mode Register	4-33
SCMOD	System Clock Mode Control Register	4-34
TMOD0	Timer/Counter 0 Mode Register	4-35
TMOD1	Timer/Counter 1 Mode Register	4-36
TOE0	Time/Output Enable Flag Register	4-37
WDFLAG	Watch-Dog Timer's Counter Clear Flag	4-38
WDMOD	Watch-Dog Timer Mode Control Register	4-39
WMOD	Watch Timer Mode Register	4-40

List of Instruction Descriptions

Instruction Mnemonic	Full Instruction Name	Page Number
ADC	Add With Carry	5-24
ADS	Add and Skip on Overflow	5-26
AND	Logical AND	5-28
BAND	Bit Logical AND	5-29
BITR	Bit Reset	5-31
BITS	Bit Set	5-33
BOR	Bit Logical OR	5-35
BTSF	Bit Test and Skip on False	5-37
BTST	Bit Test and Skip on True	5-39
BTSTZ	Bit Test and Skip on True; Clear Bit	5-41
BXOR	Bit Exclusive OR	5-43
CALL	Call Procedure	5-45
CALLS	Call Procedure (Short)	5-46
CCF	Complement Carry Flag	5-47
COM	Complement Accumulator	5-48
CPSE	Compare and Skip if Equal	5-49
DECS	Decrement and Skip on Borrow	5-50
DI	Disable Interrupts	5-51
EI	Enable Interrupts	5-52
IDLE	Idle Operation	5-53
INCS	Increment and Skip on Carry	5-54
IRET	Return from Interrupt	5-55
JP	Jump	5-56
JPS	Jump (Short)	5-57
JR	Jump Relative (Very Short)	5-58
LD	Load	5-60
LDB	Load Bit	5-64
LDC	Load Code Byte	5-66
LDD	Load Data Memory and Decrement	5-68
LDI	Load Data Memory and Increment	5-69
NOP	No Operation	5-70
OR	Logical OR	5-71
POP	Pop from Stack	5-72
PUSH	Push Onto Stack	5-73

List of Instruction Descriptions (Continued)

Instruction Mnemonic	Full Instruction Name	Page Number
RCF	Reset Carry Flag.....	5-74
REF	Reference Instruction.....	5-75
RET	Return from Subroutine.....	5-78
RRC	Rotate Accumulator Right Through Carry.....	5-79
SBC	Subtract With Carry.....	5-80
SBS	Subtract.....	5-82
SCF	Set Carry Flag.....	5-83
SMB	Select Memory Bank.....	5-84
SRB	Select Register Bank.....	5-85
SRET	Return from Subroutine and Skip.....	5-86
STOP	Stop Operation.....	5-87
VENT	Load EMB, ERB, and Vector Address.....	5-88
XCH	Exchange A or EA With Nibble or Byte.....	5-90
XCHD	Exchange and Decrement.....	5-91
XCHI	Exchange and Increment.....	5-92
XOR	Logical Exclusive OR.....	5-93

1

PRODUCT OVERVIEW

OVERVIEW

The S3C72Q5 is a SAM48 core-based 4-bit CMOS single-chip microcontroller. It has two timer/counters and LCD drivers. The S3C72Q5 is especially suited for use in data bank, telephone and LCD general purpose.

It is built around the SAM48 core CPU and contains ROM, RAM, 39 I/O lines, programmable timer/counters, buzzer output, enough LCD dot matrix, external memory interface, and segment drive pins. The S3C72Q5 can be used for dedicated control functions in a variety of applications, and is especially designed for multi data bank, telephone and LCD game.

OTP

The S3C72Q5 microcontroller is also available in OTP (One Time Programmable) version, S3P72Q5. S3P72Q5 microcontroller has an on-chip 16K-byte one-time-programmable EPROM instead of masked ROM. The S3P72Q5 is comparable to S3C72Q5, both in function and in pin configuration.

FEATURES SUMMARY

Memory

- 16 k x 8 bit program memory
- 5,120 x 4 bit data memory
- 144 x 5 bit LCD display memory

39 I/O Pins

- I/O: 23 pins
- Output: maximum 16 pins for 1-bit level output (sharing with segment driver outputs)

8-Bit Basic Timer

- Four internal timer functions
- Watch-dog timer

8-Bit Timer/Counter 0

- Programmable 8-bit timer
- External event counter
- Arbitrary clock frequency output
- External clock signal divider

8-Bit Timer/Counter 1

- Programmable 8-bit timer

Watch Timer

- Time interval generation: 0.5ms, 3.91ms at 32,768Hz
- 4 frequency (2/4/8/16 kHz) outputs to BUZ pin

Interrupts

- Three external vectored interrupts: INT0, INT1, INTPO
- Three internal vectored interrupts: INTB, INTT0, INTT1
- Two quasi-interrupts: INTW, INT2

Memory Mapped I/O Structure

LCD Display

- 60 segments and 12 common terminals
- 9, 10, 11, and 12 common selectable
- Two internal resistor circuit for LCD bias (selectable)
- 16 level LCD contrast control (software)

External Memory Interface

- 512 k x 8 bit external memory access (19 address and 8 data pins-sharing with segment driver outputs)
- Six external memory selection pins (DM0-DM5)
- 1 data read and 1 data write pins (DR, DW-sharing with segment driver outputs)

Power-Down Modes

- Idle mode (only CPU clock stops)
- Stop mode (Main-System clock, Sub-System clock and CPU clock stops)

Oscillation Sources

- Crystal, ceramic, or External RC for system clock
- Main-system clock frequency: 0.4 MHz-6MHz
- Sub-system clock frequency: 32,768 kHz
- CPU clock divider circuit (by 4,8, or 64)

Instruction Execution Times

- 0.67, 1.33, 10.7 μ s at 6 MHz
- 0.95, 1.91, 15.3 μ s at 4.19 MHz
- 122 μ s at 32.768 kHz

Operating Temperature

- -40 °C to 85 °C

Operating Voltage Range

- 1.8 V to 5.5 V

Package Type

- 100-pin QFP Package

BLOCK DIAGRAM

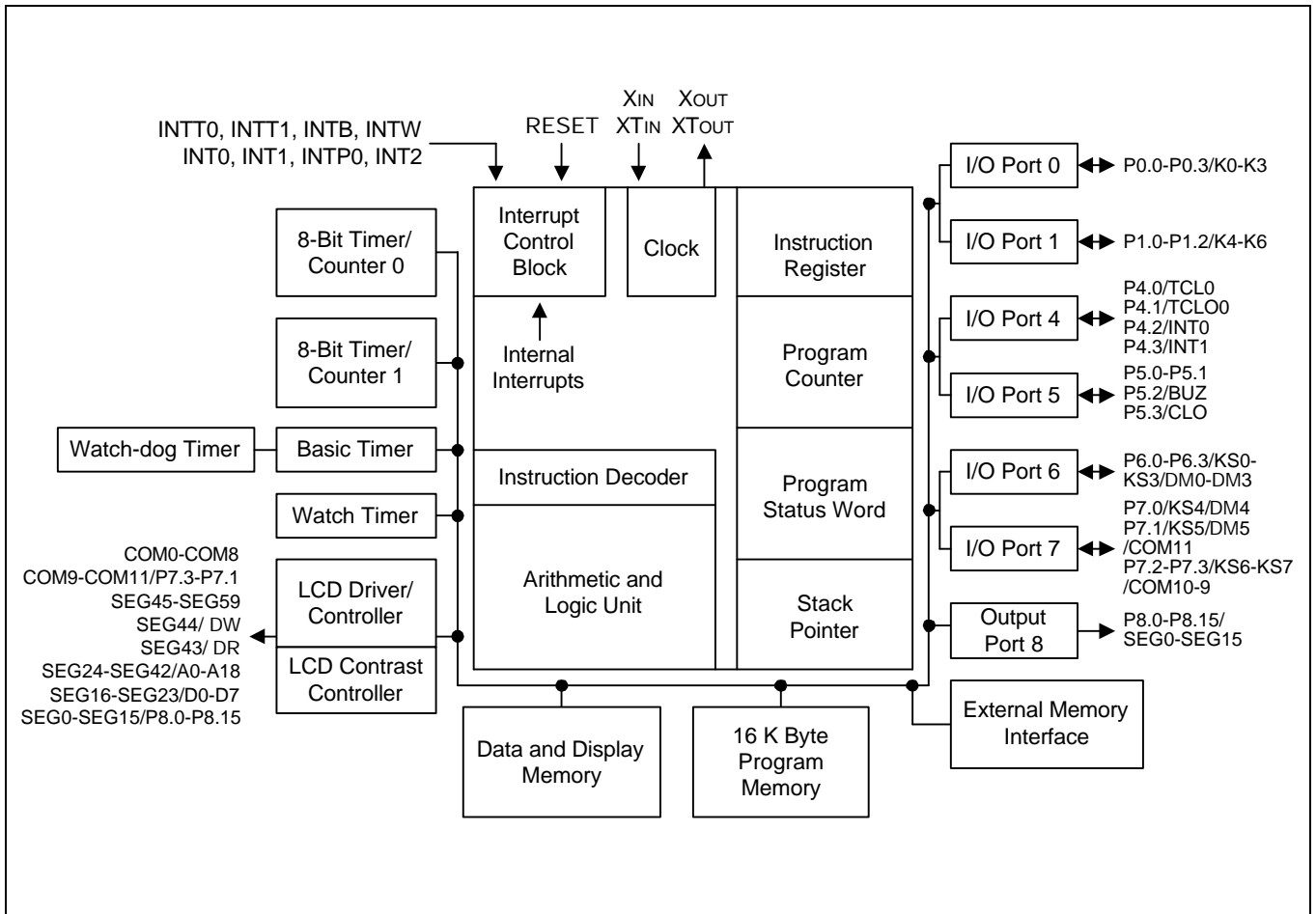


Figure 1-1. S3C72Q5/P72Q5 Specified Block Diagram

PIN ASSIGNMENTS

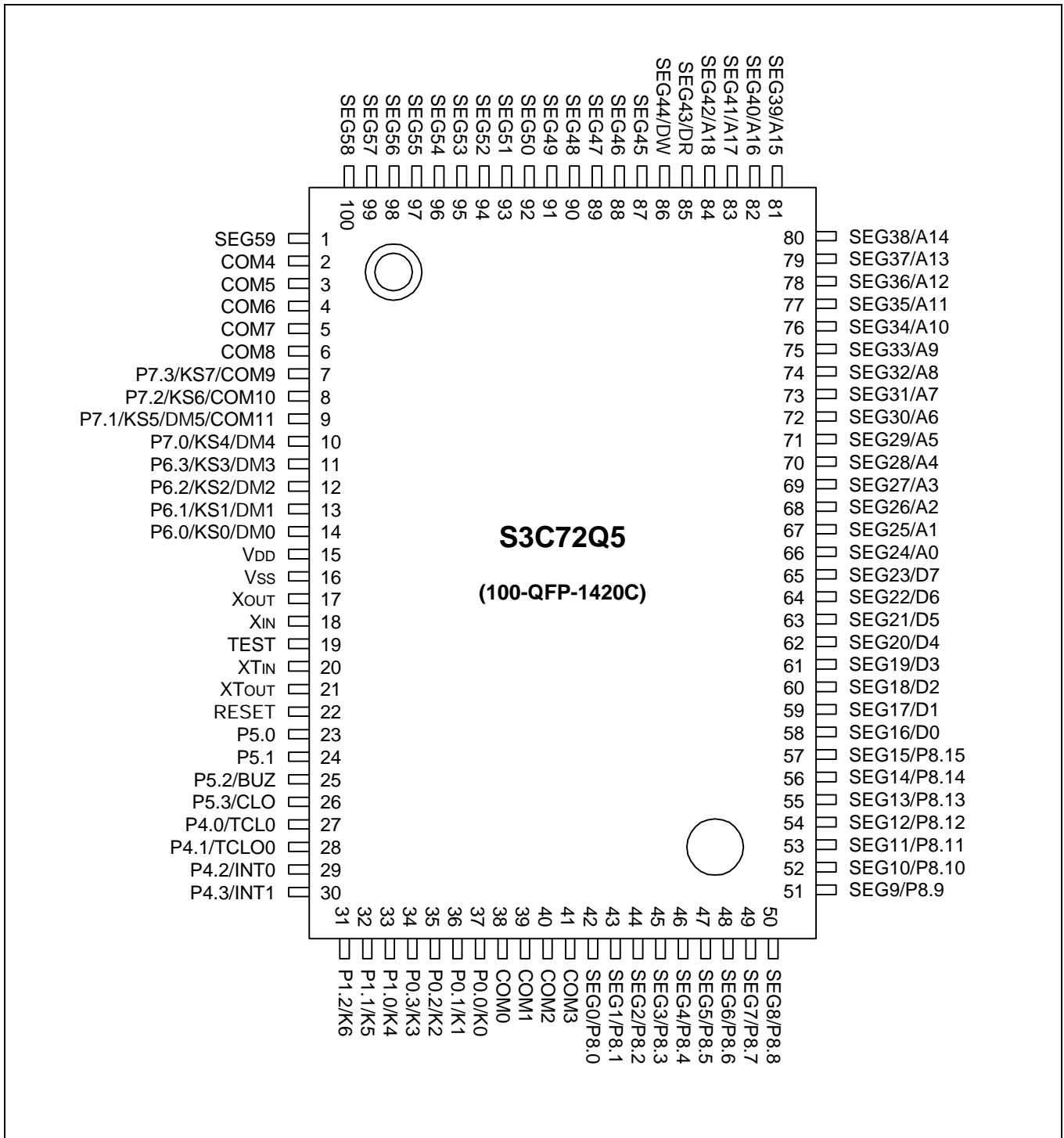


Figure 1-2. S3C72Q5 Pin Assignment Diagram

PIN DESCRIPTIONS

Table 1-1. Pin Descriptions

Pin Name	Pin Type	Description	Circuit Type	Pin Number	Share Pin
P0.0-P0.3 P1.0-P1.2	I/O	4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. Individual pin can be specified as input or output. 7-bit pull-up resistors are assignable by software. <u>Pull-up resistors are automatically disabled for output pins.</u>	E-3	37-34 33-31	K0-K3 K4-K6
P4.0 P4.1 P4.2 P4.3 P5.0-P5.1 P5.2 P5.3		4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. Individual pin can be specified as input or output. 4-bit pull-up resistors are assignable by software. <u>Pull-up resistors are automatically disabled for output pins.</u> Individual pins are software configurable as open-drain or push-pull output.	E-2	27 28 29 30 23-24 25 26	TCL0 TCLO0 INT0 INT1 – BUZ CLO
P6.0-P6.3 P7.0 P7.1 P7.2-P7.3		4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. Individual pin can be specified as input or output. 4-bit pull-up resistors are assignable by software. <u>Pull-up resistors are automatically disabled for output pins.</u>	E-3	14-11 10 9 8-7	KS0-KS3/ DM0-DM3 KS4/DM4 KS5/DM5/COM11 KS6-KS7/COM10- COM9
P8.0-P8.15	O	4-bit controllable output. (Dual function as segment output pins)	H-9	42-57	SEG0 - SEG15
SEG0-SEG15		LCD segment display signal output.	H-9	42-57	P8.0-P8.15
SEG16-SEG23	I/O	LCD segment display signal output.	H-10	58-65	D0-D7
SEG24-SEG42	O	LCD segment display signal output.	H-11	66-84	A0-A18
SEG43,SEG44		LCD segment display signal output.	H-11	85, 86	DR, DW
SEG45-SEG59		LCD segment display signal output.	H-5	87-100,1	–
COM0-COM8		LCD common signal output.	H-4	38-41 2-6	–
COM9-COM10 COM11	I/O	LCD common signal output.	H-12	7-8 9	P7.3-P7.2/ KS7-KS6 P7.1/KS5/DM5
INT0-INT1		External interrupts. The triggering edge for INT0, and INT1 is selectable	E-2	29-30	P4.2 -P4.3

NOTE: P8 can be used to normal output port, when LCD display is off. The value of P8 is determined by KSR0-KSR3 regardless of LMOD.0. (refer to P12-17)

Table 1-1. Pin Descriptions (Continued)

Pin Name	Pin Type	Description	Circuit Type	Pin Num.	Share Pin
BUZ	I/O	2,4,8 kHz or 16 kHz frequency output for buzzer signal.	E-2	25	P5.2
CLO		Clock output		26	P5.3
TCL0		External clock input for Timer/Counter0		27	P4.0
TCLO0		Timer/Counter0 clock output		28	P4.1
K0-K6	I/O	Vector interrupt input. K0-K6: falling edge detection	E-3	37-31	P0.0-P1.2
KS0-KS4		Quasi-interrupt input for falling edge detection	E-3	14-10	P6.0-P7.0/ DM0-DM4
KS5 KS6-KS7			H-12	9 8-7	P7.1/DM5/COM11 P7.2-P7.3/ COM10-COM9
DM0-DM4 DM5		External data memory select signal	E-3 H-12	14-10	P6.0-P7.0/ KS0-KS4 P7.1/KS5/COM11
D0-D7	I/O	Data signal I/O	H-10	58-65	SEG16-SEG23
A0-A18		O	Address signal out	H-11	66-84
DR, DW	I/O	External data memory read/write signal	H-11	85, 86	SEG43,SEG44
X _{IN} , X _{OUT}		–	Crystal, ceramic or RC oscillator pins for main system clock.	–	18, 17
X _{T IN} , X _{T OUT}	–	Crystal oscillator pins for sub-system clock.	–	20, 21	–
RESET	I	Reset input (active low).	B	22	–
V _{DD}	–	Power supply.	–	15	–
V _{SS}	–	Ground.	–	16	–
TEST	I	Test input: it must be connected to V _{SS}	–	19	–

PIN CIRCUIT DIAGRAMS

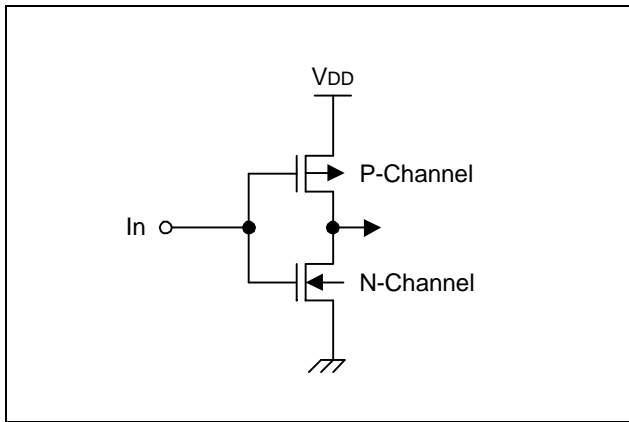


Figure 1-3. Pin Circuit Type A

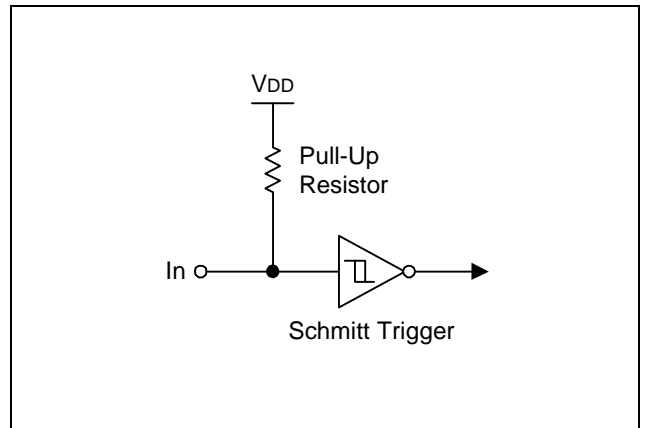


Figure 1-5. Pin Circuit Type B

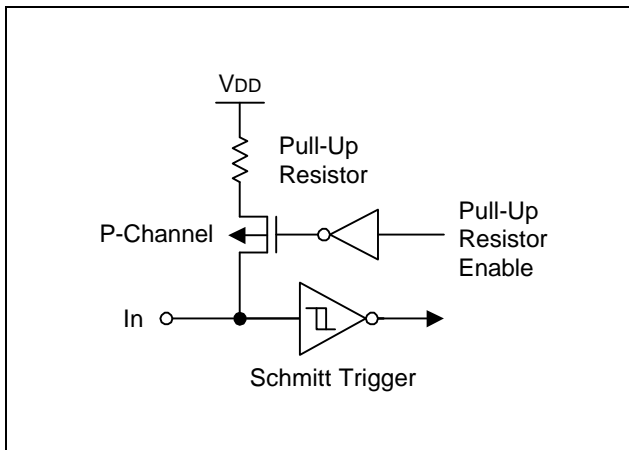


Figure 1-4. Pin Circuit Type A-3

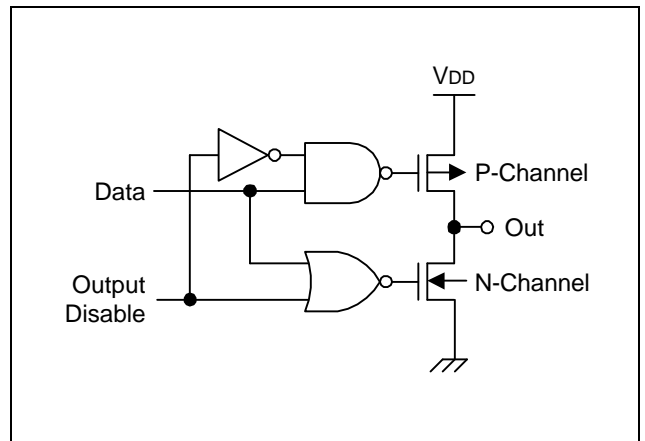


Figure 1-6. Pin Circuit Type C

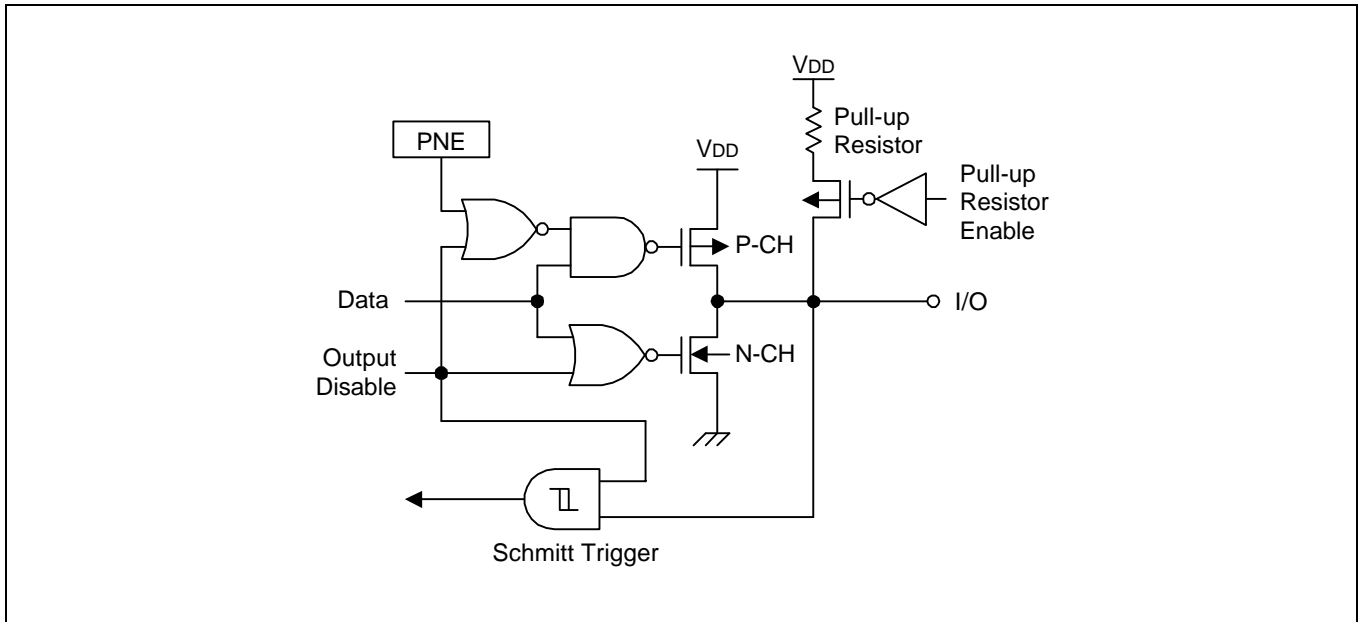


Figure 1-7. Pin Circuit Type E-2

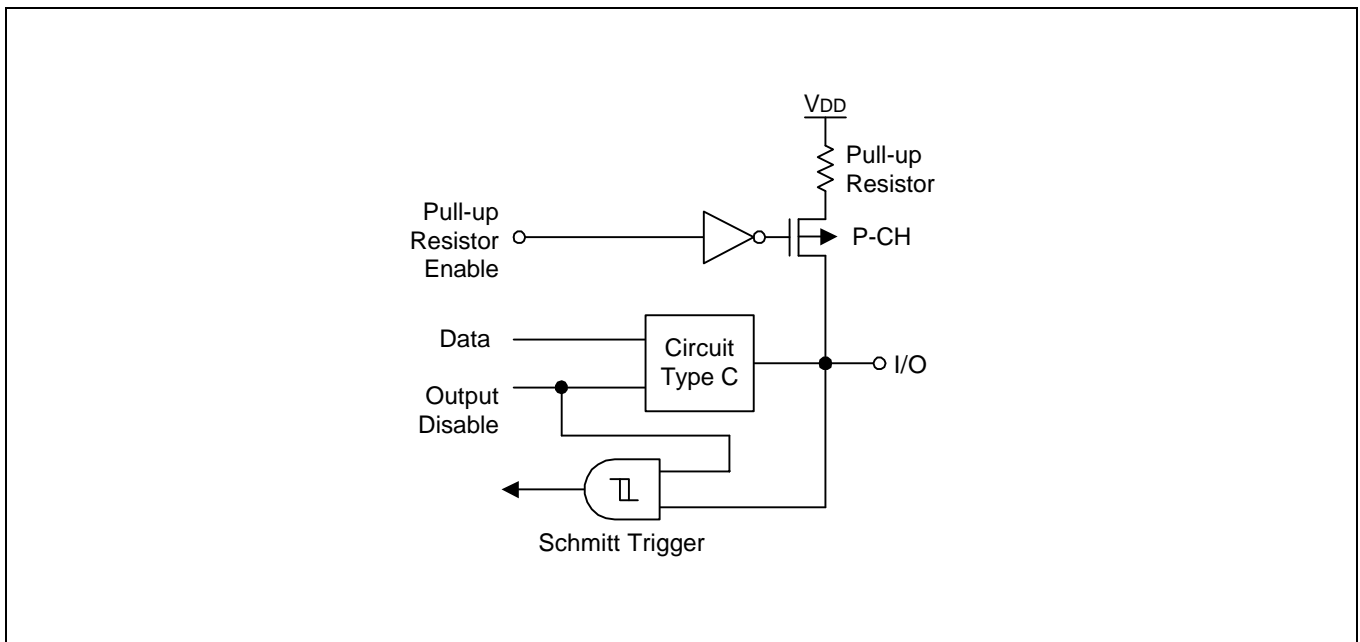


Figure 1-8. Pin Circuit Type E-3

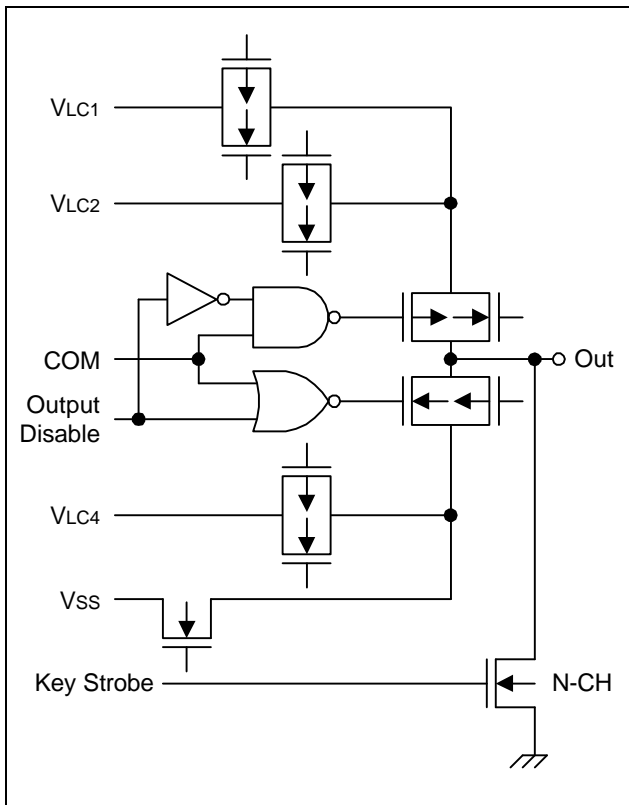


Figure 1-9. Pin Circuit Type H-4

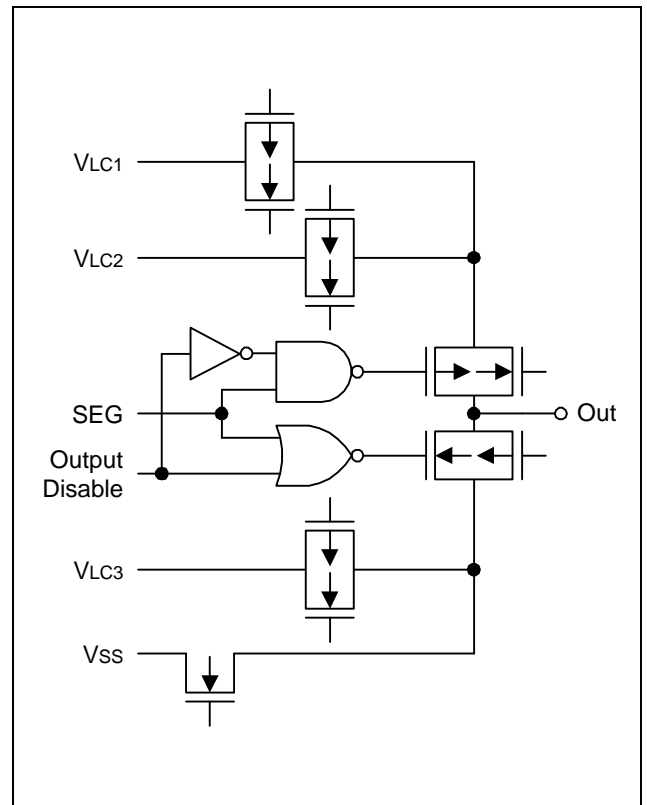


Figure 1-11. Pin Circuit Type H-6

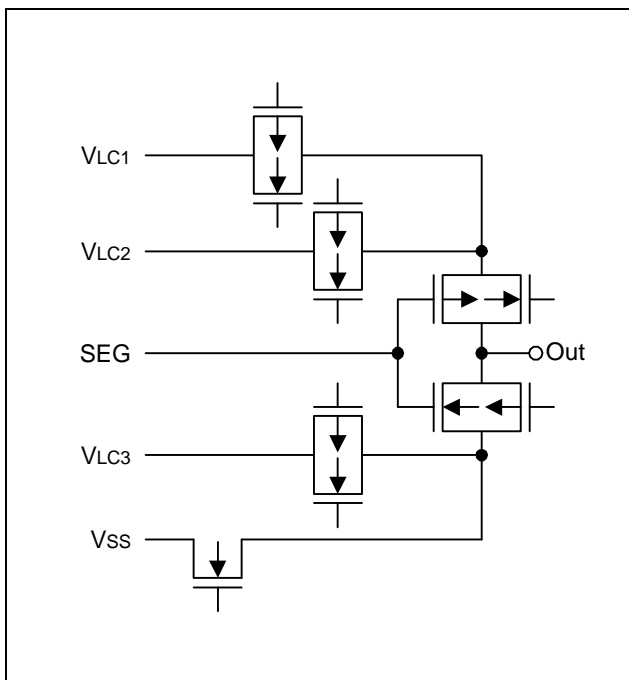


Figure 1-10. Pin Circuit Type H-5

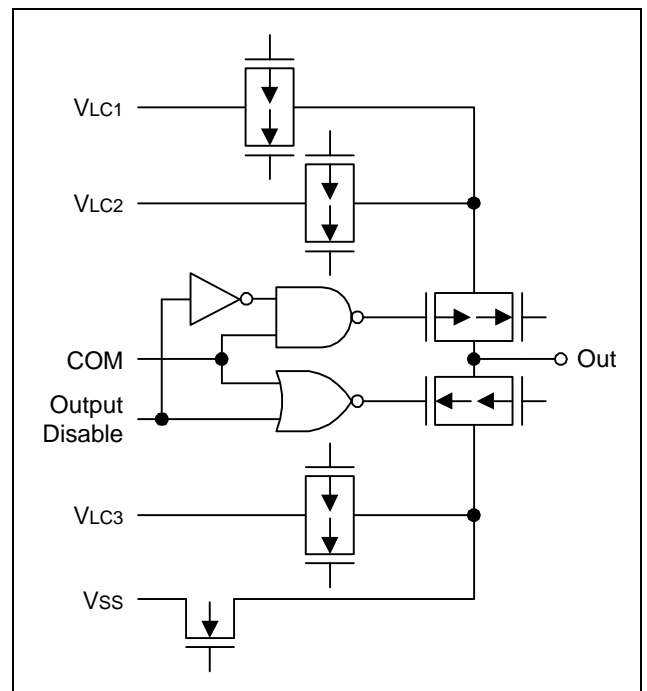


Figure 1-12. Pin Circuit Type H-7

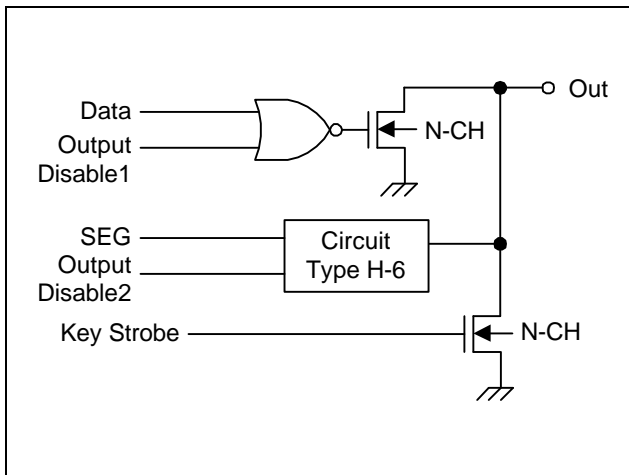


Figure 1-13. Pin Circuit Type H-9

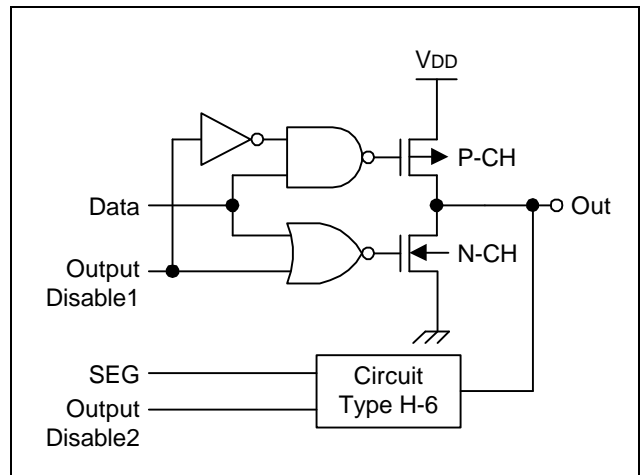


Figure 1-15. Pin Circuit Type H-11

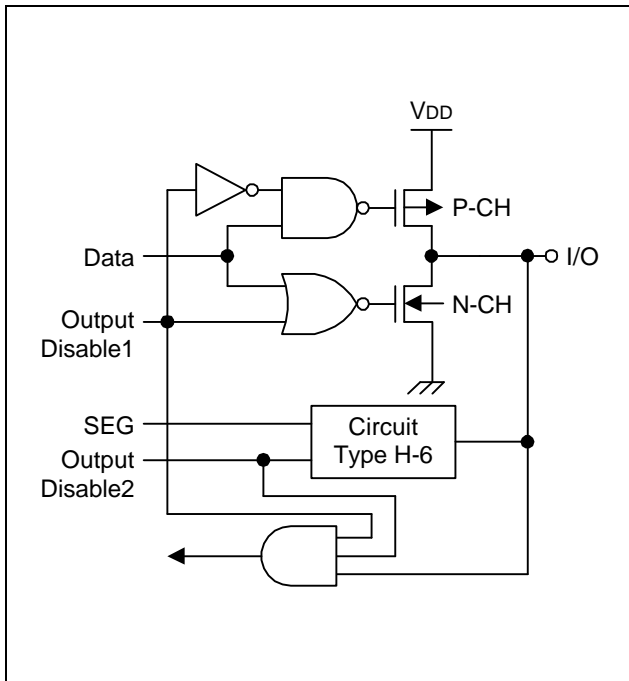


Figure 1-14. Pin Circuit Type H-10

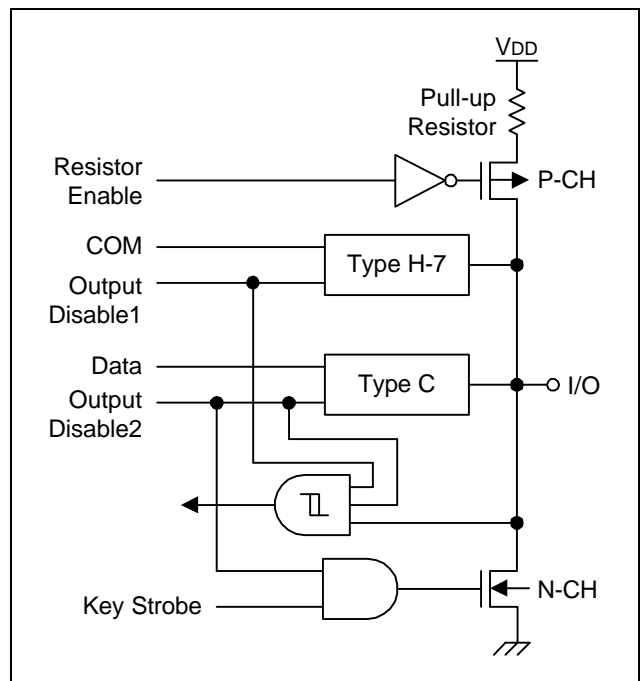


Figure 1-16. Pin Circuit Type H-12

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for S3C72Q5 devices are mask programmable at the factory. In its standard configuration, the device's 16,384 bytes program memory has four areas that are directly addressable by the program counter (PC):

- 14-byte area for vector addresses
- 18-byte general-purpose area
- 96-byte instruction reference area
- 16,256-byte general-purpose area

General-Purpose Program Memory

Two program memory areas are allocated for general-purpose use: One area is 18-byte in size and the other is 16,256-byte.

Vector Addresses

A 14-byte vector address area is used to store the vector addresses required to execute system resets and interrupts. Start addresses for interrupt service routines are stored in this area, along with the values of the enable memory bank (EMB) and enable register bank (ERB) flags that are used to set their initial value for the corresponding service routines. The 14-byte area can be used alternately as general-purpose ROM.

REF Instructions

Locations 0020H-007FH are used as a reference area (look-up table) for 1-byte REF instructions. The REF instruction reduces the byte size of instruction operands. REF can reference one 2-byte instruction, two 1-byte instructions, and one 3-byte instructions which are stored in the look-up table. Unused look-up table addresses can be used as general-purpose ROM.

Table 2-1. Program Memory Address Ranges

ROM Area Function	Address Ranges	Area Size (in Bytes)
Vector address area	0000H-000DH	14
General-purpose program memory	000EH-001FH	18
REF instruction look-up table area	0020H-007FH	96
General-purpose program memory	0080H-3FFFH	16,256

GENERAL-PURPOSE MEMORY AREAS

The 18-byte area at ROM locations 000EH-001FH and the 16,256-byte area at ROM locations 0080H-3FFFH are used as general-purpose program memory. Unused locations in the vector address area and REF instruction look-up table areas can be used as general-purpose program memory. However, care must be taken not to overwrite live data when writing programs that use special-purpose areas of the ROM.

VECTOR ADDRESS AREA

The 14-byte vector address area of the ROM is used to store the vector addresses for executing system resets and interrupts. The starting addresses of interrupt service routines are stored in this area, along with the enable memory bank (EMB) and enable register bank (ERB) flag values that are needed to initialize the service routines. 16-byte vector addresses are organized as follows:

EMB	ERB	0	PC12	PC11	PC10	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

To set up the vector address area for specific programs, use the instruction VENTn. The programming tips on the next page explain how to do this.

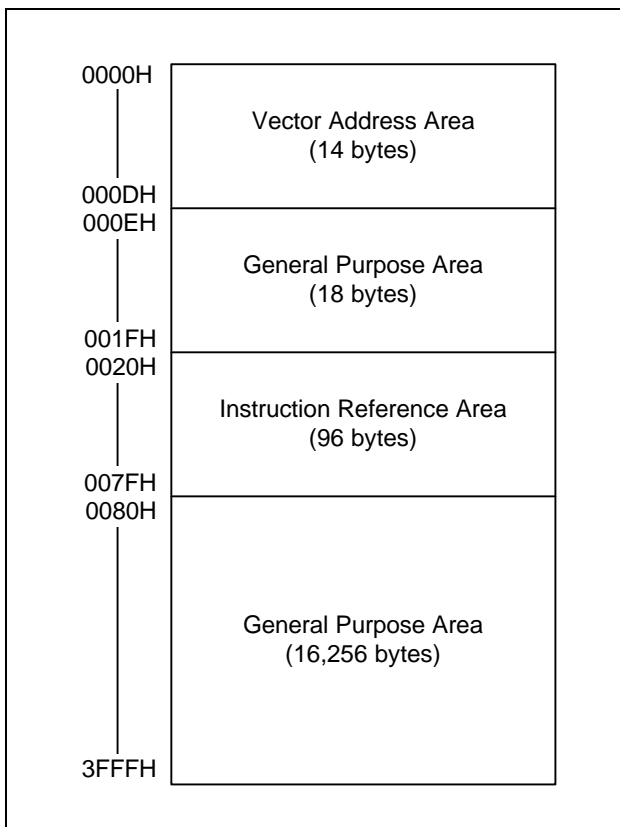


Figure 2-1. ROM Address Structure

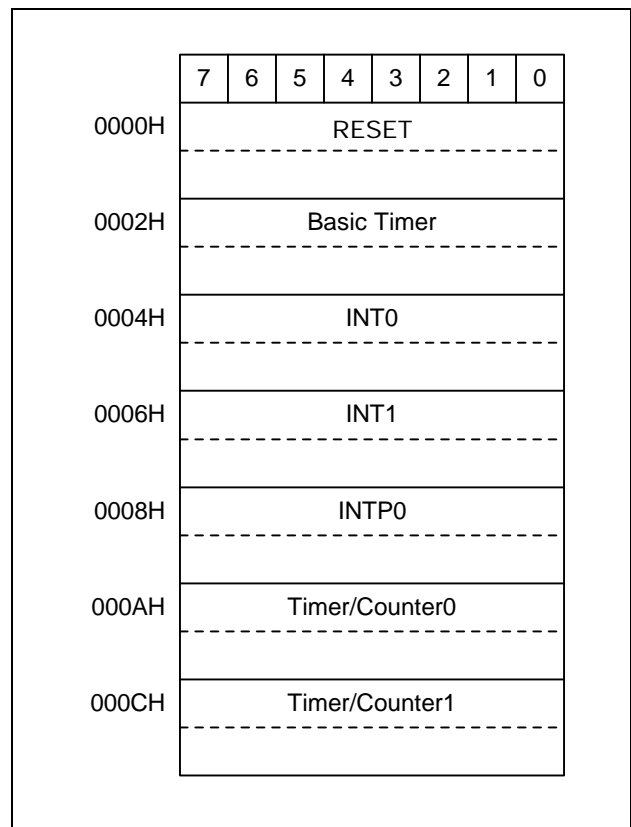


Figure 2-2. Vector Address Map

PROGRAMMING TIP – Defining Vectored Interrupts

The following examples show you several ways you can define the vectored interrupt and instruction reference areas in program memory:

1. When all vector interrupts are used:

```

ORG      0000H
;
VENT0    1,0,RESET      ; EMB ← 1, ERB ← 0; Jump to RESET address by RESET
VENT1    0,0,INTB       ; EMB ← 0, ERB ← 0; Jump to INTB address by INTB
VENT2    0,0,INT0       ; EMB ← 0, ERB ← 0; Jump to INT0 address by INT0
VENT3    0,0,INT1       ; EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
VENT4    0,0,INTP0      ; EMB ← 0, ERB ← 0; Jump to INTP0 address by INTP0
VENT5    0,0,INTT0      ; EMB ← 0, ERB ← 0; Jump to INTT0 address by INTT0
VENT6    0,0,INTT1      ; EMB ← 0, ERB ← 0; Jump to INTT1 address by INTT1

```

2. When a specific vectored interrupt such as INT0, and INTT0 is not used, the unused vector interrupt locations must be skipped with the assembly instruction ORG so that jumps will address the correct locations:

```

ORG      0000H
;
VENT0    1,0,RESET      ; EMB ← 1, ERB ← 0; Jump to RESET address by RESET
VENT1    0,0,INTB       ; EMB ← 0, ERB ← 0; Jump to INTB address by INTB
ORG      0006H          ; INT0 interrupt not used
VENT3    0,0,INT1       ; EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
VENT4    0,0,INTP0      ; EMB ← 0, ERB ← 0; Jump to INTP0 address by INTP0
; INTT0 interrupt not used
ORG      000CH
VENT6    0,0,INTT1      ; EMB ← 0, ERB ← 0; Jump to INTT1 address by INTT1

```

3. If an INT0 and INTT1 interrupt is not used and if its corresponding vector interrupt area is not fully utilized, or if it is not written by a ORG instruction in Example 2, a CPU malfunction will occur:

```

ORG      0000H
;
VENT0    1,0,RESET      ; EMB ← 1, ERB ← 0; Jump to RESET address by RESET
VENT1    0,0,INTB       ; EMB ← 0, ERB ← 0; Jump to INTB address by INTB
VENT3    0,0,INT1       ; EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
VENT4    0,0,INTP0      ; EMB ← 0, ERB ← 0; Jump to INTP0 address by INTP0
VENT5    0,0,INTT0      ; EMB ← 0, ERB ← 0; Jump to INTT0 address by INTT0

```

In this example, when an INTP0 interrupt is generated, the corresponding vector area is not VENT4 INTP0, but VENT5 INTT0. This causes an INTP0 interrupt to jump incorrectly to the INTT0 address and causes a CPU malfunction to occur.

INSTRUCTION REFERENCE AREA

Using 1-byte REF instructions, you can easily reference instructions with larger byte sizes that are stored in addresses 0020H-007FH of program memory. This 96-byte area is called the REF instruction reference area, or look-up table. Locations in the REF look-up table may contain two 1-byte instructions, one 2-byte instruction, or one 3-byte instruction such as a JP (jump) or CALL. The starting address of the instruction you are referencing must always be an even number. To reference a JP or CALL instruction, it must be written to the reference area in a two-byte format: for JP, this format is TJP; for CALL, it is TCALL. In summary, there are three ways to the REF instruction:

By using REF instructions, you can execute instructions larger than one byte. In summary, there are three ways you can use the REF instruction:

- Using the 1-byte REF instruction to execute one 2-byte or two 1-byte instructions
- Branching to any location by referencing a branch instruction stored in the look-up table
- Calling subroutines at any location by referencing a call instruction stored in the look-up table

PROGRAMMING TIP — Using the REF Look-Up Table

Here is one example of how to use the REF instruction look-up table:

```

                ORG      0020H
;
JMAIN          TJP      MAIN          ; 0, MAIN
KEYCK          BTSF    KEYFG         ; 1, KEYFG CHECK
WATCH         TCALL   CLOCK         ; 2, CALL CLOCK
INCHL         LD      @HL,A         ; 3, (HL) ← A
                INCS    HL
                .
                .
ABC           LD      EA,#00H        ; 47, EA ← #00H
                ORG      0080
;
MAIN          NOP
                NOP
                .
                .
                REF     KEYCK        ; BTSF KEYFG (1-byte instruction)
                REF     JMAIN        ; KEYFG = 1, jump to MAIN (1-byte instruction)
                REF     WATCH       ; KEYFG = 0, CALL CLOCK (1-byte instruction)
                REF     INCHL       ; LD @HL,A : INCS HL
                REF     ABC         ; LD EA,#00H (1-byte instruction)
                .
                .

```

DATA MEMORY (RAM)

OVERVIEW

In its standard configuration, the data memories have four areas:

- 32 x 4-bit working register area
- 224 x 4-bit general-purpose area in bank 0 which is also used as the stack area
- 20 pages with 256 x 4-bit in bank1
 - 19 pages for general purpose area (00H-12H page)
 - 1 page for LCD Display data memory (13H page)
- 128 x 4-bit area in bank 15 for memory-mapped I/O addresses

To make it easier to reference, the data memory area has three memory banks — bank 0, bank 1, and bank 15. The select memory bank instruction (SMB) is used to select the bank you want to select as working data memory. Data stored in RAM locations are 1-, 4-, and 8-bit addressable. One exception is the display data memory area, which is 8-bit addressable only.

Initialization values for the data memory area are not defined by hardware therefore must be initialized by program software following power RESET. However, when RESET signal is generated in power-down mode, the most of the data memory contents are held.

Bank 1 Page Selection Register (PASR)

PASR is a 5-bit write -only register for selecting the page of bank1 ,and is mapped to the RAM address FA0H. It should be written by a 8-bit RAM control instruction only and the MSB 3 bits should be "0". PASR retains the previous value as long as change is not required, and the reset value is 0. Therefore, when it returns to the Bank 1 from other bank (Bank 0 or Bank 15) without changing the contents of PASR, the previously specified Bank 1 page is selected . The PASR must not be changed in the interrupt service routine because it's value cannot be recovered as the original value when the routine is finished.

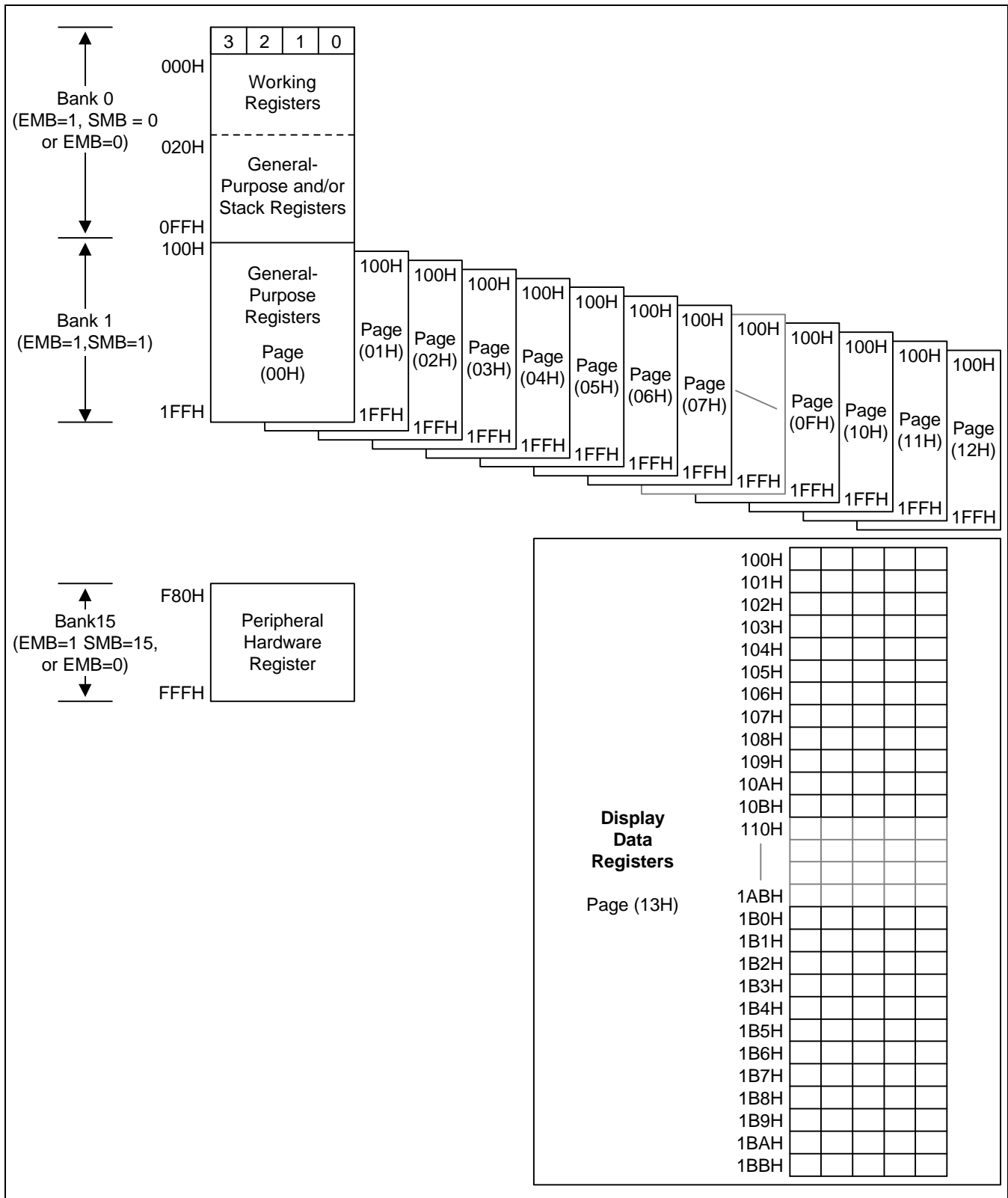


Figure 2-3. S3C72Q5 Data Memory (RAM) Map

Memory Banks 0, 1, and 15

- Bank 0 (000H-0FFH) The lowest 32 nibbles of bank 0 (000H-01FH) are used as working registers; the next 224 nibbles (020H-0FFH) can be used both as stack area and as general-purpose data memory. Use the stack area for implementing subroutine calls and returns, and for interrupt processing.
- Bank 1 (100H-1FFH) Bank 1 has the data memory of 20 pages, the 00H-12H pages for general purpose data memory are comprised of 256 x 4-bits, and the 13H page for LCD display data memory consists of 144 x 5-bits.
The S3C72Q5 use specially a Bank 1 page selection register (PASR) for selecting one of these 20 pages.
- Bank 15 (F80H-FFFH) The microcontroller uses bank 15 for memory-mapped peripheral I/O. Fixed RAM locations for each peripheral hardware address are mapped into this area.

Data Memory Addressing Modes

The enable memory bank (EMB) flag controls the addressing mode for data memory banks 0, 1, or 15. When the EMB flag is logic zero, the addressable area is restricted to specific locations, depending on whether direct or indirect addressing is used. With direct addressing, you can access locations 000H-07FH of bank 0 and bank 15. With indirect addressing, only bank 0 (000H-0FFH) can be accessed. When the EMB flag is set to logic one, all three data memory banks can be accessed according to the current SMB value.

For 8-bit addressing, two 4-bit registers are addressed as a register pair. Also, when using 8-bit instructions to address RAM locations, remember to use the even-numbered register address as the instruction operand.

Working Registers

The RAM working register area in data memory bank 0 is further divided into four *register* banks (bank 0, 1, 2, and, 3). Each register bank has eight 4-bit registers and paired 4-bit registers are 8-bit addressable.

Register A is used as a 4-bit accumulator and register pair EA as an 8-bit extended accumulator. The carry flag bit can also be used as a 1-bit accumulator. Register pairs WX, WL, and HL are used as address pointers for indirect addressing. To limit the possibility of data corruption due to incorrect register addressing, it is advisable to use register bank 0 for the main program and banks 1, 2, and, 3 for interrupt service routines.

LCD Data Register Area

Bit values for LCD segment data are stored in data memory bank 1 (13H page). Register locations in this area that are not used to store LCD data can be assigned to general-purpose use.

Table 2-2. Data Memory Organization and Addressing

Addresses	Register Areas	Bank	EMB Value	SMB Value
000H-01FH	Working registers	0	0, 1	0
020H-0FFH	Stack and general-purpose registers			
100H-1FFH	General-purpose registers (00H-12H pages) LCD display data memory (the 13th page)	1	1	1
F80H-FFFH	I/O-mapped hardware registers	15	0, 1	15

NOTE: LCD data register is 13H page in data memory Bank 1.

 **PROGRAMMING TIP – Clearing Data Memory Bank 0 ,and the page 0 in Bank 1**

Clear bank 0 of the data memory area, and the page 0 of the data memory area in Bank 1

```

        SMB      15
        LD       EA, #00H
        LD       PASR, EA

RAMCLR  SMB      1                ; page 0 in Bank 1 clear
        LD      HL, #00H
        LD      A, #0H
RMCL1   LD      @HL, A
        INCS   HL
        JR     RMCL1
;
        SMB      0                ; Bank 0 clear
RMCL0   LD      HL, #20H
        LD      @HL, A
        INCS   HL
        JR     RMCL0

```

WORKING REGISTERS

Working registers, mapped to RAM address 000H-01FH in data memory bank 0, are used to temporarily store intermediate results during program execution, as well as pointer values used for indirect addressing. Unused registers may be used as general-purpose memory. Working register data can be manipulated as 1-bit unit, 4-bit unit, or using paired registers, as 8-bit unit.

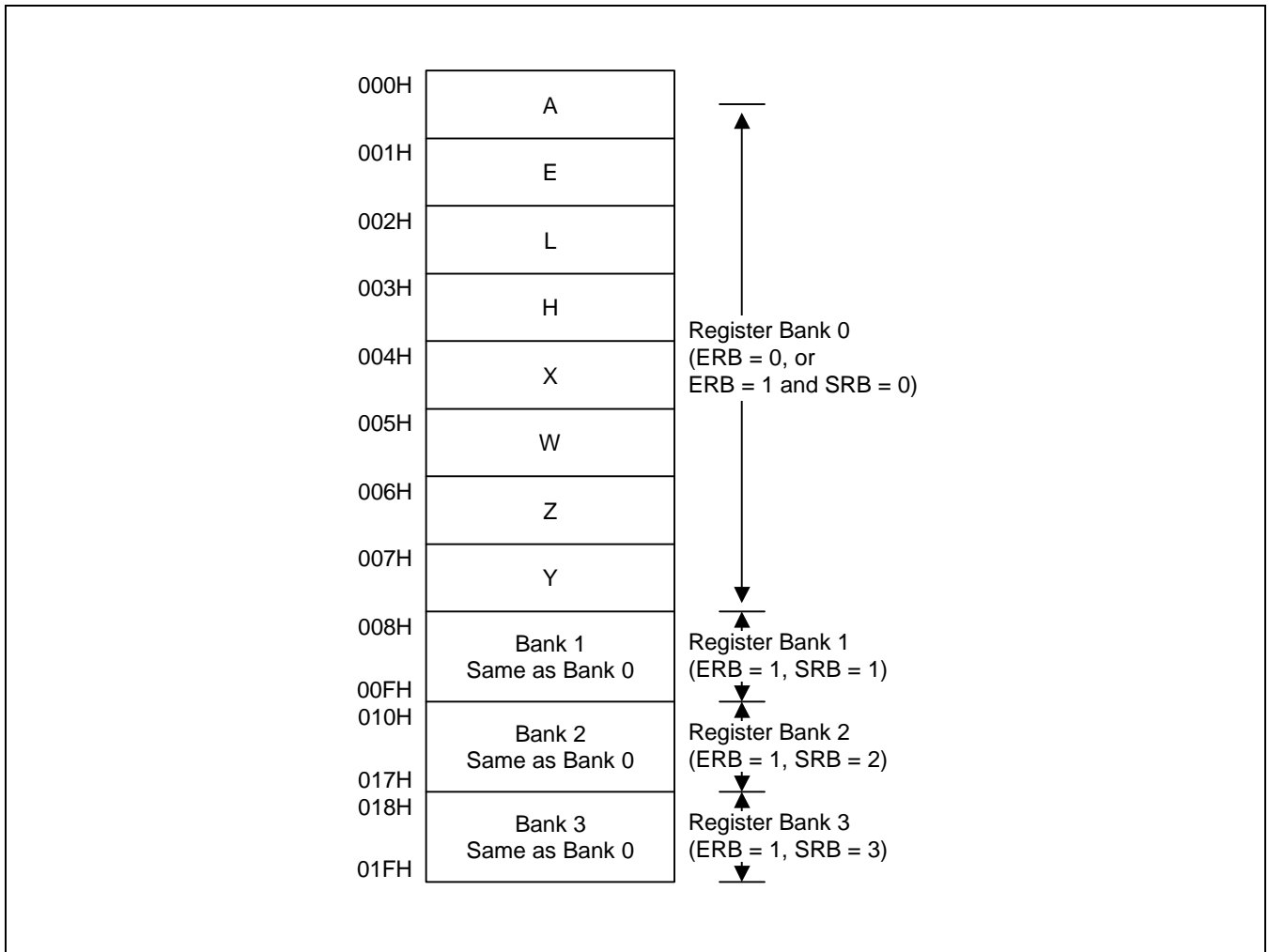


Figure 2-4. Working Register Map

Working Register Banks

For addressing purposes, the working register area is divided into four register banks - bank 0, bank 1, bank 2, and bank 3. Any one of these banks can be selected as the working register bank by the register bank selection instruction (SRB n) and by setting the status of the register bank enable flag (ERB).

Generally, working register bank 0 is used for the main program, and banks 1, 2, and 3 for interrupt service routines. Following this convention helps to prevent possible data corruption during program execution due to contention in register bank addressing.

Table 2-3. Working Register Organization and Addressing

ERB Setting	SRB Settings				Selected Register Bank
	3	2	1	0	
0	0	0	x	x	Always set to bank 0
1	0	0	0	0	Bank 0
			0	1	Bank 1
			1	0	Bank 2
			1	1	Bank 3

NOTE: 'x' means don't care.

Paired Working Registers

Each of the register banks is subdivided into eight 4-bit registers. These registers, named Y, Z, W, X, H, L, E and A, can either be manipulated individually using 4-bit instructions, or together as register pairs for 8-bit data manipulation.

The names of the 8-bit register pairs in each register bank are EA, HL, WX, YZ and WL. Registers A, L, X and Z always become the lower nibble when registers are addressed as 8-bit pairs. This makes a total of eight 4-bit registers or four 8-bit double registers in each of the four working register banks.

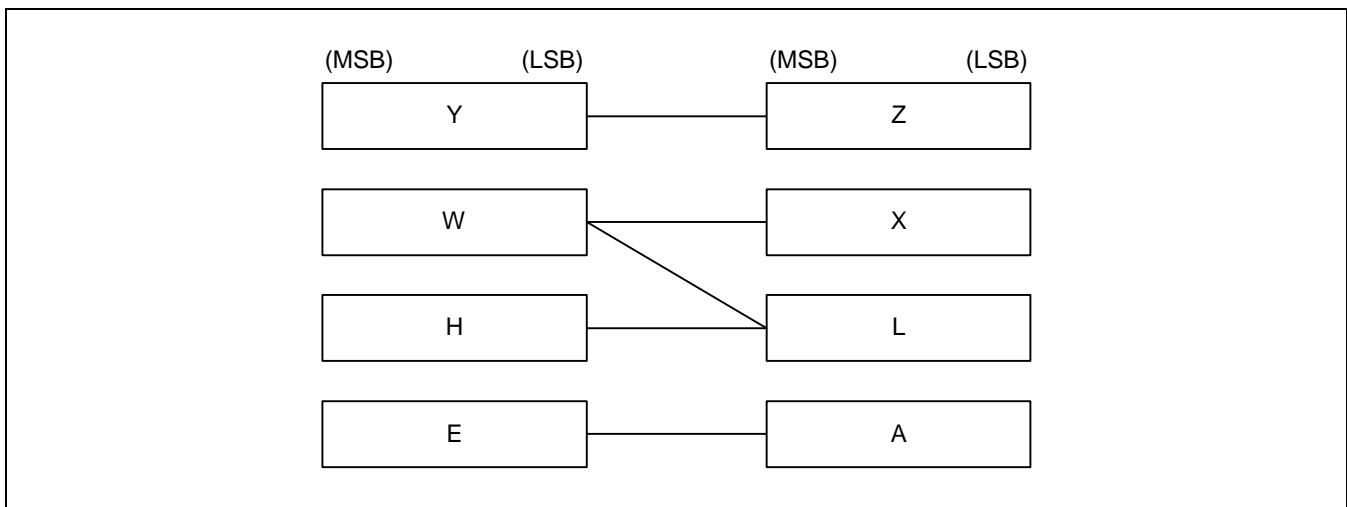


Figure 2-5. Register Pair Configuration

Special-Purpose Working Registers

Register A is used as a 4-bit accumulator and double register EA as an 8-bit accumulator. The carry flag can also be used as a 1-bit accumulator.

8-bit double registers WX, WL and HL are used as data pointers for indirect addressing. When the HL register serves as a data pointer, the instructions LDI, LDD, XCHI, and XCHD can make very efficient use of working registers as program loop counters by letting you transfer a value to the L register and increment or decrement it using a single instruction.

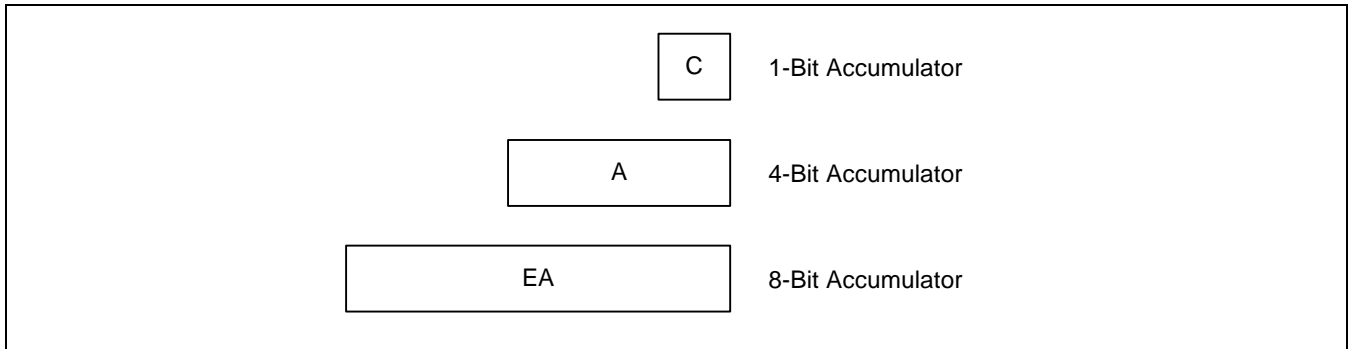


Figure 2-6. 1-Bit, 4-Bit, and 8-Bit Accumulator

Recommendation for Multiple Interrupt Processing

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have been executed successfully, you can restore the register contents from the stack to working memory by using the POP instruction.

PROGRAMMING TIP — Selecting the Working Register Area

The following examples show the correct programming method for selecting working register area:

1. When ERB = "0":

```

                VENT2    1,0,INT0          ; EMB ← 1, ERB ← 0, Jump to INT0 address
;
INT0           PUSH    SB                ; PUSH current SMB, SRB
                SRB      2                ; Instruction does not execute because ERB = "0"
                PUSH    HL                ; PUSH HL register contents to stack
                PUSH    WX                ; PUSH WX register contents to stack
                PUSH    YZ                ; PUSH YZ register contents to stack
                PUSH    EA                ; PUSH EA register contents to stack
                SMB      0
                LD      EA,#00H
                LD      80H,EA
                LD      HL,#40H
                INCS   HL
                LD      WX,EA
                LD      YZ,EA
                POP    EA                ; POP EA register contents from stack
                POP    YZ                ; POP YZ register contents from stack
                POP    WX                ; POP WX register contents from stack
                POP    HL                ; POP HL register contents from stack
                POP    SB                ; POP current SMB, SRB
                IRET

```

The POP instructions execute alternately with the PUSH instructions. If an SMB n instruction is used in an interrupt service routine, a PUSH and POP SB instruction must be used to store and restore the current SMB and SRB values, as shown in Example 2 below.

2. When ERB = "1":

```

                VENT2    1,1,INT0          ; EMB ← 1, ERB ← 1, Jump to INT0 address
;
INT0           PUSH    SB                ; Store current SMB, SRB
                SRB      2                ; Select register bank 2 because of ERB = "1"
                SMB      0
                LD      EA,#00H
                LD      80H,EA
                LD      HL,#40H
                INCS   HL
                LD      WX,EA
                LD      YZ,EA
                POP    SB                ; Restore SMB, SRB
                IRET

```

STACK OPERATIONS

STACK POINTER (SP)

The stack pointer (SP) is an 8-bit register that stores the address used to access the stack, an area of data memory set aside for temporary storage of data and addresses. The SP can be read or written by 8-bit control instructions. When addressing the SP, bit 0 must always remain cleared to logic zero.

F80H	SP3	SP2	SP1	"0"
F81H	SP7	SP6	SP5	SP4

There are two basic stack operations: writing to the top of the stack (push), and reading from the top of the stack (pop). A push decrements the SP and a pop increments it so that the SP always points to the top address of the last data to be written to the stack.

The program counter contents and program status word are stored in the stack area prior to the execution of a CALL or a PUSH instruction, or during interrupt service routines. Stack operation is a LIFO (Last In-First Out) type. The stack area is located in general-purpose data memory bank 0.

During an interrupt or a subroutine, the PC value and the PSW are saved to the stack area. When the routine has been completed, the stack pointer is referenced to restore the PC and PSW, and the next instruction is executed.

The SP can address stack registers in bank 0 (addresses 000H-0FFH) regardless of the current value of the enable memory bank (EMB) flag and the select memory bank (SMB) flag. Although general-purpose register areas can be used for stack operations, be careful to avoid data loss due to simultaneous use of the same register(s).

Since the reset value of the stack pointer is not defined in firmware, we recommend that you initialize the stack pointer by program code to location 00H. This sets the first register of the stack area to 0FFH.

NOTE

A subroutine call occupies six nibbles in the stack; an interrupt requires six. When subroutine nesting or interrupt routines are used continuously, the stack area should be set in accordance with the maximum number of subroutine levels. To do this, estimate the number of nibbles that will be used for the subroutines or interrupts and set the stack area correspondingly.

PROGRAMMING TIP — Initializing the Stack Pointer

To initialize the stack pointer (SP):

- When EMB = "1":

```
SMB      15          ; Select memory bank 15
LD       EA,#00H    ; Bit 0 of accumulator A is always cleared to "0"
LD       SP,EA      ; Stack area initial address (0FFH) ← (SP) - 1
```

- When EMB = "0":

```
LD       EA,#00H
LD       SP,EA      ; Memory addressing area (00H-7FH, F80H-FFFH)
```


PUSH OPERATIONS

Three kinds of push operations reference the stack pointer (SP) to write data from the source register to the stack: PUSH instructions, CALL instructions, and interrupts. In each case, the SP is *decremented* by a number determined by the type of push operation and then points to the next available stack location.

PUSH Instructions

A PUSH instruction references the SP to write two 4-bit data nibbles to the stack. Two 4-bit stack addresses are referenced by the stack pointer: one for the upper register value and another for the lower register. After the PUSH has been executed, the SP is decremented *by two* and points to the next available stack location.

CALL Instructions

When a subroutine call is issued, the CALL instruction references the SP to write the PC's contents to six 4-bit stack locations. Current values for the enable memory bank (EMB) flag and the enable register bank (ERB) flag are also pushed to the stack. Since six 4-bit stack locations are used per CALL, you may nest subroutine calls up to the number of levels permitted in the stack.

Interrupt Routines

An interrupt routine references the SP to push the contents of the PC and the program status word (PSW) to the stack. Six 4-bit stack locations are used to store this data. After the interrupt has been executed, the SP is decremented *by six* and points to the next available stack location. During an interrupt sequence, subroutines may be nested up to the number of levels which are permitted in the stack area.

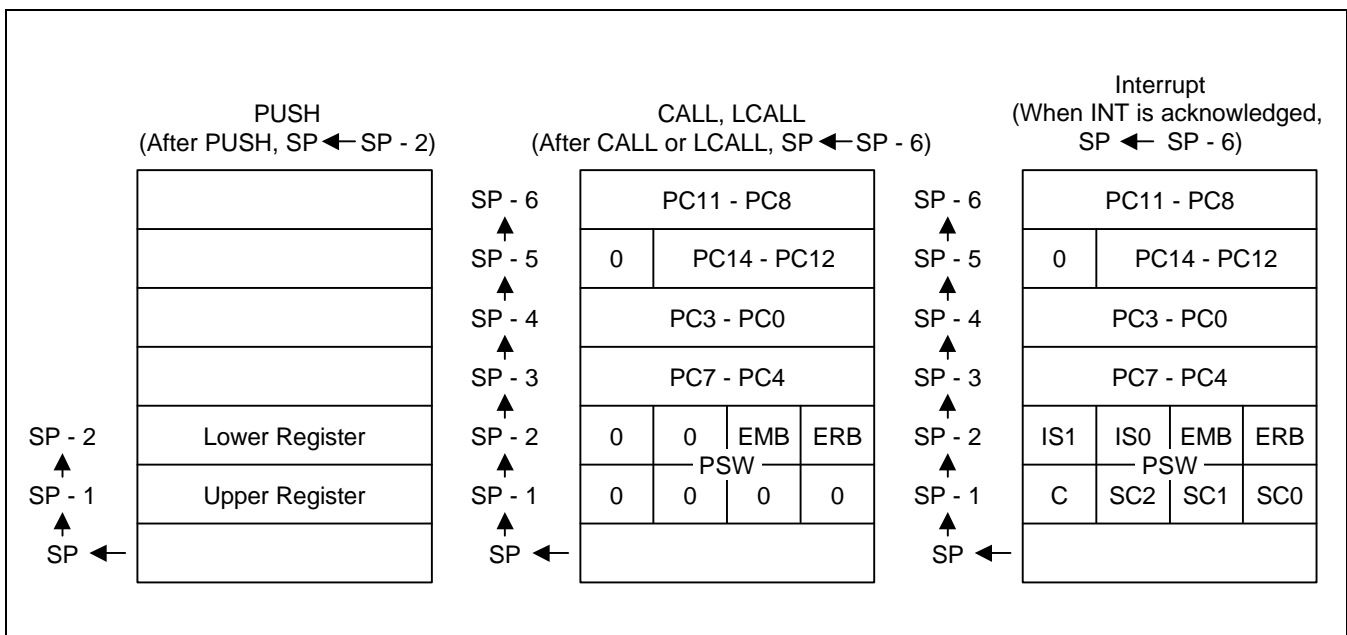


Figure 2-7. Push-Type Stack Operations

POP OPERATIONS

For each push operation, there is a corresponding pop operation to write data from the stack back to the source register or registers: for the PUSH instruction it is the POP instruction; for CALL, the instruction RET or SRET; for interrupts, the instruction IRET. When a pop operation occurs, the SP is *incremented* by a number determined by the type of operation and points to the next free stack location.

POP Instructions

A POP instruction references the SP to write data stored in two 4-bit stack locations back to the register pairs and SB register. The value of the lower 4-bit register is popped first, followed by the value of the upper 4-bit register. After the POP has been executed, the SP is incremented *by two* and points to the next free stack location.

RET and SRET Instructions

The end of a subroutine call is signaled by the return instruction, RET or SRET. The RET or SRET uses the SP to reference the six 4-bit stack locations used for the CALL and to write this data back to the PC, the EMB, and the ERB. After the RET or SRET has been executed, the SP is incremented *by six* and points to the next free stack location.

IRET Instructions

The end of an interrupt sequence is signaled by the instruction IRET. IRET references the SP to locate the six 4-bit stack addresses used for the interrupt and to write this data back to the PC and the PSW. After the IRET has been executed, the SP is incremented *by six* and points to the next free stack location.

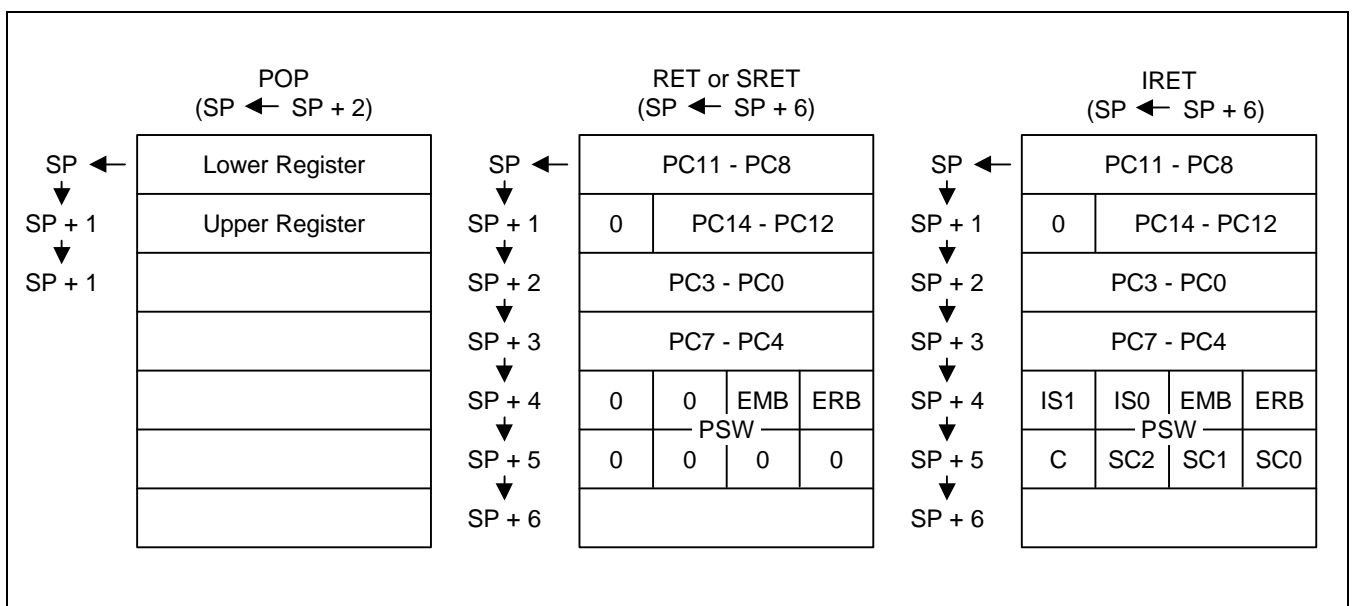


Figure 2-8. Pop-Type Stack Operations

BIT SEQUENTIAL CARRIER (BSC)

The BSC can be manipulated using 1-, 4-, and 8-bit RAM control instructions. RESET clears all BSC bit values to logic zero.

Using the BSC, you can specify sequential addresses and bit locations using 1-bit indirect addressing (memb.@L). (Bit addressing is independent of the current EMB value.) In this way, programs can process 16-bit data by moving the bit location sequentially and then incrementing or decrementing the value of the L register. BSC data can also be manipulated using direct addressing. For 8-bit manipulations, the 4-bit register names BSC0 and BSC2 must be specified and the upper and lower 8 bits manipulated separately.

If the values of the L register are 0H at BSC0.@L, the address and bit location assignment is FC0H.0. If the L register content is FH at BSC0.@L, the address and bit location assignment is FC3H.3.

Table 2-4. BSC Register Organization

Name	Address	Bit 3	Bit 2	Bit 1	Bit 0
BSC0	FC0H	BSC0.3	BSC0.2	BSC0.1	BSC0.0
BSC1	FC1H	BSC1.3	BSC1.2	BSC1.1	BSC1.0
BSC2	FC2H	BSC2.3	BSC2.2	BSC2.1	BSC2.0
BSC3	FC3H	BSC3.3	BSC3.2	BSC3.1	BSC3.0

PROGRAMMING TIP — Using the BSC Register to Output 16-Bit Data

To use the bit sequential carrier (BSC) register to output 16-bit data (5937H) to the P2.0 pin:

```

BITS      EMB
SMB       15
LD        EA,#37H      ;
LD        BSC0,EA     ; BSC0 ← A, BSC1 ← E
LD        EA,#59H     ;
LD        BSC2,EA     ; BSC2 ← A, BSC3 ← E
SMB       0
LD        L,#0H       ;
AGN       LDB         C,BSC0.@L ;
          LDB         P2.0,C    ; P2.0 ← C
INCS     L
JR       AGN
RET

```

PROGRAM COUNTER (PC)

A 13-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a reset operation or an interrupt occurs, bits PC12 through PC0 are set to the vector address.

Usually, the PC is incremented by the number of bytes of the instruction being fetched. One exception is the 1-byte REF instruction which is used to reference instructions stored in the ROM.

PROGRAM STATUS WORD (PSW)

The program status word (PSW) is an 8-bit word that defines system status and program execution status and which permits an interrupted process to resume operation after an interrupt request has been serviced. PSW values are mapped as follows:

FB0H	IS1	IS0	EMB	ERB
FB1H	C	SC2	SC1	SC0

The PSW can be manipulated by 1-bit or 4-bit read/write and by 8-bit read instructions, depending on the specific bit or bits being addressed. The PSW can be addressed during program execution regardless of the current value of the enable memory bank (EMB) flag.

Part or all of the PSW is saved to stack prior to execution of a subroutine call or hardware interrupt. After the interrupt has been processed, the PSW values are popped from the stack back to the PSW address.

When a RESET is generated, the EMB and ERB values are set according to the RESET vector address, and the carry flag is left undefined (or the current value is retained). PSW bits IS0, IS1, SC0, SC1, and SC2 are all cleared to logic zero.

Table 2-5. Program Status Word Bit Descriptions

PSW Bit Identifier	Description	Bit Addressing	Read/Write
IS1, IS0	Interrupt status flags	1, 4	R/W
EMB	Enable memory bank flag	1	R/W
ERB	Enable register bank flag	1	R/W
C	Carry flag	1	R/W
SC2, SC1, SC0	Program skip flags	8	R

INTERRUPT STATUS FLAGS (IS0, IS1)

PSW bits IS0 and IS1 contain the current interrupt execution status values. You can manipulate IS0 and IS1 flags directly using 1-bit RAM control instructions.

By manipulating interrupt status flags in conjunction with the interrupt priority register (IPR), you can process multiple interrupts by anticipating the next interrupt in an execution sequence. The interrupt priority control circuit determines the IS0 and IS1 settings in order to control multiple interrupt processing. When both interrupt status flags are set to "0", all interrupts are allowed. The priority with which interrupts are processed is then determined by the IPR.

When an interrupt occurs, IS0 and IS1 are pushed to the stack as part of the PSW and are automatically incremented to the next higher priority level. Then, when the interrupt service routine ends with an IRET instruction, IS0 and IS1 values are restored to the PSW. Table 2-6 shows the effects of IS0 and IS1 flag settings.

Since interrupt status flags can be addressed by write instructions, programs can exert direct control over interrupt processing status. Before interrupt status flags can be addressed, however, you must first execute a DI instruction to inhibit additional interrupt routines. When the bit manipulation has been completed, execute an EI instruction to re-enable interrupt processing.

Table 2-6. Interrupt Status Flag Bit Settings

IS1 Value	IS0 Value	Status of Currently Executing Process	Effect of IS0 and IS1 Settings on Interrupt Request Control
0	0	0	All interrupt requests are serviced
0	1	1	Only high-priority interrupt(s) as determined in the interrupt priority register (IPR) are serviced
1	0	2	No more interrupt requests are serviced
1	1	–	Not applicable; these bit settings are undefined

PROGRAMMING TIP — Setting ISx Flags for Interrupt Processing

The following instruction sequence shows how to use the IS0 and IS1 flags to control interrupt processing:

```
INTB      DI                ; Disable interrupt
          BITR      IS1     ; IS1 ← 0
          BITS     IS0     ; Allow interrupts according to IPR priority level
          EI                ; Enable interrupt
          •
          •
          •
          IRET
```

EMB FLAG (EMB)

The EMB flag is used to allocate specific address locations in the RAM by modifying the upper 4 bits of 12-bit data memory addresses. In this way, it controls the addressing mode for data memory banks.

When the EMB flag is "0", the data memory address space is restricted to addresses 0F80H-0FFFH of data memory bank 15 and 000H-07FH of bank 0, regardless of the SMB register contents. When the EMB flag is set to "1", the addressing area of data memory is expanded and all of data memory space can be accessed by using the appropriate SMB value.

 PROGRAMMING TIP — Using the EMB Flag to Select Memory Banks

EMB flag settings for memory bank selection:

1. When EMB = "0":

SMB	1	; Non-essential instruction since EMB = "0"
LD	A,#9H	
LD	90H,A	; (F90H) ← A, bank 15 is selected
LD	34H,A	; (034H) ← A, bank 0 is selected
SMB	0	; Non-essential instruction since EMB = "0"
LD	90H,A	; (F90H) ← A, bank 15 is selected
LD	34H,A	; (034H) ← A, bank 0 is selected
SMB	15	; Non-essential instruction, since EMB = "0"
LD	20H,A	; (020H) ← A, bank 0 is selected
LD	90H,A	; (F90H) ← A, bank 15 is selected

2. When EMB = "1":

SMB	1	; Select memory bank 1
LD	A,#9H	
LD	90H,A	; (190H) ← A, bank 1 is selected
LD	34H,A	; (134H) ← A, bank 1 is selected
SMB	0	; Select memory bank 0
LD	90H,A	; (090H) ← A, bank 0 is selected
LD	34H,A	; (034H) ← A, bank 0 is selected
SMB	15	; Select memory bank 15
LD	20H,A	; Program error, but assembler does not detect it
LD	90H,A	; (F90H) ← A, bank 15 is selected

ERB FLAG (ERB)

The 1-bit register bank enable flag (ERB) determines the range of addressable working register area. When the ERB flag is "1", the working register area from register banks 0 to 3 is selected according to the register bank selection register (SRB). When the ERB flag is "0", register bank 0 is the selected working register area, regardless of the current value of the register bank selection register (SRB).

When an internal RESET is generated, bit 6 of program memory address 0000H is written to the ERB flag. This automatically initializes the flag. When a vectored interrupt is generated, bit 6 of the respective address table in program memory is written to the ERB flag, setting the correct flag status before the interrupt service routine is executed.

During the interrupt routine, the ERB value is automatically pushed to the stack area along with the other PSW bits. Afterwards, it is popped back to the FB0H.0 bit location. The initial ERB flag settings for each vectored interrupt are defined using VENTn instructions.

 PROGRAMMING TIP — Using the ERB Flag to Select Register Banks

ERB flag settings for register bank selection:

1. When ERB = "0":

SRB	1	; Register bank 0 is selected (since ERB = "0", the SRB is configured to bank 0)
LD	EA,#34H	; Bank 0 EA ← #34H
LD	HL,EA	; Bank 0 HL ← EA
SRB	2	; Register bank 0 is selected
LD	YZ,EA	; Bank 0 YZ ← EA
SRB	3	; Register bank 0 is selected
LD	WX,EA	; Bank 0 WX ← EA

2. When ERB = "1":

SRB	1	; Register bank 1 is selected
LD	EA,#34H	; Bank 1 EA ← #34H
LD	HL,EA	; Bank 1 HL ← Bank 1 EA
SRB	2	; Register bank 2 is selected
LD	YZ,EA	; Bank 2 YZ ← BANK2 EA
SRB	3	; Register bank 3 is selected
LD	WX,EA	; Bank 3 WX ← Bank 3 EA

SKIP CONDITION FLAGS (SC2, SC1, SC0)

The skip condition flags SC2, SC1, and SC0 indicate the current program skip conditions and are set and reset automatically during program execution. Skip condition flags can only be addressed by 8-bit read instructions. Direct manipulation of the SC2, SC1, and SC0 bits is not allowed.

CARRY FLAG (C)

The carry flag is used to save the result of an overflow or borrow when executing arithmetic instructions involving a carry (ADC, SBC). The carry flag can also be used as a 1-bit accumulator for performing Boolean operations involving bit-addressed data memory.

If an overflow or borrow condition occurs when executing arithmetic instructions with carry (ADC, SBC), the carry flag is set to "1". Otherwise, its value is "0". When a RESET occurs, the current value of the carry flag is retained during power-down mode, but when normal operating mode resumes, its value is undefined.

The carry flag can be directly manipulated by predefined set of 1-bit read/write instructions, independent of other bits in the PSW. Only the ADC and SBC instructions, and the instructions listed in Table 2-7, affect the carry flag.

Table 2-7. Valid Carry Flag Manipulation Instructions

Operation Type	Instructions	Carry Flag Manipulation
Direct manipulation	SCF	Set carry flag to "1"
	RCF	Clear carry flag to "0" (reset carry flag)
	CCF	Invert carry flag value (complement carry flag)
	BTST C	Test carry and skip if C = "1"
Bit transfer	LDB (operand) ⁽¹⁾ ,C	Load carry flag value to the specified bit
	LDB C,(operand) ⁽¹⁾	Load contents of the specified bit to carry flag
Boolean manipulation	BAND C,(operand) ⁽¹⁾	AND the specified bit with contents of carry flag and save the result to the carry flag
	BOR C,(operand) ⁽¹⁾	OR the specified bit with contents of carry flag and save the result to the carry flag
	BXOR C,(operand) ⁽¹⁾	XOR the specified bit with contents of carry flag and save the result to the carry flag
Interrupt routine	INTn ⁽²⁾	Save carry flag to stack with other PSW bits
Return from interrupt	IRET	Restore carry flag from stack with other PSW bits

NOTES:

1. The operand has three bit addressing formats: mema.a, memb.@L, and @H + DA.b.
2. 'INTn' refers to the specific interrupt being executed and is not an instruction.

PROGRAMMING TIP — Using the Carry Flag as a 1-Bit Accumulator

1. Set the carry flag to logic one:

```
SCF                ; C ← 1
LD      EA,#0C3H   ; EA ← #0C3H
LD      HL,#0AAH   ; HL ← #0AAH
ADC     EA,HL      ; EA ← #0C3H + #0AAH + #1H, C ← 1
```

2. Logical-AND bit 3 of address 3FH with P2.0 and output the result to P5.0:

```
LD      H,#3H      ; Set the upper four bits of the address to the H register
                          ; value
LDB     C,@H+0FH.3 ; C ← bit 3 of 3FH
BAND    C,P2.0     ; C ← C AND P2.0
LDB     P5.0,C     ; Output result from carry flag to P5.0
```

3 ADDRESSING MODES

OVERVIEW

The enable memory bank flag, EMB, controls the two addressing modes for data memory. When the EMB flag is set to logic one, you can address the entire RAM area; when the EMB flag is cleared to logic zero, the addressable area in the RAM is restricted to specific locations.

The EMB flag works in connection with the select memory bank instruction, SMBn. You will recall that the SMBn instruction is used to select RAM bank 0, 1, or 15. The SMB setting is always contained in the upper four bits of a 12-bit RAM address. For this reason, both addressing modes (EMB = "0" and EMB = "1") apply specifically to the memory bank indicated by the SMB instruction, and any restrictions to the addressable area within banks 0, 1, or 15. Direct and indirect 1-bit, 4-bit, and 8-bit addressing methods can be used. Several RAM locations are addressable at all times, regardless of the current EMB flag setting.

Here are a few guidelines to keep in mind regarding data memory addressing:

- When you address peripheral hardware locations in bank 15, the mnemonic for the memory-mapped hardware component can be used as the operand in place of the actual address location.
- Display RAM locations in bank 1 are 8-bit addressable only.
- Always use an even-numbered RAM address as the operand in 8-bit direct and indirect addressing.
- With direct addressing, use the RAM address as the instruction operand; with indirect addressing, the instruction specifies a register which contains the operand's address.

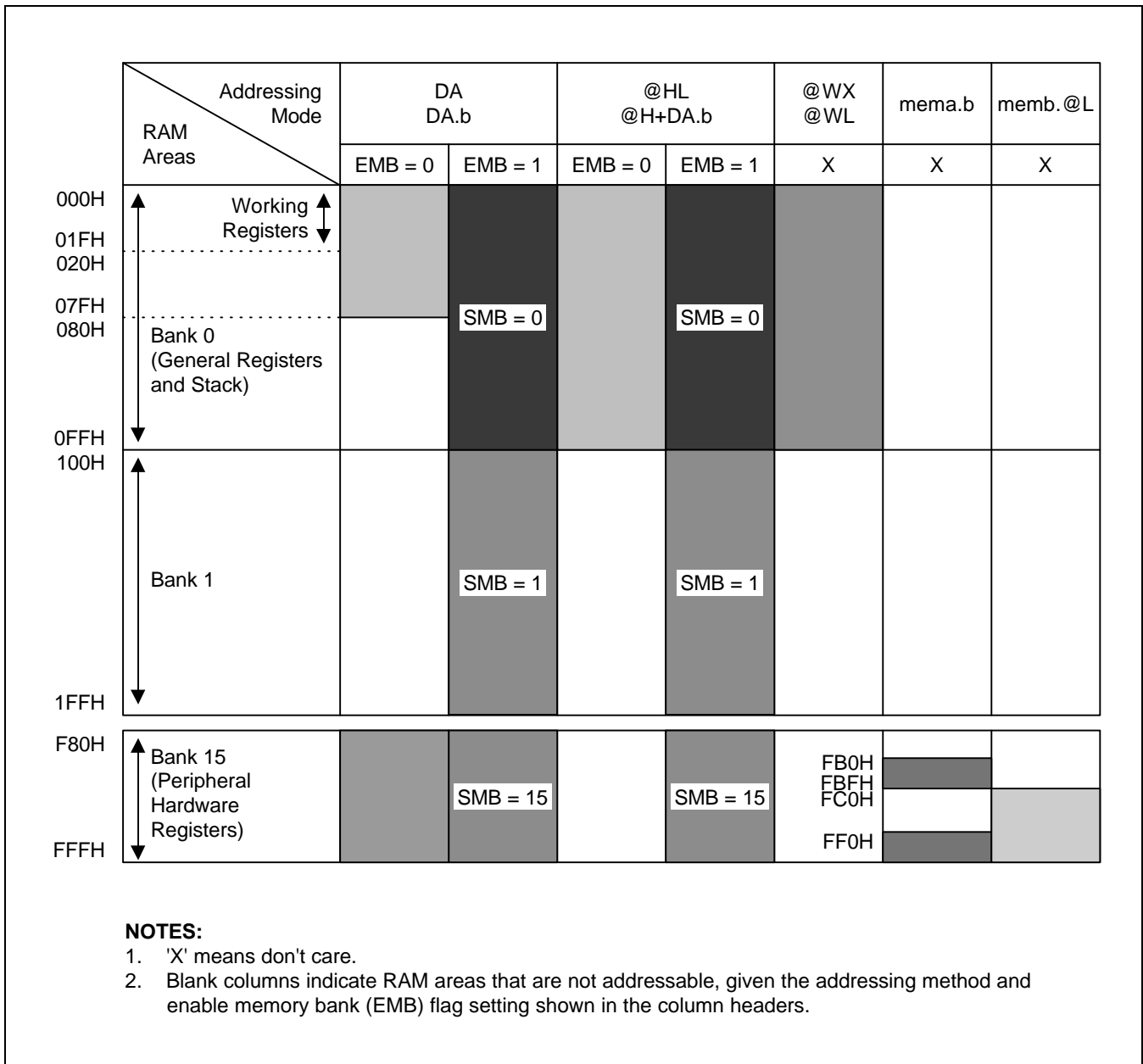


Figure 3-1. RAM Address Structure

EMB AND ERB INITIALIZATION VALUES

The EMB and ERB flag bits are set automatically by the values of the RESET vector address and the interrupt vector address. When a RESET is generated internally, bit 7 of program memory address 0000H is written to the EMB flag, initializing it automatically. When a vectored interrupt is generated, bit 7 of the respective vector address table is written to the EMB. This automatically sets the EMB flag status for the interrupt service routine. When the interrupt is serviced, the EMB value is automatically saved to stack and then restored when the interrupt routine has completed.

At the beginning of a program, the initial EMB and ERB flag values for each vectored interrupt must be set by using VENT instruction. The EMB and ERB can be set or reset by bit manipulation instructions (BITS, BITR) despite the current SMB setting.

PROGRAMMING TIP – Initializing the EMB and ERB Flags

The following assembly instructions show how to initialize the EMB and ERB flag settings:

```

ORG      0000H           ; ROM address assignment
VENT0    1,0,RESET      ; EMB ← 1, ERB ← 0, branch RESET
VENT1    0,1,INTB       ; EMB ← 0, ERB ← 1, branch INTB
VENT2    0,1,INT0       ; EMB ← 0, ERB ← 1, branch INT0
VENT3    0,1,INT1       ; EMB ← 0, ERB ← 1, branch INT1
VENT4    0,1,INTP0      ; EMB ← 0, ERB ← 1, branch INTP0
VENT5    0,1,INTT0      ; EMB ← 0, ERB ← 1, branch INTT0
VENT6    0,1,INTT1      ; EMB ← 0, ERB ← 1, branch INTT1
.
.
.
RESET    BITR           EMB

```

ENABLE MEMORY BANK SETTINGS**EMB = "1"**

When the enable memory bank flag EMB is set to logic one, you can address the data memory bank specified by the select memory bank (SMB) value (0, 1, or 15) using 1-, 4-, or 8-bit instructions. You can use both direct and indirect addressing modes. The addressable RAM areas when EMB = "1" are as follows:

If SMB = 0, 000H-0FFH

If SMB = 1, 100H-1FFH

If SMB = 15, F80H-FFFH

EMB = "0"

When the enable memory bank flag EMB is set to logic zero, the addressable area is defined independently of the SMB value, and is restricted to specific locations depending on whether a direct or indirect address mode is used.

If EMB = "0", the addressable area is restricted to locations 000H-07FH in bank 0 and to locations F80H-FFFH in bank 15 for direct addressing. For indirect addressing, only locations 000H-0FFH in bank 0 are addressable, regardless of SMB value.

To address the peripheral hardware register (bank 15) using indirect addressing, the EMB flag must first be set to "1" and the SMB value to "15". When a RESET occurs, the EMB flag is set to the value contained in bit 7 of ROM address 0000H.

EMB-Independent Addressing

At any time, several areas of the data memory can be addressed independent of the current status of the EMB flag. These exceptions are described in Table 3-1.

Table 3-1. RAM Addressing Not Affected by the EMB Value

Address	Addressing Method	Affected Hardware	Program Examples
000H-0FFH	4-bit indirect addressing using WX and WL register pairs; 8-bit indirect addressing using SP	Not applicable	LD A,@WX LD EA,SP
FB0H-FBFH FF0H-FFFH	1-bit direct addressing	PSW, SCMOD, IEx, IRQx, I/O	BITS EMB BITR IE2
FC0H-FFFH	1-bit indirect addressing using the L register	I/O	BTST F3H.@L BAND C,P3.@L

SELECT BANK REGISTER (SB)

The select bank register (SB) is used to assign the memory bank and register bank. The 8-bit SB register consists of the 4-bit select register bank register (SRB) and the 4-bit select memory bank register (SMB), as shown in Figure 3-2.

During interrupts and subroutine calls, SB register contents can be saved to stack in 8-bit units by the PUSH SB instruction. You later restore the value to the SB using the POP SB instruction.

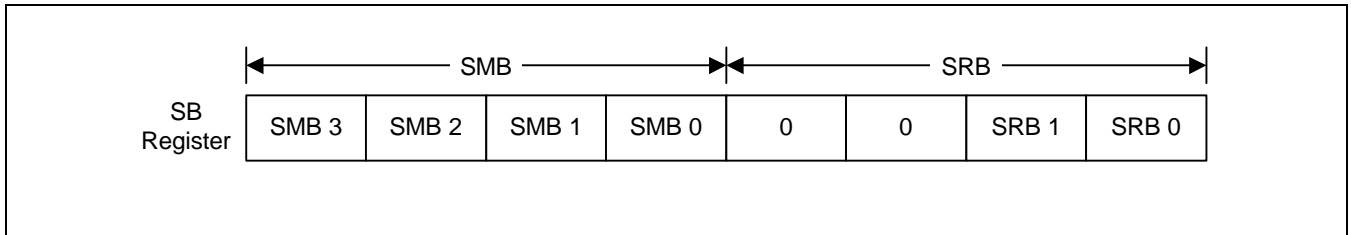


Figure 3-2. SMB and SRB Values in the SB Register

Select Register Bank (SRB) Instruction

The select register bank (SRB) value specifies which register bank is to be used as a working register bank. The SRB value is set by the 'SRB n' instruction, where n = 0, 1, 2, 3.

One of the four register banks is selected by the combination of ERB flag status and the SRB value that is set using the 'SRB n' instruction. The current SRB value is retained until another register is requested by program software. PUSH SB and POP SB instructions are used to save and restore the contents of SRB during interrupts and subroutine calls. RESET clears the 4-bit SRB value to logic zero.

Select Memory Bank (SMB) Instruction

To select one of the three available data memory banks, you must execute an SMB n instruction specifying the number of the memory bank you want (0, 1, or 15). For example, the instruction 'SMB 1' selects bank 1 and 'SMB 15' selects bank 15. (And remember to enable the selected memory bank by making the appropriate EMB flag setting).

The upper four bits of the 12-bit data memory address are stored in the SMB register. If the SMB value is not specified by software (or if a RESET does not occur) the current value is retained. RESET clears the 4-bit SMB value to logic zero.

The PUSH SB and POP SB instructions save and restore the contents of the SMB register to and from the stack area during interrupts and subroutine calls.

DIRECT AND INDIRECT ADDRESSING

1-bit, 4-bit, and 8-bit data stored in data memory locations can be addressed directly using a specific register or bit address as the instruction operand.


Indirect addressing specifies a memory location that contains the required direct address. The KS57 instruction set supports 1-bit, 4-bit, and 8-bit indirect addressing. For 8-bit indirect addressing, an even-numbered RAM address must always be used as the instruction operand.

1-BIT ADDRESSING

Table 3-2. 1-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA.b	Direct: bit is indicated by the RAM address (DA), memory bank selection, and specified bit number (b).	0	000H-07FH	Bank 0	–
			F80H-FFFH	Bank 15	All 1-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 1, 15	
mema.b	Direct: bit is indicated by addressable area (mema) and bit number (b).	x	F80H-FBFH FF0H-FFFH	Bank 15	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.m
memb.@L	Indirect: address is indicated by the upper 6 bits of RAM area (memb) and the upper 2 bits of register L, and bit is indicated by the lower 2 bits of register L.	x	FC0H-FFFH	Bank 15	Pn.m
@H + DA.b	Indirect: bit is indicated by the lower four bits of the address (DA), memory bank selection, and the H register identifier.	0	000H-0FFH	Bank 0	–
		1	000H-FFFH	SMB = 0, 1, 15	All 1-bit addressable peripherals (SMB = 15)

NOTE: 'x' means don't care.

 **PROGRAMMING TIP — 1-Bit Addressing Modes**
1-Bit Direct Addressing

1. If EMB = "0":

AFLAG	EQU	34H.3	
BFLAG	EQU	85H.3	
CFLAG	EQU	0BAH.0	
	SMB	0	
	BITS	AFLAG	; 34H.3 ← 1
	BITS	BFLAG	; F85H.3 ← 1
	BTST	CFLAG	; If FBAH.0 = 1, skip
	BITS	BFLAG	; Else if, FBAH.0 = 0, F85H.3 ← 1
	BITS	P2.0	; FF2H.0 (P2.0) ← 1

2. If EMB = "1":

AFLAG	EQU	34H.3	
BFLAG	EQU	85H.3	
CFLAG	EQU	0BAH.0	
	SMB	0	
	BITS	AFLAG	; 34H.3 ← 1
	BITS	BFLAG	; 85H.3 ← 1
	BTST	CFLAG	; If 0BAH.0 = 1, skip
	BITS	BFLAG	; Else if 0BAH.0 = 0, 085H.3 ← 1
	BITS	P2.0	; FF2H.0 (P2.0) ← 1

1-Bit Indirect Addressing

1. If EMB = "0":

AFLAG	EQU	34H.3	
BFLAG	EQU	85H.3	
CFLAG	EQU	0BAH.0	
	SMB	0	
	LD	H,#0BH	; H ← #0BH
	BTSTZ	@H+CFLAG	; If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
	BITS	CFLAG	; Else if 0BAH.0 = 0, FBAH.0 ← 1

2. If EMB = "1":

AFLAG	EQU	34H.3	
BFLAG	EQU	85H.3	
CFLAG	EQU	0BAH.0	
	SMB	0	
	LD	H,#0BH	; H ← #0BH
	BTSTZ	@H+CFLAG	; If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
	BITS	CFLAG	; Else if 0BAH.0 = 0, 0BAH.0 ← 1

4-BIT ADDRESSING

Table 3-3. 4-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 4-bit address indicated by the RAM address (DA) and the memory bank selection	0	000H-07FH	Bank 0	–
			F80H-FFFH	Bank 15	All 4-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 1, 15	(SMB = 15)
@HL	Indirect: 4-bit address indicated by the memory bank selection and register HL	0	000H-0FFH	Bank 0	–
		1	000H-FFFH	SMB = 0, 1, 15	All 4-bit addressable peripherals (SMB = 15)
@WX	Indirect: 4-bit address indicated by register WX	x	000H-0FFH	Bank 0	–
@WL	Indirect: 4-bit address indicated by register WL	x	000H-0FFH	Bank 0	

NOTE: 'x' means don't care.

PROGRAMMING TIP – 4-Bit Addressing Modes

4-Bit Direct Addressing

1. If EMB = "0":

```

ADATA    EQU    46H
BDATA    EQU    85H
          SMB    15           ; Non-essential instruction, since EMB = "0"
          LD     A,P4        ; A ← (P4)
          SMB    0           ; Non-essential instruction, since EMB = "0"
          LD     ADATA,A     ; (046H) ← A
          LD     BDATA,A     ; (F85H (BMOD)) ← A

```

2. If EMB = "1":

```

ADATA    EQU    46H
BDATA    EQU    85H
          SMB    15
          LD     A,P4        ; A ← (P4)
          SMB    0
          LD     ADATA,A     ; (046H) ← A
          LD     BDATA,A     ; (085H) ← A

```

4-Bit Indirect Addressing (Example 1)

1. If EMB = "0", compare bank 0 locations 040H-046H with bank 0 locations 060H-066H:

```

ADATA    EQU    46H
BDATA    EQU    66H
          SMB    1           ; Non-essential instruction, since EMB = "0"
          LD     HL,#BDATA
          LD     WX,#ADATA
COMP     LD     A,@WL        ; A ← bank 0 (040H-046H)
          CPSE  A,@HL        ; If bank 0 (060H-066H) = A, skip
          SRET
          DECS  L
          JR    COMP
          RET

```

2. If EMB = "1", compare bank 0 locations 040H-046H to bank 1 locations 160H-166H:

```

ADATA    EQU    46H
BDATA    EQU    66H
          SMB    1
          LD     HL,#BDATA
          LD     WX,#ADATA
COMP     LD     A,@WL        ; A ← bank 0 (040H-046H)
          CPSE  A,@HL        ; If bank 1 (160H-166H) = A, skip
          SRET
          DECS  L
          JR    COMP
          RET

```

 **PROGRAMMING TIP — 4-Bit Addressing Modes (Continued)**
4-Bit Indirect Addressing (Example 2)

1. If EMB = "0", exchange bank 0 locations 040H-046H with bank 0 locations 060H-066H:

```

ADATA    EQU    46H
BDATA    EQU    66H
          SMB    1                ; Non-essential instruction, since EMB = "0"
          LD     HL,#BDATA
          LD     WX,#ADATA
TRANS    LD     A,@WL            ; A ← bank 0 (040H-046H)
          XCHD  A,@HL            ; Bank 0 (060H-066H) ↔ A
          JR     TRANS

```

2. If EMB = "1", exchange bank 0 locations 040H-046H to bank 1 locations 160H-166H:

```

ADATA    EQU    46H
BDATA    EQU    66H
          SMB    1
          LD     HL,#BDATA
          LD     WX,#ADATA
TRANS    LD     A,@WL            ; A ← bank 0 (040H-046H)
          XCHD  A,@HL            ; Bank 1 (160H-166H) ↔ A
          JR     TRANS

```

8-BIT ADDRESSING

Table 3-4. 8-Bit Direct and Indirect RAM Addressing

Instruction Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 8-bit address indicated by the RAM address (<i>DA = even number</i>) and memory bank selection	0	000H-07FH	Bank 0	–
			F80H-FFFH	Bank 15	All 8-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 1, 15	
@HL	Indirect: the 8-bit address indicated by the memory bank selection and register HL; (the 4-bit L register value must be an even number)	0	000H-0FFH	Bank 0	–
		1	000H-FFFH	SMB = 0, 1, 15	All 8-bit addressable peripherals (SMB = 15)

 **PROGRAMMING TIP – 8-Bit Addressing Modes**
8-Bit Direct Addressing

1. If EMB = "0":

ADATA	EQU	46H	
BDATA	EQU	8CH	
	SMB	15	; Non-essential instruction, since EMB = "0"
	LD	EA,P4	; E ← (P5), A ← (P4)
	SMB	0	
	LD	ADATA,EA	; (046H) ← A, (047H) ← E
	LD	BDATA,EA	; (F8CH) ← A, (F8DH) ← E

2. If EMB = "1":

ADATA	EQU	46H	
BDATA	EQU	8CH	
	SMB	15	
	LD	EA,P4	; E ← (P5), A ← (P4)
	SMB	0	
	LD	ADATA,EA	; (046H) ← A, (047H) ← E
	LD	BDATA,EA	; (08CH) ← A, (08DH) ← E

8-Bit Indirect Addressing

1. If EMB = "0":

ADATA	EQU	46H	
	SMB	1	; Non-essential instruction, since EMB = "0"
	LD	HL,#ADATA	
	LD	EA,@HL	; A ← (046H), E ← (047H)

2. If EMB = "1":

ADATA	EQU	46H	
	SMB	1	
	LD	HL,#ADATA	
	LD	EA,@HL	; A ← (146H), E ← (147H)

4

MEMORY MAP

OVERVIEW

To support program control of peripheral hardware, I/O addresses for peripherals are memory-mapped to bank 15 of the RAM. Memory mapping lets you use a mnemonic as the operand of an instruction in place of the specific memory location.

Access to bank 15 is controlled by the select memory bank (SMB) instruction and by the enable memory bank flag (EMB) setting. If the EMB flag is "0", bank 15 can be addressed using direct addressing, regardless of the current SMB value. 1-bit direct and indirect addressing can be used for specific locations in bank 15, regardless of the current EMB value.

I/O MAP FOR HARDWARE REGISTERS

Table 4-1 contains detailed information about I/O mapping for peripheral hardware in bank 15 (register locations F80H-FFFH). Use the I/O map as a quick-reference source when writing application programs. The I/O map gives you the following information:

- Register address
- Register name (mnemonic for program addressing)
- Bit values (both addressable and non-manipulable)
- Read-only, write-only, or read and write addressability
- 1-bit, 4-bit, or 8-bit data manipulation characteristics

Table 4-1. I/O Map for Memory Bank 15

Addressing	Symbol	Description	Affected Memory mapped I/O
1-bit direct addressing	DA.b	The bit indicated by memory bank, DA and bit. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 1 bit.
4-bit direct addressing	DA	The address indicated by memory bank and DA. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 4 bits.
8-bit direct addressing	DA	The address (DA specifies an even address) indicated by memory bank and DA. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 8 bits.
4-bit indirect addressing	@HL	The address indicated by memory bank and HL register. (EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 4 bits.
8-bit indirect addressing	@HL	The address indicated by memory bank and HL (the contents of the L register are even). (EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 8 bit.
1-bit manipulating addressing	mema.b	The bit indicated by mema and bit. (regardless of the status of EMB and SMB)	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.m
	memb.@L	The bit indicated by the lower 2 bits of the L register of the address indicated by the upper 10 bits of memb and the upper 2 bits of the L register. (regardless of the status of EMB and SMB)	Pn.m
	@H+DA.b	The bit of the address indicated by memory bank, H register and the lower 4 bits of DA. (EMB=1 and SMB=15)	All peripheral hardware that can be manipulated in 1 bit.

Table 4-2. I/O Map for Memory Bank 15

Memory Bank 15							Addressing Mode		
Address	Register	Bit 3	Bit 2	Bit 1	Bit 0	R/W	1-Bit	4-Bit	8-Bit
F80H	SP	.3	.2	.1	"0"	R/W	No	No	Yes
F81H		.7	.6	.5	.4				
Locations F82H-F84H are not mapped.									
F85H	BMOD	.3	.2	.1	.0	W	.3	Yes	No
F86H	BCNT					R	No	No	Yes
F87H									
F88H	WMOD	.3	.2	.1	.0	W	.3 ⁽¹⁾	No	Yes
F89H		.7	"0"	.5	.4				
F8AH	LCNST	.3	.2	.1	.0	W	No	No	Yes
F8BH		.7	"0"	"0"	"0"				
F8CH	LMOD	.3	.2	.1	.0	W	No	No	Yes
F8DH		"0"	.6	.5	.4				
F8EH	LCON0	.3	.2	.1	.0	W	No	Yes	No
F8FH	LCON1	.3	.2	.1	.0	W	No	Yes	No
F90H	TMOD0	.3	.2	"0"	"0"	W	.3	No	Yes
F91H		"0"	.6	.5	.4				
F92H		"0"	TOE0	"0"	"0"	R/W	Yes	Yes	No
Location F93H is not mapped.									
F94H	TCNT0					R	No	No	Yes
F95H									
F96H	TREF0					W	No	No	Yes
F97H									
F98H	WDMOD	.3	.2	.1	.0	W	No	No	Yes
F99H		.7	.6	.5	.4				
F9AH	WDFLAG ⁽²⁾	WDTCF	"0"	"0"	"0"	W	.3	Yes	No
Locations F9BH-F9FH are not mapped.									
FA0H	PASR	.3	.2	.1	.0	W	No	No	Yes
FA1H		"0"	"0"	"0"	.4				
FA2H	KSR0	.3	.2	.1	.0	W	No	Yes	No
FA3H	KSR1	.3	.2	.1	.0				
FA4H	KSR2	.3	.2	.1	.0				
FA5H	KSR3	.3	.2	.1	.0				

Table 4-2. I/O Map for Memory Bank 15 (Continued)

Memory Bank 15							Addressing Mode		
Address	Register	Bit 3	Bit 2	Bit 1	Bit 0	R/W	1-Bit	4-Bit	8-Bit
FA6H	TMOD1	.3	.2	"0"	"0"	W	.3	No	Yes
FA7H		"0"	.6	.5	.4				
FA8H	TCNT1					R	No	No	Yes
FA9H									
FAAH	TREF1					W	No	No	Yes
FABH									
Locations FACH-FAFH are not mapped.									
FB0H	PSW	IS1	IS0	EMB	ERB	R/W	Yes	Yes	Yes
FB1H		C ⁽³⁾	SC2	SC1	SC0	R	No	No	
FB2H	IPR	IME	.2	.1	.0	W	IME	Yes	No
FB3H	PCON	.3	.2	.1	.0	W	No	Yes	No
FB4H	IMOD0	"0"	"0"	.1	.0	W	No	Yes	No
FB5H	IMOD1	"0"	"0"	.1	.0	W	No	Yes	No
FB6H	IMOD2	"0"	.2	.1	.0	W	No	Yes	No
FB7H	SCMOD	.3	.2	"0"	.0	W	Yes	No	No
FB8H		"0"	"0"	IEB	IRQB	R/W	Yes	Yes	No
Location FB9H is not mapped.									
FBAH		"0"	"0"	IEW	IRQW	R/W	Yes	Yes	No
FBBH		"0"	"0"	IET1	IRQT1	R/W	Yes	Yes	No
FBCH		"0"	"0"	IET0	IRQT0	R/W	Yes	Yes	No
FBDH		"0"	"0"	IEP0	IRQP0	R/W	Yes	Yes	No
FBEH		IE1	IRQ1	IE0	IRQ0	R/W	Yes	Yes	No
FBFH		"0"	"0"	IE2	IRQ2	R/W	Yes	Yes	No
FC0H		BSC0					R/W	Yes	Yes
FC1H	BSC1								
FC2H	BSC2								
FC3H	BSC3								
Locations FC4H-FC9H are not mapped.									
FCAH	EMAR0	.3	.2	.1	.0	R/W	No	No	Yes
FCBH		.7	.6	.5	.4				
FCCH	EMAR1	.3/.11	.2/.10	.1/.9	.0/.8				
FCDH		.7/.15	.6/.14	.5/.13	.4/.12				
FCEH	EMAR2	"0"	.2/.18	.1/.17	.0/.16	R/W	No	Yes	No
Location FCFH is not mapped.									

Table 4-2. I/O Map for Memory Bank 15 (Continued)

Memory Bank 15							Addressing Mode		
Address	Register	Bit 3	Bit 2	Bit 1	Bit 0	R/W	1-Bit	4-Bit	8-Bit
FD0H	CLMOD	.3	"0"	.1	.0	W	No	Yes	No
Location FD1H is not mapped.									
FD2H	EMCON	.3	.2	.1	.0	W	No	No	Yes
FD3H		.7	.6	.5	.4				
FD4H	EMDR0	.3	.2	.1	.0	R/W	No	No	Yes
FD5H		.7	.6	.5	.4				
Locations FD6H-FE5H are not mapped.									
FE6H	PNE0 ⁽⁵⁾	.3	.2	.1	.0	W	No	No	Yes
FE7H		.7	.6	.5	.4				
FE8H	PUMOD0 ⁽⁶⁾	"0"	"0"	"0"	PUR0	W	No	No	Yes
FE9H		PUR7	PUR6	PUR5	PUR4				
FEAH	PMG0	PM0.3	PM0.2	PM0.1	PM0.0	W	No	No	Yes
FEBH		"0"	PM1.2	PM1.1	PM1.0				
FECH	PMG1	PM4.3	PM4.2	PM4.1	PM4.0	W	No	No	Yes
FEDH		PM5.3	PM5.2	PM5.1	PM5.0				
FEEH	PMG2	PM6.3	PM6.2	PM6.1	PM6.0	W	No	No	Yes
FEFH		PM7.3	PM7.2	PM7.1	PM7.0				
FF0H	Port0 (P0)	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF1H	Port1 (P1)	"0"	.2/.6	.1/.5	.0/.4				
Locations FF2H-FF3H are not mapped.									
FF4H	Port4 (P4)	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF5H	Port5 (P5)	.3/.7	.2/.6	.1/.5	.0/.4				
FF6H	Port6 (P6)	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF7H	Port7 (P7)	.3/.7	.2/.6	.1/.5	.0/.4				

NOTES:

1. Bit 3 in the WMOD register is read only.
2. F9AH.0, F9AH.1 and F9AH.2 are fixed to "0".
3. The carry flag can be read or written by specific bit manipulation instructions only.
4. The PNE0 register is used to select the output types of ports 4,5 (zero: push-pull output type, one: open-drain output type). The reset value of the PNE1 register is "00H".
5. The PUMOD0 register is used to enable/disable the internal pull-up resistors of port 0, 1, 4, 5, 6 and 7 (zero: disable, one: enable). The reset value of the PUMOD0 register is "00H".

REGISTER DESCRIPTIONS

In this section, register descriptions are presented in a consistent format to familiarize you with the memory-mapped I/O locations in bank 15 of the RAM. Figure 4-1 describes features of the register description format. Register descriptions are arranged in alphabetical order. Programmers can use this section as a quick-reference source when writing application programs.

Counter registers, buffer registers, and reference registers, as well as the stack pointer and port I/O latches, are not included in these descriptions. More detailed information about how these registers are used is included in Part II of this manual, "Hardware Descriptions," in the context of the corresponding peripheral hardware module descriptions.

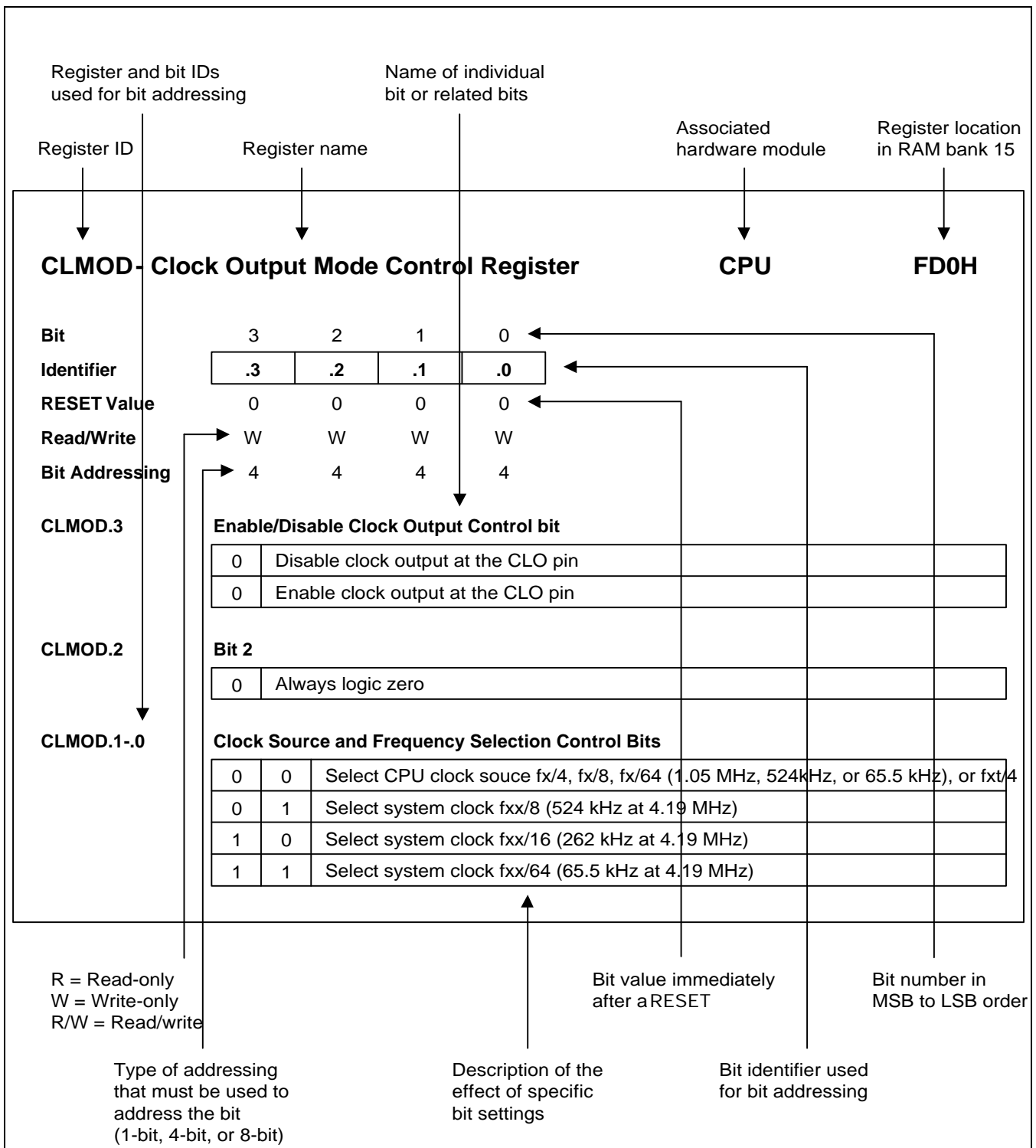


Figure 4-1. Register Description Format

BMOD — Basic Timer Mode Register

BT

F85H

Bit	3	2	1	0
Identifier	.3	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

BMOD.3**Basic Timer Restart Bit**

1	Restart basic timer, then clear IRQB flag, BCNT and BMOD.3 to logic zero
---	--

BMOD.2-0**Input Clock Frequency and Interrupt Interval Time**

0	0	0	Input clock frequency: Interrupt interval time (wait time):	$f_{xx}/2^{12}$ (1.02 kHz) $2^{20}/f_{xx}$ (250 ms)
0	1	1	Input clock frequency: Interrupt interval time (wait time):	$f_{xx}/2^9$ (8.18 kHz) $2^{17}/f_{xx}$ (31.3 ms)
1	0	1	Input clock frequency: Interrupt interval time (wait time):	$f_{xx}/2^7$ (32.7 kHz) $2^{15}/f_{xx}$ (7.82 ms)
1	1	1	Input clock frequency: Interrupt interval time (wait time):	$f_{xx}/2^5$ (131 kHz) $2^{13}/f_{xx}$ (1.95 ms)

NOTES:

- When a RESET occurs, the oscillator stabilization wait time is 31.3 ms ($2^{17}/f_{xx}$) at 4.19 MHz.
- 'fxx' is the system clock frequency (assume that fxx is 4.19 MHz).

CLMOD — Clock Output Mode Register

CPU

FD0H

Bit	3	2	1	0
Identifier	.3	"0"	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

CLMOD.3**Enable/Disable Clock Output Control Bit**

0	Disable clock output at the CLO pin
1	Enable clock output at the CLO pin

CLMOD.2**Bit 2**

0	Always logic zero
---	-------------------

CLMOD.1-0**Clock Source and Frequency Selection Control Bits**

0	0	Select CPU clock source $fx/4$, $fx/8$, or $fx/64$ (1 MHz, 524 kHz, or 65.5 kHz) or $fx/4$
0	1	Select system clock $fx/8$ (524 kHz)
1	0	Select system clock $fx/16$ (262 kHz)
1	1	Select system clock $fx/64$ (65.5 kHz)

NOTE: 'fxx' is the system clock, given a clock frequency of 4.19 MHz.

EMCON — External Memory Control Register CPU FD3H, FD2H

Bit	7	6	5	4	3	2	1	0
Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	R/W
Bit Addressing	8	8	8	8	8	8	8	8

EMCON.7**Memory Read/Write Control Bit**

0	Memory read signal output
1	Memory write signal output

EMCON.6-5**Memory Access Clock Selection Bits**

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

EMCON.4**Address Increment Control Bit**

0	The address(a value of EMAR2 - EMAR0) is not increased automatically after memory access.
1	The address(a value of EMAR2 - EMAR0) is increased automatically after memory access.

EMCON.3-1**Memory Selection Bits**

.3	.2	.1	External data memory selection
0	0	0	Data memory 0(DM0 active)
0	0	1	Data memory 1(DM1 active)
0	1	0	Data memory 2(DM2 active)
0	1	1	Data memory 3(DM3 active)
1	0	0	Data memory 4(DM4 active)
1	0	1	Data memory 5(DM5 active)

EMCON.0**Memory Access Start Bit**

0	Not busy (read)
1	Start a memory access (write), Busy (read)

NOTES:

- When it reads data from a external memory, the data are written to the register EMDR0.
- When it writes data to a external memory, the data to the register EMDR0 are written to a external memory.
- The external memory selection pins of P6.0/DM0 - P7.1/DM5 should be set to push-pull output and the latches should be set to logic "1".
- P7.1/DM5/COM11 is not used as DM5, when it is selected to COM. For using DM5, this pin is set to output High. (in this case, this DM signal is falling to low on access start.)
- EMCON.0 is cleared automatically when memory access is finished.

IE0, IRQ0 — INT0 Interrupt Enable/Request Flags

CPU

FBEH

IE1, IRQ1 — INT1 Interrupt Enable/Request Flags

CPU

FBEH

Bit	3	2	1	0
Identifier	IE1	IRQ1	IE0	IRQ0
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

IE1**INT1 Interrupt Enable Flag**

0	Disable interrupt requests at the INT1 pin
1	Enable interrupt requests at the INT1 pin

IRQ1**INT1 Interrupt Request Flag**

–	Generate INT1 interrupt (This bit is set and cleared by hardware when rising or falling edge detected at INT1 pin.)
---	---

IE0**INT0 Interrupt Enable Flag**

0	Disable interrupt requests at the INT0 pin
1	Enable interrupt requests at the INT0 pin

IRQ0**INT0 Interrupt Request Flag**

–	Generate INT0 interrupt (This bit is set and cleared automatically by hardware when rising or falling edge detected at INT0 pin.)
---	---

IE2, IRQ2 — INT2 Interrupt Enable/Request Flags

CPU

FBFH

Bit	3	2	1	0
Identifier	"U"	"U"	IE2	IRQ2
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-.2

Bits 3-2

U	This bit is undefined.
---	------------------------

IE2

INT2 Interrupt Enable Flag

0	Disable INT2 interrupt requests at the KS0-KS7 pins
1	Enable INT2 interrupt requests at the KS0-KS7 pins

IRQ2

INT2 Interrupt Request Flag

–	Generate INT2 quasi-interrupt (This bit is set and is not cleared automatically by hardware when a falling edge is detected at one of the KS0-KS7 pins. Since INT2 is a quasi-interrupt, IRQ2 flag must be cleared by software.)
---	--

NOTE: The "U" means a undefined register bit

IEB, IRQB — INTB Interrupt Enable/Request Flags

CPU

FB8H

Bit	3	2	1	0
Identifier	"U"	"U"	IEB	IRQB
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-.2

Bits 3-2

U	This bit is undefined.
---	------------------------

IEB

INTB Interrupt Enable Flag

0	Disable INTB interrupt requests
1	Enable INTB interrupt requests

IRQB

INTB Interrupt Request Flag

–	Generate INTB interrupt (This bit is set and cleared automatically by hardware when reference interval signal received from basic timer.)
---	---

NOTE: The "U" means a undefined register bit.

IEP0, IRQP0 — INTP0 Interrupt Enable/Request Flags CPU FBDH

Bit	3	2	1	0
Identifier	"U"	"U"	IEP0	IRQP0
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-2

Bits 3-2

U	This bit is undefined.
---	------------------------

IEP0

INTS Interrupt Enable Flag

0	Disable INTP0 interrupt requests
1	Enable INTP0 interrupt requests

IRQP0

INTS Interrupt Request Flag

–	Generate INTP0 interrupt (This bit is set and cleared automatically by hardware when falling edge is detected at K0-K3 pin.)
---	--

NOTES:

1. The "U" means a undefined register bit.
2. To use INTP0 interrupt, P0 and P1 must be set to external interrupt pins by LMOD.6-LMOD.4, input mode by PMG0 and pull-up resistor enable by PUMOD0.

IET0, IRQT0 — INTT0 Interrupt Enable/Request Flags CPU FBCH

Bit	3	2	1	0
Identifier	"U"	"U"	IET0	IRQT0
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-.2

Bits 3-2

U	This bit is undefined.
---	------------------------

IET0

INTT0 Interrupt Enable Flag

0	Disable INTT0 interrupt requests
1	Enable INTT0 interrupt requests

IRQT0

INTT1 Interrupt Request Flag

–	Generate INTT0 interrupt (This bit is set and cleared automatically by hardware when contents of TCNT0 and TREF0 registers match.)
---	--

NOTE: The "U" means a undefined register bit.

IET1, IRQT1 — INTT1 Interrupt Enable/Request Flags CPU FBBH

Bit	3	2	1	0
Identifier	"U"	"U"	IET1	IRQT1
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-.2

Bits 3-2

U	This bit is undefined.
---	------------------------

IET1

INTT1 Interrupt Enable Flag

0	Disable INTT1 interrupt requests
1	Enable INTT1 interrupt requests

IRQT1

INTT1 Interrupt Request Flag

–	Generate INTT1 interrupt (This bit is set and cleared automatically by hardware when contents of TCNT1 and TREF1 registers match.)
---	--

NOTE: The "U" means a undefined register bit.

IEW, IRQW — INTW Interrupt Enable/Request Flags

CPU

FBAH

Bit	3	2	1	0
Identifier	"U"	"U"	IEW	IRQW
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3-2

Bits 3-2

U	This bit is undefined.
---	------------------------

IEW

INTW Interrupt Enable Flag

0	Disable INTW interrupt requests
1	Enable INTW interrupt requests

IRQW

INTW Interrupt Request Flag

–	Generate INTW interrupt (This bit is set when the timer interval is set to 0.5 seconds or 3.91 milliseconds.)
---	---

NOTES:

1. Since INTW is a quasi-interrupt, the IRQW flag must be cleared by software.
2. The "U" means a undefined register bit.

IMOD0 — External Interrupt 0 (INT0) Mode Register

CPU

FB4H

Bit	3	2	1	0
Identifier	"0"	"0"	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD0.3-2**Bits 3-2**

0	Always logic zero
---	-------------------

IMOD0.1-0**External Interrupt Mode Control Bits**

0	0	Interrupt requests are triggered by a rising edge
0	1	Interrupt requests are triggered by a falling edge
1	0	Interrupt requests are triggered by both rising and falling edges
1	1	Interrupt request flag (IRQ0) cannot be set to logic one

IMOD1 — External Interrupt 1 (INT1) Mode Register

CPU

FB5H

Bit	3	2	1	0
Identifier	"0"	"0"	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD1.3-2**Bits 3-2**

0	Always logic zero
---	-------------------

IMOD1.1-0**External Interrupt Mode Control Bits**

0	0	Interrupt requests are triggered by a rising edge
0	1	Interrupt requests are triggered by a falling edge
1	0	Interrupt requests are triggered by both rising and falling edges
1	1	Interrupt request flag (IRQ1) cannot be set to logic one

IMOD2 — External Interrupt 2 (INT2) Mode Register

CPU

FB6H

Bit	3	2	1	0
Identifier	"0"	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD2.3**Bits 3**

0	Always logic zero
---	-------------------

IMOD2.2-0**External Interrupt 2 Edge Detection Selection Bit**

0	0	0	Interrupt request at KS0-KS3 triggered by falling edge
0	0	1	Interrupt request at KS0-KS4 triggered by falling edge
0	1	0	Interrupt request at KS0-KS5 triggered by falling edge
0	1	1	Interrupt request at KS0-KS6 triggered by falling edge
1	0	0	Interrupt request at KS0-KS7 triggered by falling edge
1	0	1	Not available
1	1	0	
1	1	1	

IPR — Interrupt Priority Register

CPU

FB2H

Bit	3	2	1	0
Identifier	IME	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

IME**Interrupt Master Enable Bit**

0	Disable all interrupt processing
1	Enable processing for all interrupt service requests

IPR.2-.0**Interrupt Priority Register Setting**

IPR.2	IPR.1	IPR.0	Result of IPR Bit Setting
0	0	0	Normal interrupt handling according to default priority settings
0	0	1	Process INTB interrupt at highest priority
0	1	0	Process INT0 interrupt at highest priority
0	1	1	Process INT1 interrupt at highest priority
1	0	0	Process INTP0 interrupt at highest priority
1	0	1	Process INTT0 interrupt at highest priority
1	1	0	Process INTT1 interrupt at highest priority
1	1	1	Not available

LCNST — LCD Contrast Control Register

LCD

F8BH, F8AH

Bit	7	6	5	4	3	2	1	0
Identifier	.7	"0"	"0"	"0"	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

LCNST.7

Enable/Disable LCD Contrast Control Bit

0	Disable LCD contrast control
1	Enable LCD contrast control

LCNST.6-4

Bits 6-4

0	Always logic zero
---	-------------------

LCNST.3-0

LCD Contrast Level Control Bits(16 steps)

.3	.2	.1	.0	Step
0	0	0	0	1/16 step (The dimmest level)
0	0	0	1	2/16 step
0	0	1	0	3/16 step
		•		•••••
		•		
		•		
		•		
1	1	1	1	16/16 step (The brightest level)

NOTE: $V_{LCD} = V_{DD} \times (1 - (16-n)/48)$, where $n = 0 - 15$ (At normal LCD dividing resistors).

LCON0 — LCD Output Control Register 0

LCD

F8EH

Bit	3	2	1	0
Identifier	.3	.2	"0"	"0"
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

LCON0.3-.2**Bits3-2**

0	0	1/9 duty(COM0-COM8 select)
0	1	1/10 duty(COM0-COM9 select)
1	0	1/11 duty(COM0-COM10 select)
1	1	1/12 duty(COM0-COM11 select)

LCON0.1-.0**Bits 1-0**

0	Always logic zero
---	-------------------

NOTES:

1. COM has priority over normal port in P7.3/COM9-P7.1/COM11. This means these port are assigned to COM pins regardless of the value of PMG2, when duty is selected to 1/10, 1/11, or 1/12 at LCON0.
2. The port used COM must be set to output to prevent LCD display distortion.

LCON1 — LCD Output Control Register 1

LCD

F8FH

Bit	3	2	1	0
Identifier	.3	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

LCON1.3-.0**Bits 3-0**

0	1	0	0	LCD display on
0	1	0	1	Dimming mode
1	0	0	1	Key check signal output with LCD display off

NOTES:

1. To turn off LCD display, you must set LCON1 to 9 not 0.
2. P8 can be used to normal output port, when LCD display is off. The value of P8 is determined by KSR0-KSR3 regardless of LMOD.0. (refer to P12-17)

LMOD — LCD Mode Register

LCD

F8DH,F8CH

Bit	7	6	5	4	3	2	1	0
Identifier	"0"	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

LMOD.7**Bit 7**

0	Always logic zero
---	-------------------

LMOD.6 - .4**External Interrupt (INTP0) Pins Selection Bits ⁽¹⁾**

0	0	0	Interrupt request at K0 triggered by falling edge
0	0	1	Interrupt request at K0-K1 triggered by falling edge
0	1	0	Interrupt request at K0-K2 triggered by falling edge
0	1	1	Interrupt request at K0-K3 triggered by falling edge
1	0	0	Interrupt request at K0-K4 triggered by falling edge
1	0	1	Interrupt request at K0-K5 triggered by falling edge
1	1	0	Interrupt request at K0-K6 triggered by falling edge
1	1	1	Interrupt request flag (IRQP0) cannot be set to logic one

LMOD.3 - .2**Watch Timer Clock Selection Bits ⁽²⁾**

.3	.2	When main system clock is selected as watch timer clock by WMOD.0
0	0	fx/128
0	1	fx/64
1	0	fx/32
1	1	fx/16

LMOD.1**LCD Dividing Resistor Selection Bit**

0	Normal LCD dividing resistors
1	Diminish LCD dividing resistors to strengthen LCD drive

LMOD.0**Key Strobe Signal Output Control Bit(SEG0/P8.0 - SEG15/P8.15)**

0	Enable key strobe signal output
1	Disable key strobe signal output ⁽³⁾

NOTES:

- The pins which are not selected as external interrupts(K0 - K6) can be used to normal I/O. To use external interrupts, corresponding pins must be set to input and pull-up enable mode.
- LCD clock can be selected only when main clock(fx) is used as clock source of watch timer. When sub clock(fxt) is used as clock source of watch timer, LCD clock is always fw/48(1/9 duty), fw/44(1/10 duty), fw/40(1/11 duty), or fw/36(1/12 duty).
- Refer to page 12-7.

PASR — Page Selection Register

MEMORY

FA1H,FA0H

Bit	7	6	5	4	3	2	1	0
Identifier	"0"	"0"	"0"	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PASR.7-5**Bits 7-5**

0	Always logic zero
---	-------------------

PASR.4-0**Page Selection Register in the Bank1**

0	0	0	0	0	00H page in the Bank1
0	0	0	0	1	01H page in the Bank1
		•			
		•			
		•			•••••
		•			
		•			
1	0	0	1	1	13H page for LCD display register in the Bank1

PCON — Power Control Register

CPU

FB3H

Bit	3	2	1	0
Identifier	.3	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

PCON.3 - .2**CPU Operating Mode Control Bits**

0	0	Enable normal CPU operating mode
0	1	Initiate idle power-down mode
1	0	Initiate stop power-down mode

PCON.1 - .0**CPU Clock Frequency Selection Bits**

0	0	If SCMOD.0 = "0", fx/64; if SCMOD.0 = "1", fxt/4
1	0	If SCMOD.0 = "0", fx/8; if SCMOD.0 = "1", fxt/4
1	1	If SCMOD.0 = "0", fx/4; if SCMOD.0 = "1", fxt/4

NOTE: 'fx' is the main system clock; 'fxt' is the subsystem clock.

PMG0 — PORT I/O MODE REGISTER 0 (Group 0: Port 0,1) I/O FEBH, FEAH

Bit	7	6	5	4	3	2	1	0
Identifier	"0"	PM1.2	PM1.1	PM1.0	PM0.3	PM0.2	PM0.1	PM0.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PMG0.7**Bit 7**

0	Always logic zero
---	-------------------

PM1.2**P1.2 I/O Mode Selection Flag**

0	Set P1.2 to input mode
1	Set P1.2 to output mode

PM1.1**P1.1 I/O Mode Selection Flag**

0	Set P1.1 to input mode
1	Set P1.1 to output mode

PM1.0**P1.0 I/O Mode Selection Flag**

0	Set P1.0 to input mode
1	Set P1.0 to output mode

PM0.3**P0.3 I/O Mode Selection Flag**

0	Set P0.3 to input mode
1	Set P0.3 to output mode

PM0.2**P0.2 I/O Mode Selection Flag**

0	Set P0.2 to input mode
1	Set P0.2 to output mode

PM0.1**P0.1 I/O Mode Selection Flag**

0	Set P0.1 to input mode
1	Set P0.1 to output mode

PM0.0**P0.0 I/O Mode Selection Flag**

0	Set P0.0 to input mode
1	Set P0.0 to output mode

NOTE: To used INTP0 interrupt, P0 and P1 must be set to external interrupt pins by LMOD.6-LMOD.4, input mode by PMG0 and pull-up resistor enable by PUMOD0.

PMG1 — PORT I/O MODE REGISTER 1(Group 1: Port 4,5) I/O **FEDH, FECH**

Bit	7	6	5	4	3	2	1	0
Identifier	PM5.3	PM5.2	PM5.1	PM5.0	PM4.3	PM4.2	PM4.1	PM4.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM5.3 P5.3 I/O Mode Selection Flag

0	Set P5.3 to input mode
1	Set P5.3 to output mode

PM5.2 P5.2 I/O Mode Selection Flag

0	Set P5.2 to input mode
1	Set P5.2 to output mode

PM5.1 P5.1 I/O Mode Selection Flag

0	Set P5.1 to input mode
1	Set P5.1 to output mode

PM5.0 P5.0 I/O Mode Selection Flag

0	Set P5.0 to input mode
1	Set P5.0 to output mode

PM4.3 P4.3 I/O Mode Selection Flag

0	Set P4.3 to input mode
1	Set P4.3 to output mode

PM4.2 P4.2 I/O Mode Selection Flag

0	Set P4.2 to input mode
1	Set P4.2 to output mode

PM4.1 P4.1 I/O Mode Selection Flag

0	Set P4.1 to input mode
1	Set P4.1 to output mode

PM4.0 P4.0 I/O Mode Selection Flag

0	Set P4.0 to input mode
1	Set P4.0 to output mode

PMG2 — Port I/O Mode Register 2 (Group 2: Port 6,7) I/O FEFH, FEEH

Bit	7	6	5	4	3	2	1	0
Identifier	PM7.3	PM7.2	PM7.1	PM7.0	PM6.3	PM6.2	PM6.1	PM6.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM7.3**P7.3 I/O Mode Selection Flag**

0	Set P7.3 to input mode
1	Set P7.3 to output mode

PM7.2**P7.2 I/O Mode Selection Flag**

0	Set P7.2 to input mode
1	Set P7.2 to output mode

PM7.1**P7.1 I/O Mode Selection Flag**

0	Set P7.1 to input mode
1	Set P7.1 to output mode

PM7.0**P7.0 I/O Mode Selection Flag**

0	Set P7.0 to input mode
1	Set P7.0 to output mode

PM6.3**P6.3 I/O Mode Selection Flag**

0	Set P6.3 to input mode
1	Set P6.3 to output mode

PM6.2**P6.2 I/O Mode Selection Flag**

0	Set P6.2 to input mode
1	Set P6.2 to output mode

PM6.1**P6.1 I/O Mode Selection Flag**

0	Set P6.1 to input mode
1	Set P6.1 to output mode

PM6.0**P6.0 I/O Mode Selection Flag**

0	Set P6.0 to input mode
1	Set P6.0 to output mode

NOTE: COM has priority over normal port in P7.3/COM9-P7.1/COM11. This means these port are assigned to COM pins regardless of the value of PMG2, when duty is selected to 1/10, 1/11 or 1/12 at LCON0.

PNE0 — N-Channel Open-Drain Mode Register 0 **I/O** **FE7H, FE6H**

Bit	7	6	5	4	3	2	1	0
Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PNE0.7**P5.3 N-Channel Open-Drain Configurable Bit**

0	Configure P5.3 as a push-pull
1	Configure P5.3 as a n-channel open-drain

PNE0.6**P5.2 N-Channel Open-Drain Configurable Bit**

0	Configure P5.2 as a push-pull
1	Configure P5.2 as a n-channel open-drain

PNE0.5**P5.1 N-Channel Open-Drain Configurable Bit**

0	Configure P5.1 as a push-pull
1	Configure P5.1 as a n-channel open-drain

PNE0.4**P5.0 N-Channel Open-Drain Configurable Bit**

0	Configure P5.0 as a push-pull
1	Configure P5.0 as a n-channel open-drain

PNE0.3**P4.3 N-Channel Open-Drain Configurable Bit**

0	Configure P4.3 as a push-pull
1	Configure P4.3 as a n-channel open-drain

PNE0.2**P4.2 N-Channel Open-Drain Configurable Bit**

0	Configure P4.2 as a push-pull
1	Configure P4.2 as a n-channel open-drain

PNE0.1**P4.1 N-Channel Open-Drain Configurable Bit**

0	Configure P4.1 as a push-pull
1	Configure P4.1 as a n-channel open-drain

PNE0.0**P4.0 N-Channel Open-Drain Configurable Bit**

0	Configure P4.0 as a push-pull
1	Configure P4.0 as a n-channel open-drain

PSW — Program Status Word

CPU

FB1H,FB0H

Bit	7	6	5	4	3	2	1	0
Identifier	C	SC2	SC1	SC0	IS1	IS0	EMB	ERB
RESET Value	(1)	0	0	0	0	0	0	0
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W
Bit Addressing	(2)	8	8	8	1/4/8	1/4/8	1/4/8	1/4/8

C**Carry Flag**

0	No overflow or borrow condition exists
1	An overflow or borrow condition does exist

SC2-SC0**Skip Condition Flags**

0	No skip condition exists; no direct manipulation of these bits is allowed
1	A skip condition exists; no direct manipulation of these bits is allowed

IS1, IS0**Interrupt Status Flags**

0	0	Service all interrupt requests
0	1	Service only the high-priority interrupt(s) as determined in the interrupt priority register (IPR)
1	0	Do not service any more interrupt requests
1	1	Undefined

EMB**Enable Data Memory Bank Flag**

0	Restrict program access to data memory to bank 15 (F80H-FFFH) and to the locations 000H-07FH in the bank 0 only
1	Enable full access to data memory banks 0, 1, and 15

ERB**Enable Register Bank Flag**

0	Select register bank 0 as working register area
1	Select register banks 0, 1, 2, or 3 as working register area in accordance with the select register bank (SRB) instruction operand

NOTES:

1. The value of the carry flag after a RESET occurs during normal operation is undefined. If a RESET occurs during power-down mode (IDLE or STOP), the current value of the carry flag is retained.
2. The carry flag can only be addressed by a specific set of 1-bit manipulation instructions. See Section 2 for detailed information.

PUMOD0 — Pull-Up Resistor Mode Register

I/O

FE9H, FE8H

Bit	7	6	5	4	3	2	1	0
Identifier	PUR7	PUR6	PUR5	PUR4	"0"	"0"	"0"	PUR0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PUR7**Connect/Disconnect Port 7 Pull-Up Resistor Control Bit**

0	Disconnect port 7 pull-up resistor
1	Connect port 7 pull-up resistor

PUR6**Connect/Disconnect Port 6 Pull-Up Resistor Control Bit**

0	Disconnect port 6 pull-up resistor
1	Connect port 6 pull-up resistor

PUR5**Connect/Disconnect Port 5 Pull-Up Resistor Control Bit**

0	Disconnect port 5 pull-up resistor
1	Connect port 5 pull-up resistor

PUR4**Connect/Disconnect Port 4 Pull-Up Resistor Control Bit**

0	Disconnect port 4 pull-up resistor
1	Connect port 4 pull-up resistor

PUMOD0.3-1**Bit 3-1**

0	Always logic zero
---	-------------------

PUR0**Connect/Disconnect Port 0,1 Pull-Up Resistor Control Bit**

0	Disconnect port 0,1 pull-up resistor
1	Connect port 0,1 pull-up resistor

NOTES:

1. If port is set to output mode, pull-up resistor is disabled automatically.
2. When P0, P1 are used to external interrupt pins, the pull-up resistors of input mode are determined by key strobe signal (refer to P12-7).

SCMOD — System Clock Mode Control Register

CPU

FB7H

Bit	3	2	1	0
Identifier	.3	.2	"0"	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1	1	1	1

SCMOD.3**Bit 3**

0	Enable main system clock
1	Disable main system clock

SCMOD.2**Bit 2**

0	Enable sub system clock
1	Disable sub system clock

SCMOD.1**Bit 1**

0	Always logic zero
---	-------------------

SCMOD.0**Bit 0**

0	Select main system clock
1	Select sub system clock

NOTES:

- Sub-oscillation goes into stop mode only by SCMOD.2. PCON which revokes stop mode cannot stop the sub-oscillation.
- You can use SCMOD.2 as follows (ex; after data bank was used, a few minutes have passed):
Main operation → sub-operation → sub-idle (LCD on, after a few minutes later without any external input) → sub operation → main operation → SCMOD.2 = 1 → main stop mode (LCD off).
- SCMOD bit3-0 can not be modified simultaneously by a 4-bit instruction; They can only be modified by separate 1-bit instructions.

TMOD0 — Timer/Counter 0 Mode Register

T/C0

F90H,F91H

Bit	7	6	5	4	3	2	1	0
Identifier	"0"	.6	.5	.4	.3	.2	"0"	"0"
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	w	W	W	W
Bit Addressing	8	8	8	8	1/8	8	8	8

TMOD0.7**Bit 7**

0	Always logic zero
---	-------------------

TMOD0.6-.4**Timer/Counter0 Input Clock Selection Bits ^{(1) (2)}**

0	0	0	External clock input at TCL0 pin on rising edge
0	0	1	External clock input at TCL0 pin on falling edge
0	1	x	fxt(Subsystem clock: 32.768 kHz)
1	0	0	fxx /2 ¹⁰ (4.09 kHz)
1	0	1	fxx/2 ⁶ (65.5 kHz)
1	1	0	fxx/2 ⁴ (262kHz)
1	1	1	fxx (4.19 MHz)

TMOD0.3**Clear Counter And Resume Counting Control Bit**

1	Clear TCNT0,IRQT0,and TOL0 and resume counting immediately.(This bit is cleared automatically when counting starts)
---	---

TMOD0.2**Enable/Disable Timer/Counter0 Bit**

0	Disable timer/counter0 ; retain TCNT0 contents
1	Enable timer/counter0

TMOD0.1**Bit 1**

0	Always logic zero
---	-------------------

TMOD0.0**Bit 0**

0	Always logic zero
---	-------------------

NOTES:

1. 'fxx' is the system clock frequency (assume that fxx is 4.19MHz).
2. 'x' is don't care.

TMOD1 — Timer/Counter 1 Mode Register

T/C1

FA7H, FA6H

Bit	7	6	5	4	3	2	1	0
Identifier	"0"	.6	.5	.4	.3	.2	"0"	"0"
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	1/8	8	8	8

TMOD1.7**Bit 7**

0	Always logic zero
---	-------------------

TMOD1.6-.4**Timer/Counter1 Input Clock Selection Bits ^{(1) (2)}**

0	1	x	
0	1	x	fxt(Subsystem clock: 32.768 kHz)
1	0	0	fxx /2 ¹⁰ (4.09 kHz)
1	0	1	fxx/2 ⁶ (65.5 kHz)
1	1	0	fxx/2 ⁴ (262kHz)
1	1	1	fxx (4.19 MHz)

TMOD1.3**Clear Counter And Resume Counting Control Bit**

1	Clear TCNT1 and IRQT1 and resume counting immediately.(This bit is cleared automatically when counting starts)
---	--

TMOD1.2**Enable/Disable Timer/Counter1 Bit**

0	Disable timer/counter1 ; retain TCNT1 contents
1	Enable timer/counter1

TMOD1.1**Bit 1**

0	Always logic zero
---	-------------------

TMOD1.0**Bit 0**

0	Always logic zero
---	-------------------

NOTES:

- 'fxx' is the system clock frequency (assume that fxx is 4.19MHz).
- 'x' is don't care.

TOE0 — Timer/Output Enable Flag Register

T/C0

F92H

Bit	3	2	1	0
Identifier	"0"	TOE0	"0"	"0"
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3	Bit 3	
	0	Always logic zero

TOE0	Clear Counter And Resume Counting Control Bit	
	0	Disable timer/counter 0 clock output at the TCLO0 pin
	1	Enable timer/counter 0 clock output at the TCLO0 pin

.1-0	Bit 1-0	
	0	Always logic zero

WDFLAG — Watch-Dog Timer's Counter Clear Flag

BT

F9AH.3

Bit	3	2	1	0
Identifier	WDTCF	"0"	"0"	"0"
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	1/4	1/4	1/4

WDTCF**Watch-dog Timer's Counter Clear Bit**

0	—
1	Clear the WDT's counter to zero and restart the WDT's counter

WDFLAG.2-0**Bit 2-0**

0	Always logic zero
---	-------------------

WDMOD — Watch-Dog Timer Mode Control Register **BT** **F99H, F98H**

Bit	7	6	5	4	3	2	1	0
Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	0	1	0	0	1	0	1
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

WDMOD.7-.0**Watch-Dog Timer Enable/Disable Control**

0	1	0	1	1	0	1	0	Disable watch-dog timer function
Other Values								Enable watch-dog timer function

WMOD — Watch Timer Mode Register**WT****F89H, F88H**

Bit	7	6	5	4	3	2	1	0
Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	(note)	0	0	0
Read/Write	W	W	W	W	R	W	W	W
Bit Addressing	8	8	8	8	1	8	8	8

WMOD.7**Enable/Disable Buzzer Output Bit**

0	Disable buzzer (BUZ) signal output at the BUZ pin
1	Enable buzzer (BUZ) signal output at the BUZ pin

WMOD.6**Bit 6**

0	Always logic zero
---	-------------------

WMOD.5-.4**Output Buzzer Frequency Selection Bits**

0	0	2 kHz buzzer (BUZ) signal output
0	1	4 kHz buzzer (BUZ) signal output
1	0	8 kHz buzzer (BUZ) signal output
1	1	16 kHz buzzer (BUZ) signal output

WMOD.3**XT_{IN} Input Level Control Bit**

0	Input level to XT _{IN} pin is low; 1-bit read-only addressable for tests
1	Input level to XT _{IN} pin is high; 1-bit read-only addressable for tests

WMOD.2**Enable/Disable Watch Timer Bit**

0	Disable watch timer and clear frequency dividing circuits
1	Enable watch timer

WMOD.1**Watch Timer Speed Control Bit**

0	Normal speed: Set IRQW to 0.5 seconds
1	High-speed operation: Set IRQW to 3.91 ms

WMOD.0**Watch Timer Clock Selection Bit**

0	Select main system clock(fx)/128 as the watch timer clock Select main system clock (fx) as a LCD clock source.
1	Select a subsystem clock as the watch timer clock Select a subsystem clock as a LCD clock source.

NOTE: RESET sets WMOD.3 to the current input level of the subsystem clock, XTIN. If the input level is high, WMOD.3 is set to logic one; if low, WMOD.3 is cleared to zero along with all the other bits in the WMOD register

5

SAM48 INSTRUCTION SET

OVERVIEW

The SAM48 instruction set is specifically designed to support the large register files typically founded in most S3C7-series microcontrollers. The SAM48 instruction set includes 1-bit, 4-bit, and 8-bit instructions for data manipulation, logical and arithmetic operations, program control, and CPU control. I/O instructions for peripheral hardware devices are flexible and easy to use. Symbolic hardware names can be substituted as the instruction operand in place of the actual address. Other important features of the SAM48 instruction set include:

- 1-byte referencing of long instructions (REF instruction)
- Redundant instruction reduction (string effect)
- Skip feature for ADC and SBC instructions

Instruction operands conform to the operand format defined for each instruction. Several instructions have multiple operand formats.

Predefined values or labels can be used as instruction operands when addressing immediate data. Many of the symbols for specific registers and flags may also be substituted as labels for operations such DA, mema, memb, b, and so on. Using instruction labels can greatly simplify programming and debugging tasks.

INSTRUCTION SET FEATURES

In this section, the following SAM48 instruction set features are described in detail:

- Instruction reference area
- Instruction redundancy reduction
- Flexible bit manipulation
- ADC and SBC instruction skip condition

NOTES:

1. The ROM size accessed by instruction may change for different devices in the SAM48 product family (JP, JPS, CALL, and CALLS).
2. The number of memory bank selected by SMB may change for different devices in the SAM48 product family.
3. The port names used in the instruction set may change for different devices in the SAM48 product family.
4. The interrupt names and the interrupt numbers used in the instruction set may change for different devices in the SAM48 product family.

INSTRUCTION REFERENCE AREA

Using the 1-byte REF (Reference) instruction, you can reference instructions stored in addresses 0020H-007FH of program memory (the REF instruction look-up table). The location referenced by REF may contain either two 1-byte instructions or a single 2-byte instruction. The starting address of the instruction being referenced must always be an even number.

3-byte instructions such as JP or CALL may also be referenced using REF. To reference these 3-byte instructions, the 2-byte pseudo commands TJP and TCALL must be written in the reference.

The PC is not incremented when a REF instruction is executed. After it executes, the program's instruction execution sequence resumes at the address immediately following the REF instruction. By using REF instructions to execute instructions larger than one byte, as well as branches and subroutines, you can reduce the program size. To summarize, the REF instruction can be used in three ways:

- Using the 1-byte REF instruction to execute one 2-byte or two 1-byte instructions;
- Branching to any location by referencing a branch address that is stored in the look-up table;
- Calling subroutines at any location by referencing a call address that is stored in the look-up table.

If necessary, a REF instruction can be circumvented by means of a skip operation prior to the REF in the execution sequence. In addition, the instruction immediately following a REF can also be skipped by using an appropriate reference instruction or instructions.

Two-byte instruction can be referenced by using a REF instruction (An exception is XCH A, DA). If the MSB value of the first one-byte instruction in the reference area is "0", the instruction cannot be referenced by a REF instruction. Therefore, if you use REF to reference two 1-byte instruction stored in the reference area, specific combinations must be used for the first and second 1-byte instruction. These combination examples are described in Table 5-1.

Table 5-1. Valid 1-Byte Instruction Combinations for REF Look-Ups

First 1-Byte Instruction		Second 1-Byte Instruction	
Instruction	Operand	Instruction	Operand
LD	A, #im	INCS <small>(note)</small> INCS DECS <small>(note)</small>	R RRb R
LD	A, @RRa	INCS <small>(note)</small> INCS DECS <small>(note)</small>	R RRb R
LD	@HL, A	INCS <small>(note)</small> INCS DECS <small>(note)</small>	R RRb R

NOTE: The MSB value of the instruction is "0".

REDUCING INSTRUCTION REDUNDANCY

When redundant instructions such as LD A,#im and LD EA,#imm are used consecutively in a program sequence, only the first instruction is executed. The redundant instructions which follow are ignored, that is, they are handled like a NOP instruction. When LD HL,#imm instructions are used consecutively, redundant instructions are also ignored.

In the following example, only the 'LD A, #im' instruction will be executed. The 8-bit load instruction which follows it is interpreted as redundant and is ignored:

```
LD      A,#im      ; Load 4-bit immediate data (#im) to accumulator
LD      EA,#imm    ; Load 8-bit immediate data (#imm) to extended accumulator
```

In this example, the statements 'LD A,#2H' and 'LD A,#3H' are ignored:

```
BITR    EMB
LD      A,#1H      ; Execute instruction
LD      A,#2H      ; Ignore, redundant instruction
LD      A,#3H      ; Ignore, redundant instruction
LD      23H,A      ; Execute instruction, 023H ← #1H
```

If consecutive LD HL, #imm instructions (load 8-bit immediate data to the 8-bit memory pointer pair, HL) are detected, only the first LD is executed and the LDs which immediately follow are ignored. For example,

```
LD      HL,#10H    ; HL ← 10H
LD      HL,#20H    ; Ignore, redundant instruction
LD      A,#3H      ; A ← 3H
LD      EA,#35H    ; Ignore, redundant instruction
LD      @HL,A      ; (10H) ← 3H
```

If an instruction reference with a REF instruction has a redundancy effect, the following conditions apply:

- If the instruction *preceding* the REF has a redundancy effect, this effect is cancelled and the referenced instruction is not skipped.
- If the instruction *following* the REF has a redundancy effect, the instruction following the REF is skipped.

PROGRAMMING TIP — Example of the Instruction Redundancy Effect

```
ABC      ORG      0020H
LD      EA,#30H    ; Stored in REF instruction reference area
ORG      0080H
.
.
.
LD      EA,#40H    ; Redundancy effect is encountered
REF     ABC        ; No skip (EA ← #30H)
.
.
.
REF     ABC        ; EA ← #30H
LD      EA,#50H    ; Skip
```

FLEXIBLE BIT MANIPULATION

In addition to normal bit manipulation instructions like set and clear, the SAM48 instruction set can also perform bit tests, bit transfers, and bit Boolean operations. Bits can also be addressed and manipulated by special bit addressing modes. Three types of bit addressing are supported:

- mema.b
- memb.@L
- @H+DA.b

The parameters of these bit addressing modes are described in more detail in Table 5-2.

Table 5-2. Bit Addressing Modes and Parameters

Addressing Mode	Addressable Peripherals	Address Range
mema.b	ERB, EMB, IS1, IS0, IEx, IRQx	FB0H-FBFH
	Ports	FF0H-FFFH
memb.@L	Ports, and BSC	FC0H-FFFH
@H+DA.b	All bit-manipulatable peripheral hardware	All bits of the memory bank specified by EMB and SMB that are bit-manipulatable

NOTE: Some device in the SAM48 product family don't have BSC.

INSTRUCTIONS WHICH HAVE SKIP CONDITIONS

The following instructions have a skip function when an overflow or borrow occurs:

XCHI	INCS
XCHD	DECS
LDI	ADS
LDD	SBS

If there is an overflow or borrow from the result of an increment or decrement, a skip signal is generated and a skip is executed. However, the carry flag value is unaffected.

The instructions BTST, BTSF, and CPSE also generate a skip signal and execute a skip when they meet a skip condition, and the carry flag value is also unaffected.

INSTRUCTIONS WHICH AFFECT THE CARRY FLAG

The only instructions which do not generate a skip signal, but which do affect the carry flag are as follows:

ADC	LDB	C,(operand)
SBC	BAND	C,(operand)
SCF	BOR	C,(operand)
RCF	BXOR	C,(operand)
CCF	IRET	
RRC		

ADC AND SBC INSTRUCTION SKIP CONDITIONS

The instructions 'ADC A,@HL' and 'SBC A,@HL' can generate a skip signal, and set or clear the carry flag, when they are executed in combination with the instruction 'ADS A,#im'.

If an 'ADS A,#im' instruction immediately follows an 'ADC A,@HL' or 'SBC A,@HL' instruction in a program sequence, the ADS instruction does not skip the instruction following it, even if it has a skip function. If, however, an 'ADC A,@HL' or 'SBC A,@HL' instruction is immediately followed by an 'ADS A,#im' instruction, the ADC (or SBC) skips on overflow (or if there is no borrow) to the instruction immediately following the ADS, and program execution continues. Table 5-3 contains additional information and examples of the 'ADC A,@HL' and 'SBC A,@HL' skip feature.

Table 5-3. Skip Conditions for ADC and SBC Instructions

Sample Instruction Sequences		If the result of instruction 1 is:	Then, the execution sequence is:	Reason
ADC A,@HL	1	Overflow	1, 3, 4	ADS cannot skip instruction 3, even if it has a skip function.
ADS A,#im	2	No overflow	1, 2, 3, 4	
xxx	3			
xxx	4			
SBC A,@HL	1	Borrow	1, 2, 3, 4	ADS cannot skip instruction 3, even if it has a skip function.
ADS A,#im	2	No borrow	1, 3, 4	
xxx	3			
xxx	4			

SYMBOLS AND CONVENTIONS

Table 5-4. Data Type Symbols

Symbol	Data Type
d	Immediate data
a	Address data
b	Bit data
r	Register data
f	Flag data
i	Indirect addressing data
t	memc × 0.5 immediate data

Table 5-5. Register Identifiers

Full Register Name	ID
4-bit accumulator	A
4-bit working registers	E, L, H, X, W, Z, Y
8-bit extended accumulator	EA
8-bit memory pointer	HL
8-bit working registers	WX, YZ, WL
Select register bank 'n'	SRB n
Select memory bank 'n'	SMB n
Carry flag	C
Program status word	PSW
Port 'n'	Pn
'm'-th bit of port 'n'	Pn.m
Interrupt priority register	IPR
Enable memory bank flag	EMB
Enable register bank flag	ERB

Table 5-6. Instruction Operand Notation

Symbol	Definition
DA	Direct address
@	Indirect address prefix
src	Source operand
dst	Destination operand
(R)	Contents of register R
.b	Bit location
im	4-bit immediate data (number)
imm	8-bit immediate data (number)
#	Immediate data prefix
ADR	000H-3FFFH immediate address
ADRn	'n' bit address
R	A, E, L, H, X, W, Z, Y
Ra	E, L, H, X, W, Z, Y
RR	EA, HL, WX, YZ
RRa	HL, WX, WL
RRb	HL, WX, YZ
RRc	WX, WL
mema	FB0H-FBFH, FF0H-FFFH
memb	FC0H-FFFH
memc	Code direct addressing: 0020H-007FH
SB	Select bank register (8 bits)
XOR	Logical exclusive-OR
OR	Logical OR
AND	Logical AND
[(RR)]	Contents addressed by RR

OPCODE DEFINITIONS

Table 5-7. Opcode Definitions (Direct)

Register	r2	r1	r0
A	0	0	0
E	0	0	1
L	0	1	0
H	0	1	1
X	1	0	0
W	1	0	1
Z	1	1	0
Y	1	1	1
EA	0	0	0
HL	0	1	0
WX	1	0	0
YZ	1	1	0

r = Immediate data for register

Table 5-8. Opcode Definitions (Indirect)

Register	i2	i1	i0
@HL	1	0	1
@WX	1	1	0
@WL	1	1	1

i = Immediate data for indirect addressing

CALCULATING ADDITIONAL MACHINE CYCLES FOR SKIPS

A machine cycle is defined as one cycle of the selected CPU clock. Three different clock rates can be selected using the PCON register.

In this document, the letter 'S' is used in tables when describing the number of additional machine cycles required for an instruction to execute, given that the instruction has a skip function ('S' = skip). The addition number of machine cycles that will be required to perform the skip usually depends on the size of the instruction being skipped — whether it is a 1-byte, 2-byte, or 3-byte instruction. A skip is also executed for SMB and SRB instructions.

The values in additional machine cycles for 'S' for the three cases in which skip conditions occur are as follows:

- | | |
|--|--------------|
| Case 1: No skip | S = 0 cycles |
| Case 2: Skip is 1-byte or 2-byte instruction | S = 1 cycle |
| Case 3: Skip is 3-byte instruction | S = 2 cycles |

NOTE: REF instructions are skipped in one machine cycle.

HIGH-LEVEL SUMMARY

This section contains a high-level summary of the SAM48 instruction set in table format. The tables are designed to familiarize you with the range of instructions that are available in each instruction category.

These tables are a useful quick-reference resource when writing application programs.

If you are reading this user's manual for the first time, however, you may want to scan this detailed information briefly, and then return to it later on. The following information is provided for each instruction:

- Instruction name
- Operand(s)
- Brief operation description
- Number of bytes of the instruction and operand(s)
- Number of machine cycles required to execute the instruction

The tables in this section are arranged according to the following instruction categories:

- CPU control instructions
- Program control instructions
- Data transfer instructions
- Logic instructions
- Arithmetic instructions
- Bit manipulation instructions

Table 5-9. CPU Control Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
SCF	–	Set carry flag to logic one	1	1
RCF		Reset carry flag to logic zero	1	1
CCF		Complement carry flag	1	1
EI		Enable all interrupts	2	2
DI		Disable all interrupts	2	2
IDLE		Engage CPU idle mode	2	2
STOP		Engage CPU stop mode	2	2
NOP		No operation	1	1
SMB		n	Select memory bank	2
SRB	n	Select register bank	2	2
REF	memc	Reference code	1	1
VENTn	EMB (0,1) ERB (0,1) ADR	Load enable memory bank flag (EMB) and the enable register bank flag (ERB) and program counter to vector address, then branch to the corresponding location	2	2

Table 5-10. Program Control Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
CPSE	R,#im	Compare and skip if register equals #im	2	2 + S
	@HL,#im	Compare and skip if indirect data memory equals #im	2	2 + S
	A,R	Compare and skip if A equals R	2	2 + S
	A,@HL	Compare and skip if A equals indirect data memory	1	1 + S
	EA,@HL	Compare and skip if EA equals indirect data memory	2	2 + S
	EA,RR	Compare and skip if EA equals RR	2	2 + S
JP	ADR	Jump to direct address (14 bits)	3	3
JPS	ADR	Jump direct in page (12 bits)	2	2
JR	#im	Jump to immediate address	1	2
	@WX	Branch relative to WX register	2	3
	@EA	Branch relative to EA	2	3
CALL	ADR	Call direct in page (14 bits)	3	4
CALLS	ADR	Call direct in page (11 bits)	2	3
RET	–	Return from subroutine	1	3
IRET	–	Return from interrupt	1	3
SRET	–	Return from subroutine and skip	1	3 + S

Table 5-11. Data Transfer Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
XCH	A,DA	Exchange A and direct data memory contents	2	2
	A,Ra	Exchange A and register (Ra) contents	1	1
	A,@RRa	Exchange A and indirect data memory	1	1
	EA,DA	Exchange EA and direct data memory contents	2	2
	EA,RRb	Exchange EA and register pair (RRb) contents	2	2
	EA,@HL	Exchange EA and indirect data memory contents	2	2
XCHI	A,@HL	Exchange A and indirect data memory contents; increment contents of register L and skip on carry	1	2 + S
XCHD	A,@HL	Exchange A and indirect data memory contents; decrement contents of register L and skip on carry	1	2 + S
LD	A,#im	Load 4-bit immediate data to A	1	1
	A,@RRa	Load indirect data memory contents to A	1	1
	A,DA	Load direct data memory contents to A	2	2
	A,Ra	Load register contents to A	2	2
	Ra,#im	Load 4-bit immediate data to register	2	2
	RR,#imm	Load 8-bit immediate data to register	2	2
	DA,A	Load contents of A to direct data memory	2	2
	Ra,A	Load contents of A to register	2	2
	EA,@HL	Load indirect data memory contents to EA	2	2
	EA,DA	Load direct data memory contents to EA	2	2
	EA,RRb	Load register contents to EA	2	2
	@HL,A	Load contents of A to indirect data memory	1	1
	DA,EA	Load contents of EA to data memory	2	2
	RRb,EA	Load contents of EA to register	2	2
@HL,EA	Load contents of EA to indirect data memory	2	2	
LDI	A,@HL	Load indirect data memory to A; increment register L contents and skip on carry	1	2 + S
LDD	A,@HL	Load indirect data memory contents to A; decrement register L contents and skip on carry	1	2 + S
LDC	EA,@WX	Load code byte from WX to EA	1	3
	EA,@EA	Load code byte from EA to EA	1	3
RRC	A	Rotate right through carry bit	1	1
PUSH	RR	Push register pair onto stack	1	1
	SB	Push SMB and SRB values onto stack	2	2
POP	RR	Pop to register pair from stack	1	1
	SB	Pop SMB and SRB values from stack	2	2

Table 5-12. Logic Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
AND	A,#im	Logical-AND A immediate data to A	2	2
	A,@HL	Logical-AND A indirect data memory to A	1	1
	EA,RR	Logical-AND register pair (RR) to EA	2	2
	RRb,EA	Logical-AND EA to register pair (RRb)	2	2
OR	A, #im	Logical-OR immediate data to A	2	2
	A, @HL	Logical-OR indirect data memory contents to A	1	1
	EA,RR	Logical-OR double register to EA	2	2
	RRb,EA	Logical-OR EA to double register	2	2
XOR	A,#im	Exclusive-OR immediate data to A	2	2
	A,@HL	Exclusive-OR indirect data memory to A	1	1
	EA,RR	Exclusive-OR register pair (RR) to EA	2	2
	RRb,EA	Exclusive-OR register pair (RRb) to EA	2	2
COM	A	Complement accumulator (A)	2	2

Table 5-13. Arithmetic Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
ADC	A,@HL	Add indirect data memory to A with carry	1	1
	EA,RR	Add register pair (RR) to EA with carry	2	2
	RRb,EA	Add EA to register pair (RRb) with carry	2	2
ADS	A, #im	Add 4-bit immediate data to A and skip on carry	1	1 + S
	EA,#imm	Add 8-bit immediate data to EA and skip on carry	2	2 + S
	A,@HL	Add indirect data memory to A and skip on carry	1	1 + S
	EA,RR	Add register pair (RR) contents to EA and skip on carry	2	2 + S
	RRb,EA	Add EA to register pair (RRb) and skip on carry	2	2 + S
SBC	A,@HL	Subtract indirect data memory from A with carry	1	1
	EA,RR	Subtract register pair (RR) from EA with carry	2	2
	RRb,EA	Subtract EA from register pair (RRb) with carry	2	2
SBS	A,@HL	Subtract indirect data memory from A; skip on borrow	1	1 + S
	EA,RR	Subtract register pair (RR) from EA; skip on borrow	2	2 + S
	RRb,EA	Subtract EA from register pair (RRb); skip on borrow	2	2 + S
DECS	R	Decrement register (R); skip on borrow	1	1 + S
	RR	Decrement register pair (RR); skip on borrow	2	2 + S
INCS	R	Increment register (R); skip on carry	1	1 + S
	DA	Increment direct data memory; skip on carry	2	2 + S
	@HL	Increment indirect data memory; skip on carry	2	2 + S
	RRb	Increment register pair (RRb); skip on carry	1	1 + S

Table 5-14. Bit Manipulation Instructions — High-Level Summary

Name	Operand	Operation Description	Bytes	Cycles
BTST	C	Test specified bit and skip if carry flag is set	1	1 + S
	DA.b	Test specified bit and skip if memory bit is set	2	2 + S
	mema.b			
	memb.@L			
@H+DA.b				
BTSF	DA.b	Test specified memory bit and skip if bit equals "0"		
	mema.b			
	memb.@L			
	@H+DA.b			
BTSTZ	mema.b	Test specified bit; skip and clear if memory bit is set		
	memb.@L			
	@H+DA.b			
BITS	DA.b	Set specified memory bit	2	2
	mema.b			
	memb.@L			
	@H+DA.b			
BITR	DA.b	Clear specified memory bit to logic zero		
	mema.b			
	memb.@L			
	@H+DA.b			
BAND	C,mema.b	Logical-AND carry flag with specified memory bit		
	C,memb.@L			
	C,@H+DA.b			
BOR	C,mema.b	Logical-OR carry with specified memory bit		
	C,memb.@L			
	C,@H+DA.b			
BXOR	C,mema.b	Exclusive-OR carry with specified memory bit		
	C,memb.@L			
	C,@H+DA.b			
LDB	mema.b,C	Load carry bit to a specified memory bit		
	memb.@L,C	Load carry bit to a specified indirect memory bit		
	@H+DA.b,C			
	C,mema.b	Load specified memory bit to carry bit		
	C,memb.@L	Load specified indirect memory bit to carry bit		
	C,@H+DA.b			

BINARY CODE SUMMARY

This section contains binary code values and operation notation for each instruction in the SAM48 instruction set in an easy-to-read, tabular format. It is intended to be used as a quick-reference source for programmers who are experienced with the SAM48 instruction set. The same binary values and notation are also included in the detailed descriptions of individual instructions later in Section 5.

If you are reading this user's manual for the first time, please just scan this very detailed information briefly. Most of the general information you will need to write application programs can be found in the high-level summary tables in the previous section. The following information is provided for each instruction:

- Instruction name
- Operand(s)
- Binary values
- Operation notation

The tables in this section are arranged according to the following instruction categories:

- CPU control instructions
- Program control instructions
- Data transfer instructions
- Logic instructions
- Arithmetic instructions
- Bit manipulation instructions

Table 5-15. CPU Control Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation	
SCF	-	1	1	1	0	0	1	1	1	$C \leftarrow 1$	
RCF		1	1	1	0	0	1	1	0	$C \leftarrow 0$	
CCF		1	1	0	1	0	1	1	0	$C \leftarrow C$	
EI		1	1	1	1	1	1	1	1	$IME \leftarrow 1$	
		1	0	1	1	0	0	1	0		
DI		1	1	1	1	1	1	1	0	$IME \leftarrow 0$	
		1	0	1	1	0	0	1	0		
IDLE		1	1	1	1	1	1	1	1	$PCON.2 \leftarrow 1$	
		1	0	1	0	0	0	1	1		
STOP		1	1	1	1	1	1	1	1	$PCON.3 \leftarrow 1$	
		1	0	1	1	0	0	1	1		
NOP			1	0	1	0	0	0	0	No operation	
SMB		n	1	1	0	1	1	1	0	1	$SMB \leftarrow n$
			0	1	0	0	d3	d2	d1	d0	
SRB	n	1	1	0	1	1	1	0	1	$SRB \leftarrow n$ (n = 0, 1, 2, 3)	
		0	1	0	1	0	0	d1	d0		
REF	memc	t7	t6	t5	t4	t3	t2	t1	t0	$PC13-0 \leftarrow memc.5-0 + (memc+1).7-0$	
VENTn	EMB (0,1) ERB (0,1) ADR	E	E	a13	a12	a11	a10	a9	a8	ROM (2 x n) 7-6 → EMB, ERB ROM (2 x n) 5-4 → PC12, PC13 ROM (2 x n) 3-0 → PC11-8 ROM (2 x n + 1) 7-0 → PC7-0 (n = 0, 1, 2, 3, 4, 5, 6, 7)	
		M	R								
		B	B								
		a7	a6	a5	a4	a3	a2	a1	a0		

Table 5-16. Program Control Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation
CPSE	R,#im	1	1	0	1	1	0	0	1	Skip if R = im
		d3	d2	d1	d0	0	r2	r1	r0	
	@HL,#im	1	1	0	1	1	1	0	1	Skip if (HL) = im
		0	1	1	1	d3	d2	d1	d0	
	A,R	1	1	0	1	1	1	0	1	Skip if A = R
		0	1	1	0	1	r2	r1	r0	
	A,@HL	0	0	1	1	1	0	0	0	Skip if A = (HL)
EA,@HL	1	1	0	1	1	1	0	0	Skip if A = (HL), E = (HL+1)	
	0	0	0	0	1	0	0	1		
EA,RR	1	1	0	1	1	1	0	0	Skip if EA = RR	
	1	1	1	0	1	r2	r1	0		
JP	ADR	1	1	0	1	1	0	1	1	PC13-0 ← ADR13-0
		0	0	a13	a12	a11	a10	a9	a8	
		a7	a6	a5	a4	a3	a2	a1	a0	
JPS	ADR	1	0	0	1	a11	a10	a9	a8	PC13-0 ← PC13-12 + ADR11-0
		a7	a6	a5	a4	a3	a2	a1	a0	
JR	#im *									PC13-0 ← ADR (PC-15 to PC+16)
	@WX	1	1	0	1	1	1	0	1	PC13-0 ← PC13-8 + (WX)
		0	1	1	0	0	1	0	0	
	@EA	1	1	0	1	1	1	0	1	PC13-0 ← PC13-8 + (EA)
0		1	1	0	0	0	0	0		
CALL	ADR	1	1	0	1	1	0	1	1	[(SP-1) (SP-2)] ← EMB, ERB
		0	1	a13	a12	a11	a10	a9	a8	[(SP-3) (SP-4)] ← PC7-0
		a7	a6	a5	a4	a3	a2	a1	a0	[(SP-5) (SP-6)] ← PC13-8
CALLS	ADR	1	1	1	0	1	a10	a9	a8	[(SP-1) (SP-2)] ← EMB, ERB
		a7	a6	a5	a4	a3	a2	a1	a0	[(SP-3) (SP-4)] ← PC7-0 [(SP-5) (SP-6)] ← PC14-8

	First Byte							Condition	
* JR #im	0	0	0	1	a3	a2	a1	a0	PC ← PC+2 to PC+16
	0	0	0	0	a3	a2	a1	a0	PC ← PC-1 to PC-15

Table 5-16. Program Control Instructions — Binary Code Summary (Continued)

Name	Operand	Binary Code								Operation Notation
RET	—	1	1	0	0	0	1	0	1	PC13-8 \leftarrow (SP + 1) (SP) PC7-0 \leftarrow (SP + 3) (SP + 2) EMB,ERB \leftarrow (SP + 4) SP \leftarrow SP + 6
IRET	—	1	1	0	1	0	1	0	1	PC13-8 \leftarrow (SP + 1) (SP) PC7-0 \leftarrow (SP + 3) (SP + 2) PSW \leftarrow (SP + 5) (SP + 4) SP \leftarrow SP + 6
SRET	—	1	1	1	0	0	1	0	1	PC13-8 \leftarrow (SP + 1) (SP) PC7-0 \leftarrow (SP + 3) (SP + 2) EMB,ERB \leftarrow (SP + 4) SP \leftarrow SP + 6

Table 5-17. Data Transfer Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation
XCH	A,DA	0	1	1	1	1	0	0	1	A \leftrightarrow DA
		a7	a6	a5	a4	a3	a2	a1	a0	
	A,Ra	0	1	1	0	1	r2	r1	r0	A \leftrightarrow Ra
	A,@RRa	0	1	1	1	1	i2	i1	i0	A \leftrightarrow (RRa)
	EA,DA	1	1	0	0	1	1	1	1	A \leftrightarrow DA, E \leftrightarrow DA + 1
		a7	a6	a5	a4	a3	a2	a1	a0	
	EA,RRb	1	1	0	1	1	1	0	0	EA \leftrightarrow RRb
1		1	1	0	0	r2	r1	0		
EA,@HL	1	1	0	1	1	1	0	0	A \leftrightarrow (HL), E \leftrightarrow (HL + 1)	
	0	0	0	0	0	0	0	1		
XCHI	A,@HL	0	1	1	1	1	0	1	0	A \leftrightarrow (HL), then L \leftarrow L+1; skip if L = 0H
XCHD	A,@HL	0	1	1	1	1	0	1	1	A \leftrightarrow (HL), then L \leftarrow L-1; skip if L = 0FH
LD	A,#im	1	0	1	1	d3	d2	d1	d0	A \leftarrow im
	A,@RRa	1	0	0	0	1	i2	i1	i0	A \leftarrow (RRa)
	A,DA	1	0	0	0	1	1	0	0	A \leftarrow DA
		a7	a6	a5	a4	a3	a2	a1	a0	
	A,Ra	1	1	0	1	1	1	0	1	A \leftarrow Ra
0		0	0	0	1	r2	r1	r0		

Table 5-17. Data Transfer Instructions — Binary Code Summary (Continued)

Name	Operand	Binary Code								Operation Notation
LD	Ra,#im	1	1	0	1	1	0	0	1	Ra ← im
		d3	d2	d1	d0	1	r2	r1	r0	
	RR,#imm	1	0	0	0	0	r2	r1	1	RR ← imm
		d7	d6	d5	d4	d3	d2	d1	d0	
	DA,A	1	0	0	0	1	0	0	1	DA ← A
		a7	a6	a5	a4	a3	a2	a1	a0	
	Ra,A	1	1	0	1	1	1	0	1	Ra ← A
		0	0	0	0	0	r2	r1	r0	
	EA,@HL	1	1	0	1	1	1	0	0	A ← (HL), E ← (HL + 1)
		0	0	0	0	1	0	0	0	
	EA,DA	1	1	0	0	1	1	1	0	A ← DA, E ← DA + 1
		a7	a6	a5	a4	a3	a2	a1	a0	
	EA,RRb	1	1	0	1	1	1	0	0	EA ← RRb
		1	1	1	1	1	r2	r1	0	
	@HL,A	1	1	0	0	0	1	0	0	(HL) ← A
	DA,EA	1	1	0	0	1	1	0	1	DA ← A, DA + 1 ← E
a7		a6	a5	a4	a3	a2	a1	a0		
RRb,EA	1	1	0	1	1	1	0	0	RRb ← EA	
	1	1	1	1	0	r2	r1	0		
@HL,EA	1	1	0	1	1	1	0	0	(HL) ← A, (HL + 1) ← E	
	0	0	0	0	0	0	0	0		
LDI	A,@HL	1	0	0	0	1	0	1	0	A ← (HL), then L ← L+1; skip if L = 0H
LDD	A,@HL	1	0	0	0	1	0	1	1	A ← (HL), then L ← L-1; skip if L = 0FH
LDC	EA,@WX	1	1	0	0	1	1	0	0	EA ← [PC13-8 + (WX)]
	EA,@EA	1	1	0	0	1	0	0	0	EA ← [PC13-8 + (EA)]
RRC	A	1	0	0	0	1	0	0	0	C ← A.0, A3 ← C A.n-1 ← A.n (n = 1, 2, 3)
PUSH	RR	0	0	1	0	1	r2	r1	1	((SP-1)) ((SP-2)) ← (RR), (SP) ← (SP)-2
	SB	1	1	0	1	1	1	0	1	((SP-1)) ← (SMB), ((SP-2)) ← (SRB), (SP) ← (SP)-2
		0	1	1	0	0	1	1	1	

Table 5-17. Data Transfer Instructions — Binary Code Summary (Concluded)

Name	Operand	Binary Code								Operation Notation
POP	RR	0	0	1	0	1	r2	r1	0	$RR_L \leftarrow (SP), RR_H \leftarrow (SP + 1)$ $SP \leftarrow SP + 2$
	SB	1	1	0	1	1	1	0	1	$(SRB) \leftarrow (SP), SMB \leftarrow (SP + 1),$ $SP \leftarrow SP + 2$
		0	1	1	0	0	1	1	0	

Table 5-18. Logic Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation
AND	A, #im	1	1	0	1	1	1	0	1	$A \leftarrow A \text{ AND } im$
		0	0	0	1	d3	d2	d1	d0	
	A, @HL	0	0	1	1	1	0	0	1	$A \leftarrow A \text{ AND } (HL)$
	EA, RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA \text{ AND } RR$
		0	0	0	1	1	r2	r1	0	
	RRb, EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb \text{ AND } EA$
0		0	0	1	0	r2	r1	0		
OR	A, #im	1	1	0	1	1	1	0	1	$A \leftarrow A \text{ OR } im$
		0	0	1	0	d3	d2	d1	d0	
	A, @HL	0	0	1	1	1	0	1	0	$A \leftarrow A \text{ OR } (HL)$
	EA, RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA \text{ OR } RR$
		0	0	1	0	1	r2	r1	0	
	RRb, EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb \text{ OR } EA$
0		0	1	0	0	r2	r1	0		
XOR	A, #im	1	1	0	1	1	1	0	1	$A \leftarrow A \text{ XOR } im$
		0	0	1	1	d3	d2	d1	d0	
	A, @HL	0	0	1	1	1	0	1	1	$A \leftarrow A \text{ XOR } (HL)$
	EA, RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA \text{ XOR } (RR)$
		0	0	1	1	0	r2	r1	0	
	RRb, EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb \text{ XOR } EA$
0		0	1	1	0	r2	r1	0		
COM	A	1	1	0	1	1	1	0	1	$A \leftarrow A$
		0	0	1	1	1	1	1	1	

Table 5-19. Arithmetic Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation
ADC	A,@HL	0	0	1	1	1	1	1	0	$C, A \leftarrow A + (HL) + C$
	EA,RR	1	1	0	1	1	1	0	0	$C, EA \leftarrow EA + RR + C$
		1	0	1	0	1	r2	r1	0	
	RRb,EA	1	1	0	1	1	1	0	0	$C, RRb \leftarrow RRb + EA + C$
		1	0	1	0	0	r2	r1	0	
ADS	A, #im	1	0	1	0	d3	d2	d1	d0	$A \leftarrow A + im$; skip on carry
	EA,#imm	1	1	0	0	1	0	0	1	$EA \leftarrow EA + imm$; skip on carry
		d7	d6	d5	d4	d3	d2	d1	d0	
	A,@HL	0	0	1	1	1	1	1	1	$A \leftarrow A + (HL)$; skip on carry
	EA,RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA + RR$; skip on carry
		1	0	0	1	1	r2	r1	0	
	RRb,EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb + EA$; skip on carry
1		0	0	1	0	r2	r1	0		
SBC	A,@HL	0	0	1	1	1	1	0	0	$C, A \leftarrow A - (HL) - C$
	EA,RR	1	1	0	1	1	1	0	0	$C, EA \leftarrow EA - RR - C$
		1	1	0	0	1	r2	r1	0	
	RRb,EA	1	1	0	1	1	1	0	0	$C, RRb \leftarrow RRb - EA - C$
		1	1	0	0	0	r2	r1	0	
SBS	A,@HL	0	0	1	1	1	1	0	1	$A \leftarrow A - (HL)$; skip on borrow
	EA,RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA - RR$; skip on borrow
		1	0	1	1	1	r2	r1	0	
	RRb,EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb - EA$; skip on borrow
		1	0	1	1	0	r2	r1	0	
DECS	R	0	1	0	0	1	r2	r1	r0	$R \leftarrow R - 1$; skip on borrow
	RR	1	1	0	1	1	1	0	0	$RR \leftarrow RR - 1$; skip on borrow
		1	1	0	1	1	r2	r1	0	
INCS	R	0	1	0	1	1	r2	r1	r0	$R \leftarrow R + 1$; skip on carry
	DA	1	1	0	0	1	0	1	0	$DA \leftarrow DA + 1$; skip on carry
		a7	a6	a5	a4	a3	a2	a1	a0	
	@HL	1	1	0	1	1	1	0	1	$(HL) \leftarrow (HL) + 1$; skip on carry
		0	1	1	0	0	0	1	0	
RRb	1	0	0	0	0	r2	r1	0	$RRb \leftarrow RRb + 1$; skip on carry	

Table 5-20. Bit Manipulation Instructions — Binary Code Summary

Name	Operand	Binary Code								Operation Notation
BTST	C	1	1	0	1	0	1	1	1	Skip if C = 1
	DA.b	1	1	b1	b0	0	0	1	1	Skip if DA.b = 1
		a7	a6	a5	a4	a3	a2	a1	a0	
	mema.b *	1	1	1	1	1	0	0	1	Skip if mema.b = 1
	memb.@L	1	1	1	1	1	0	0	1	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 1
0		1	0	0	a5	a4	a3	a2		
@H+DA.b	1	1	1	1	1	0	0	1	Skip if [H + DA.3-0].b = 1	
	0	0	b1	b0	a3	a2	a1	a0		
BTSF	DA.b	1	1	b1	b0	0	0	1	0	Skip if DA.b = 0
		a7	a6	a5	a4	a3	a2	a1	a0	
	mema.b *	1	1	1	1	1	0	0	0	Skip if mema.b = 0
	memb.@L	1	1	1	1	1	0	0	0	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 0
		0	1	0	0	a5	a4	a3	a2	
@H DA.b	1	1	1	1	1	0	0	0	Skip if [H + DA.3-0].b = 0	
	0	0	b1	b0	a3	a2	a1	a0		
BTSTZ	mema.b *	1	1	1	1	1	1	0	1	Skip if mema.b = 1 and clear
	memb.@L	1	1	1	1	1	1	0	1	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 1 and clear
		0	1	0	0	a5	a4	a3	a2	
	@H+DA.b	1	1	1	1	1	1	0	1	Skip if [H + DA.3-0].b = 1 and clear
		0	0	b1	b0	a3	a2	a1	a0	
BITS	DA.b	1	1	b1	b0	0	0	0	1	DA.b ← 1
		a7	a6	a5	a4	a3	a2	a1	a0	
	mema.b *	1	1	1	1	1	1	1	1	mema.b ← 1
	memb.@L	1	1	1	1	1	1	1	1	[memb.7-2 + L.3-2].[L.1-0] ← 1
		0	1	0	0	a5	a4	a3	a2	
@H+DA.b	1	1	1	1	1	1	1	1	[H + DA.3-0].b ← 1	
	0	0	b1	b0	a3	a2	a1	a0		

Table 5-20. Bit Manipulation Instructions — Binary Code Summary (Continued)

Name	Operand	Binary Code								Operation Notation	
BITR	DA.b	1	1	b1	b0	0	0	0	0	DA.b ← 0	
		a7	a6	a5	a4	a3	a2	a1	a0		
	mema.b *	1	1	1	1	1	1	1	0	mema.b ← 0	
	memb.@L		1	1	1	1	1	1	1	0	[memb.7-2 + L3-2].[L.1-0] ← 0
			0	1	0	0	a5	a4	a3	a2	
@H+DA.b		1	1	1	1	1	1	1	0	[H + DA.3-0].b ← 0	
		0	0	b1	b0	a3	a2	a1	a0		
BAND	C,mema.b *	1	1	1	1	0	1	0	1	C ← C AND mema.b	
	C,memb.@L		1	1	1	1	0	1	0	1	C ← C AND [memb.7-2 + L.3-2]. [L.1-0]
			0	1	0	0	a5	a4	a3	a2	
	C,@H+DA.b		1	1	1	1	0	1	0	1	C ← C AND [H + DA.3-0].b
			0	0	b1	b0	a3	a2	a1	a0	
BOR	C,mema.b *	1	1	1	1	0	1	1	0	C ← C OR mema.b	
	C,memb.@L		1	1	1	1	0	1	1	0	C ← C OR [memb.7-2 + L.3-2]. [L.1-0]
			0	1	0	0	a5	a4	a3	a2	
	C,@H+DA.b		1	1	1	1	0	1	1	0	C ← C OR [H + DA.3-0].b
			0	0	b1	b0	a3	a2	a1	a0	
BXOR	C,mema.b *	1	1	1	1	0	1	1	1	C ← C XOR mema.b	
	C,memb.@L		1	1	1	1	0	1	1	1	C ← C XOR [memb.7-2 + L.3-2]. [L.1-0]
			0	1	0	0	a5	a4	a3	a2	
	C,@H+DA.b		1	1	1	1	0	1	1	1	C ← C XOR [H + DA.3-0].b
			0	0	b1	b0	a3	a2	a1	a0	

* mema.b	Second Byte								Bit Addresses
	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH	

Table 5-20. Bit Manipulation Instructions — Binary Code Summary (Concluded)

Name	Operand	Binary Code								Operation Notation
LDB	mema.b,C *	1	1	1	1	1	1	0	0	mema.b ← C
	memb.@L,C	1	1	1	1	1	1	0	0	memb.7-2 + [L.3-2]. [L.1-0] ← C
		0	1	0	0	a5	a4	a3	a2	
	@H+DA.b,C	1	1	1	1	1	1	0	0	H + [DA.3-0].b ← (C)
		0	0	b1	b0	a3	a2	a1	a0	
	C,mema.b *	1	1	1	1	0	1	0	0	C ← mema.b
	C,memb.@L	1	1	1	1	0	1	0	0	C ← memb.7-2 + [L.3-2] . [L.1-0]
		0	1	0	0	a5	a4	a3	a2	
	C,@H+DA.b	1	1	1	1	0	1	0	0	C ← [H + DA.3-0].b
		0	0	b1	b0	a3	a2	a1	a0	

* mema.b	Second Byte								Bit Addresses
	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
1	1	b1	b0	a3	a2	a1	a0	FFF0H-FFFH	

INSTRUCTION DESCRIPTIONS

This section contains detailed information and programming examples for each instruction of the SAM48 instruction set. Information is arranged in a consistent format to improve readability and for use as a quick-reference resource for application programmers.

If you are reading this user's manual for the first time, please just scan this very detailed information briefly in order to acquaint yourself with the basic features of the instruction set. The information elements of the instruction description format are as follows:

- Instruction name (mnemonic)
- Full instruction name
- Source/destination format of the instruction operand
- Operation overview (from the "High-Level Summary" table)
- Textual description of the instruction's effect
- Binary code overview (from the "Binary Code Summary" table)
- Programming example(s) to show how the instruction is used

ADC — ADD With Carry

ADC dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Add indirect data memory to A with carry	1	1
	EA,RR	Add register pair (RR) to EA with carry	2	2
	RRb,EA	Add EA to register pair (RRb) with carry	2	2

Description: The source operand, along with the setting of the carry flag, is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. If there is an overflow from the most significant bit of the result, the carry flag is set; otherwise, the carry flag is cleared.

If 'ADC A,@HL' is followed by an 'ADS A,#im' instruction in a program, ADC skips the ADS instruction if an overflow occurs. If there is no overflow, the ADS instruction is executed normally. (This condition is valid only for 'ADC A,@HL' instructions. If an overflow occurs following an 'ADS A,#im' instruction, the next instruction will not be skipped.)

Operand	Binary Code								Operation Notation
A,@HL	0	0	1	1	1	1	1	0	$C, A \leftarrow A + (HL) + C$
EA,RR	1	1	0	1	1	1	0	0	$C, EA \leftarrow EA + RR + C$
	1	0	1	0	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	$C, RRb \leftarrow RRb + EA + C$
	1	0	1	0	0	r2	r1	0	

Examples:

- The extended accumulator contains the value 0C3H, register pair HL the value 0AAH, and the carry flag is set to "1":

```
SCF                ; C ← "1"
ADC    EA,HL        ; EA ← 0C3H + 0AAH + 1H = 6EH, C ← "1"
JPS    XXX          ; Jump to XXX; no skip after ADC
```

- If the extended accumulator contains the value 0C3H, register pair HL the value 0AAH, and the carry flag is cleared to "0":

```
RCF                ; C ← "0"
ADC    EA,HL        ; EA ← 0C3H + 0AAH + 0H = 6DH, C ← "1"
JPS    XXX          ; Jump to XXX; no skip after ADC
```

ADC — Add with Carry

ADC (Continued)

Examples: 3. If ADC A,@HL is followed by an ADS A,#im, the ADC skips on carry to the instruction immediately after the ADS. An ADS instruction immediately after the ADC does not skip even if an overflow occurs. This function is useful for decimal adjustment operations.

a. 8 + 9 decimal addition (the contents of the address specified by the HL register is 9H):

```
RCF                ; C ← "0"
LD      A,#8H      ; A ← 8H
ADS     A,#6H      ; A ← 8H + 6H = 0EH
ADC     A,@HL      ; A ← 0EH + 9H + C(0) = 7H, C ← "1"
ADS     A,#0AH     ; Skip this instruction because C = "1" after ADC result
JPS     XXX
```

b. 3 + 4 decimal addition (the contents of the address specified by the HL register is 4H):

```
RCF                ; C ← "0"
LD      A,#3H      ; A ← 3H
ADS     A,#6H      ; A ← 3H + 6H = 9H
ADC     A,@HL      ; A ← 9H + 4H + C(0) = 0DH
ADS     A,#0AH     ; No skip. A ← 0DH + 0AH = 7H
                ; (The skip function for 'ADS A,#im' is inhibited after an
                ; 'ADC A,@HL' instruction even if an overflow occurs.)
JPS     XXX
```

ADS — Add and Skip on Overflow

ADS dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A, #im	Add 4-bit immediate data to A and skip on overflow	1	1 + S
	EA,#imm	Add 8-bit immediate data to EA and skip on overflow	2	2 + S
	A,@HL	Add indirect data memory to A and skip on overflow	1	1 + S
	EA,RR	Add register pair (RR) contents to EA and skip on overflow	2	2 + S
	RRb,EA	Add EA to register pair (RRb) and skip on overflow	2	2 + S

Description: The source operand is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. If there is an overflow from the most significant bit of the result, the skip signal is generated and a skip is executed, but the carry flag value is unaffected.

If 'ADS A,#im' follows an 'ADC A,@HL' instruction in a program, ADC skips the ADS instruction if an overflow occurs. If there is no overflow, the ADS instruction is executed normally. This skip condition is valid only for 'ADC A,@HL' instructions, however. If an overflow occurs following an ADS instruction, the next instruction is not skipped.

Operand	Binary Code								Operation Notation
A, #im	1	0	1	0	d3	d2	d1	d0	$A \leftarrow A + im$; skip on overflow
EA,#imm	1	1	0	0	1	0	0	1	$EA \leftarrow EA + imm$; skip on overflow
	d7	d6	d5	d4	d3	d2	d1	d0	
A,@HL	0	0	1	1	1	1	1	1	$A \leftarrow A + (HL)$; skip on overflow
EA,RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA + RR$; skip on overflow
	1	0	0	1	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb + EA$; skip on overflow
	1	0	0	1	0	r2	r1	0	

Examples: 1. The extended accumulator contains the value 0C3H, register pair HL the value 0AAH, and the carry flag = "0":

```

ADS     EA,HL           ; EA ← 0C3H + 0AAH = 6DH
                        ; ADS skips on overflow, but carry flag value is not affected.
JPS     XXX             ; This instruction is skipped since ADS had an overflow.
JPS     YYY             ; Jump to YYY.
```


ADS — Add and Skip on Overflow

ADS (Continued)

Examples: 2. If the extended accumulator contains the value 0C3H, register pair HL the value 12H, and the carry flag = "0":

```
ADS    EA,HL           ; EA ← 0C3H + 12H = 0D5H
JPS    XXX            ; Jump to XXX; no skip after ADS.
```

3. If 'ADC A,@HL' is followed by an 'ADS A,#im', the ADC skips on overflow to the instruction immediately after the ADS. An 'ADS A,#im' instruction immediately after the 'ADC A,@HL' does not skip even if overflow occurs. This function is useful for decimal adjustment operations.

a. 8 + 9 decimal addition (the contents of the address specified by the HL register is 9H):

```
RCF                    ; C ← "0"
LD      A,#8H         ; A ← 8H
ADS     A,#6H         ; A ← 8H + 6H = 0EH
ADC     A,@HL        ; A ← 0EH + 9H + C(0) = 7H, C ← "1"
ADS     A,#0AH       ; Skip this instruction because C = "1" after ADC result.
JPS     XXX
```

b. 3 + 4 decimal addition (the contents of the address specified by the HL register is 4H):

```
RCF                    ; C ← "0"
LD      A,#3H         ; A ← 3H
ADS     A,#6H         ; A ← 3H + 6H = 9H
ADC     A,@HL        ; A ← 9H + 4H + C(0) = 0DH, C ← "0"
ADS     A,#0AH       ; No skip. A ← 0DH + 0AH = 7H
                    ; (The skip function for 'ADS A,#im' is inhibited after an
                    ; 'ADC A,@HL' instruction even if an overflow occurs.)
JPS     XXX
```

AND — Logical AND

AND dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,#im	Logical-AND A immediate data to A	2	2
	A,@HL	Logical-AND A indirect data memory to A	1	1
	EA,RR	Logical-AND register pair (RR) to EA	2	2
	RRb,EA	Logical-AND EA to register pair (RRb)	2	2

Description: The source operand is logically ANDed with the destination operand. The result is stored in the destination. The logical AND operation results in a "1" whenever the corresponding bits in the two operands are both "1"; otherwise a "0" is stored in the corresponding destination bit. The contents of the source are unaffected.

Operand	Binary Code								Operation Notation
A,#im	1	1	0	1	1	1	0	1	A ← A AND im
	0	0	0	1	d3	d2	d1	d0	
A,@HL	0	0	1	1	1	0	0	1	A ← A AND (HL)
EA,RR	1	1	0	1	1	1	0	0	EA ← EA AND RR
	0	0	0	1	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	RRb ← RRb AND EA
	0	0	0	1	0	r2	r1	0	

Example: If the extended accumulator contains the value 0C3H (11000011B) and register pair HL the value 55H (01010101B), the instruction

```
AND EA,HL
```

leaves the value 41H (01000001B) in the extended accumulator EA .

BAND — Bit Logical AND

BAND C,src.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	C,mema.b	Logical-AND carry flag with memory bit	2	2
	C,memb.@L		2	2
	C,@H+DA.b		2	2

Description: The specified bit of the source is logically ANDed with the carry flag bit value. If the Boolean value of the source bit is a logic zero, the carry flag is cleared to "0"; otherwise, the current carry flag setting is left unaltered. The bit value of the source operand is not affected.

Operand	Binary Code								Operation Notation
C,mema.b *	1	1	1	1	0	1	0	1	C ← C AND mema.b
C,memb.@L	1	1	1	1	0	1	0	1	C ← C AND [memb.7-2 + L.3-2]. [L.1-0]
	0	1	0	0	a5	a4	a3	a2	
C,@H+DA.b	1	1	1	1	0	1	0	1	C ← C AND [H + DA.3-0].b
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. The following instructions set the carry flag if P1.0 (port 1.0) is equal to "1" (and assuming the carry flag is already set to "1"):

```
SMB      15                ; C ← "1"
BAND     C,P1.0           ; If P1.0 = "1", C ← "1"
                          ; If P1.0 = "0", C ← "0"
```

2. Assume the P1 address is FF1H and the value for register L is 5H (0101B). The address (memb.7-2) is 111100B; (L.3-2) is 01B. The resulting address is 11110001B or FF1H, specifying P1. The bit value for the BAND instruction, (L.1-0) is 01B which specifies bit 1. Therefore, P1.@L = P1.1:

```
LD       L,#5H
BAND     C,P1.@L         ; P1.@L is specified as P1.1
                          ; C AND P1.1
```

BAND — Bit Logical AND

BAND (Continued)

Examples: 3. Register H contains the value 2H and FLAG = 20H.3. The address of H is 0010B and FLAG(3-0) is 0000B. The resulting address is 00100000B or 20H. The bit value for the BAND instruction is 3. Therefore, @H+FLAG = 20H.3:

```
FLAG EQU 20H.3
LD H,#2H
BAND C,@H+FLAG ; C AND FLAG (20H.3)
```

BITR — Bit Reset

BITR dst.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	DA.b	Clear specified memory bit to logic zero	2	2
	mema.b		2	2
	memb.@L		2	2
	@H+DA.b		2	2

Description: A BITR instruction clears to logic zero (resets) the specified bit within the destination operand. No other bits in the destination are affected.

Operand	Binary Code								Operation Notation
DA.b	1	1	b1	b0	0	0	0	0	DA.b ← 0
	a7	a6	a5	a4	a3	a2	a1	a0	
mema.b *	1	1	1	1	1	1	1	0	mema.b ← 0
memb.@L	1	1	1	1	1	1	1	0	[memb.7-2 + L3-2].[L.1-0] ← 0
	0	1	0	0	a5	a4	a3	a2	
@H+DA.b	1	1	1	1	1	1	1	0	[H + DA.3-0].b ← 0
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples:

- If the bit location 30H.2 in the RAM has a current value of "1". The following instruction clears the third bit of location 30H to "0":

```
BITR       30H.2                   ; 30H.2 ← "0"
```

- You can use BITR in the same way to manipulate a port address bit:

```
BITR       P0.0                   ; P0.0 ← "0"
```

BITR — Bit Reset

BITR (Continued)

Examples: 3. For clearing P0.2, P0.3, and P1.0-P1.3 to "0":

```

BP2      LD      L,#2H
          BITR   P0.@L ; First, P0.@2H = P0.2
                               ; (111100B) + 00B.10B = 0F0H.2
          INCS   L
          CPSE   L,#8H
          JR     BP2
  
```

4. If bank 0, location 0A0H.0 is cleared (and regardless of whether the EMB value is logic zero), BITR has the following effect:

```

FLAG     EQU     0A0H.0
          •
          •
          •
          BITR   EMB
          •
          •
          •
          LD     H,#0AH
          BITR   @H+FLAG; Bank 0 (AH + 0H).0 = 0A0H.0 ← "0"
  
```

NOTE: Since the BITR instruction is used for output functions, the pin names used in the examples above may change for different devices in the SAM48 product family.

BITS — Bit Set

BITS dst.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	DA.b	Set specified memory bit	2	2
	mema.b		2	2
	memb.@L		2	2
	@H+DA.b		2	2

Description: This instruction sets the specified bit within the destination without affecting any other bits in the destination. BITS can manipulate any bit that is addressable using direct or indirect addressing modes.

Operand	Binary Code								Operation Notation
DA.b	1	1	b1	b0	0	0	0	1	DA.b ← 1
	a7	a6	a5	a4	a3	a2	a1	a0	
mema.b *	1	1	1	1	1	1	1	1	mema.b ← 1
memb.@L	1	1	1	1	1	1	1	1	[memb.7-2 + L.3-2].[L.1-0] ← 1
	0	1	0	0	a5	a4	a3	a2	
@H+DA.b	1	1	1	1	1	1	1	1	[H + DA.3-0] ← 1
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples:

- If the bit location 30H.2 in the RAM has a current value of "0", the following instruction sets the second bit of location 30H to "1".

BITS 30H.2 ; 30H.2 ← "1"

- You can use BITS in the same way to manipulate a port address bit:

BITS P0.0 ; P0.0 ← "1"

BITS — Bit Set

BITS (Continued)

Examples: 3. For setting P0.2, P0.3, and P1.0-P1.3 to "1":

```

BP2 LD      L,#2H
    BITS    P0.@L ; First, P0.@2H = P0.2
                        ; (111100B) + 00B.10B = 0F0H.2
    INCS    L
    CPSE    L,#8H
    JR      BP2
  
```

4. If bank 0, location 0A0H.0, is set to "1" and the EMB = "0", BITS has the following effect:

```

FLAG EQU      0A0H.0
    .
    .
    .
    BITR      EMB
    .
    .
    .
    LD        H,#0AH
    BITS      @H+FLAG; Bank 0 (AH + 0H).0 = 0A0H.0 ← "1"
  
```

NOTE: Since the BITS instruction is used for output functions, pin names used in the examples above may change for different devices in the SAM48 product family.

BOR — Bit Logical OR

BOR C,src.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	C,mema.b	Logical-OR carry with specified memory bit	2	2
	C,memb.@L		2	2
	C,@H+DA.b		2	2

Description: The specified bit of the source is logically ORed with the carry flag bit value. The value of the source is unaffected.

Operand	Binary Code								Operation Notation
C,mema.b *	1	1	1	1	0	1	1	0	C ← C OR mema.b
C,memb.@L	1	1	1	1	0	1	1	0	C ← C OR [memb.7-2 + L.3-2]. [L.1-0]
	0	1	0	0	a5	a4	a3	a2	
C,@H+DA.b	1	1	1	1	0	1	1	0	C ← C OR [H + DA.3-0].b
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. The carry flag is logically ORed with the P1.0 value:

```
RCF          ; C ← "0"
BOR    C,P1.0 ; If P1.0 = "1", then C ← "1"; if P1.0 = "0", then C ← "0"
```

2. The P1 address is FF1H and register L contains the value 1H (0001B). The address (memb.7-2) is 111100B and (L.3-2) = 00B. The resulting address is 11110000B or FF0H, specifying P0. The bit value for the BOR instruction, (L.1-0) is 01B which specifies bit 1. Therefore, P1.@L = P0.1:

```
LD    L,#1H
BOR    C,P1.@L ; P1.@L is specified as P0.1; C OR P0.1
```

BOR — Bit Logical OR

BOR (Continued)

Examples: 3. Register H contains the value 2H and FLAG = 20H.3. The address of H is 0010B and FLAG(3-0) is 0000B. The resulting address is 00100000B or 20H. The bit value for the BOR instruction is 3. Therefore, @H+FLAG = 20H.3:

```
FLAG EQU 20H.3
LD H,#2H
BOR C,@H+FLAG ; C OR FLAG (20H.3)
```

BTSF — Bit Test and Skip on False

BTSF dst.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	DA.b	Test specified memory bit and skip if bit equals "0"	2	2 + S
	mema.b		2	2 + S
	memb.@L		2	2 + S
	@H+DA.b		2	2 + S

Description: The specified bit within the destination operand is tested. If it is a "0", the BTSF instruction skips the instruction which immediately follows it; otherwise the instruction following the BTSF is executed. The destination bit value is not affected.

Operand	Binary Code								Operation Notation
DA.b	1	1	b1	b0	0	0	1	0	Skip if DA.b = 0
	a7	a6	a5	a4	a3	a2	a1	a0	
mema.b *	1	1	1	1	1	0	0	0	Skip if mema.b = 0
memb.@L	1	1	1	1	1	0	0	0	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 0
	0	1	0	0	a5	a4	a3	a2	
@H + DA.b	1	1	1	1	1	0	0	0	Skip if [H + DA.3-0].b = 0
	0	0	b1	b0	a3	a2	a1	a0	

		Second Byte							Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. If RAM bit location 30H.2 is set to "0", the following instruction sequence will cause the program to continue execution from the instruction identified as LABEL2:

```

BTSF    30H.2           ; If 30H.2 = "0", then skip
RET     ; If 30H.2 = "1", return
JP     LABEL2

```

2. You can use BTSF in the same way to test a port pin address bit:

```

BTSF    P1.0           ; If P1.0 = "0", then skip
RET     ; If P1.0 = "1", then return
JP     LABEL3

```

BTSF — Bit Test and Skip on False

BTSF (Continued)

Examples: 3. P0.2, P0.3 and P1.0-P1.3 are tested:

```

BP2      LD      L,#2H
         BTSF   P0.@L ; First, P1.@2H = P0.2
                               ; (111100B) + 00B.10B = 0F0H.2
         RET
         INCS   L
         CPSE   L,#8H
         JR     BP2

```

4. Bank 0, location 0A0H.0, is tested and (regardless of the current EMB value) BTSF has the following effect:

```

FLAG     EQU     0A0H.0
         •
         •
         •
         BITR   EMB
         •
         •
         LD     H,#0AH
         BTSF   @H+FLAG; If bank 0 (AH + 0H).0 = 0A0H.0 = "0", then skip
         RET
         •
         •
         •

```

BTST — Bit Test and Skip on True

BTST dst.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	C	Test carry bit and skip if set (= "1")	1	1 + S
	DA.b	Test specified bit and skip if memory bit is set	2	2 + S
	mema.b		2	2 + S
	memb.@L		2	2 + S
	@H+DA.b		2	2 + S

Description: The specified bit within the destination operand is tested. If it is "1", the instruction that immediately follows the BTST instruction is skipped; otherwise the instruction following the BTST instruction is executed. The destination bit value is not affected.

Operand	Binary Code								Operation Notation
C	1	1	0	1	0	1	1	1	Skip if C = 1
DA.b	1	1	b1	b0	0	0	1	1	Skip if DA.b = 1
	a7	a6	a5	a4	a3	a2	a1	a0	
mema.b *	1	1	1	1	1	0	0	1	Skip if mema.b = 1
memb.@L	1	1	1	1	1	0	0	1	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 1
	0	1	0	0	a5	a4	a3	a2	
@H+DA.b	1	1	1	1	1	0	0	1	Skip if [H + DA.3-0].b = 1
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. If RAM bit location 30H.2 is set to "0", the following instruction sequence will execute the RET instruction:

```

BTST    30H.2           ; If 30H.2 = "1", then skip
RET                                ; If 30H.2 = "0", return
JP      LABEL2

```

BTST — Bit Test and Skip on True

BTST (Continued)

Examples: 2. You can use BTST in the same way to test a port pin address bit:

```
BTST    P1.0           ; If P1.0 = "1", then skip
RET     ; If P1.0 = "0", then return
JP      LABEL3
```

3. P0.2, P0.3 and P1.0-P1.3 are tested :

```
LD      L,#2H
BP2     BTST    P0.@L ; First, P0.@2H = P0.2
                          ; (111100B) + 00B.10B = 0F0H.2
RET
INCS    L
CPSE    L,#8H
JR      BP2
```

4. Bank 0, location 0A0H.0, is tested and (regardless of the current EMB value) BTST has the following effect:

```
FLAG    EQU      0A0H.0
        .
        .
        .
        BITR    EMB
        .
        .
        .
        LD      H,#0AH
        BTST   @H+FLAG; If bank 0 (AH + 0H).0 = 0A0H.0 = "1", then skip
        RET
        .
        .
        .
```

BTSTZ — Bit Test and Skip on True; Clear Bit

BTSTZ dst.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	mema.b	Test specified bit; skip and clear if memory bit is set	2	2 + S
	memb.@L		2	2 + S
	@H+DA.b		2	2 + S

Description: The specified bit within the destination operand is tested. If it is a "1", the instruction immediately following the BTSTZ instruction is skipped; otherwise the instruction following the BTSTZ is executed. The destination bit value is cleared.

Operand	Binary Code								Operation Notation
mema.b *	1	1	1	1	1	1	0	1	Skip if mema.b = 1 and clear
memb.@L	1	1	1	1	1	1	0	1	Skip if [memb.7-2 + L.3-2]. [L.1-0] = 1 and clear
	0	1	0	0	a5	a4	a3	a2	
@H+DA.b	1	1	1	1	1	1	0	1	Skip if [H + DA.3-0].b = 1 and clear
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. Port pin P0.0 is toggled by checking the P0.0 value (level):

```
BTSTZ   P0.0           ; If P0.0 = "1", then P0.0 ← "0" and skip
BITS    P0.0           ; If P0.0 = "0", then P0.0 ← "1"
JP      LABEL3
```

2. For toggling P2.2, P2.3 and P3.0-P3.3:

```
LD      L,#0AH
BP2     BTSTZ   P2.@L   ; First, P2.@0AH = P2.2
                          ; (111100B) + 10B.10B = 0F2H.2

BITS    P2.@L
INCS    L
JR      BP2
```

BTSTZ — Bit Test and Skip on True; Clear Bit

BTSTZ (Continued)

Examples: 3. Bank 0, location 0A0H.0, is tested and EMB = "0":

```

FLAG      EQU      0A0H.0
          .
          .
          .
          BITR     EMB
          .
          .
          .
LD        H,#0AH
BTSTZ    @H+FLAG; If bank 0 (AH + 0H).0 = 0A0H.0 = "1", clear and skip
BITS     @H+FLAG; If 0A0H.0 = "0", then 0A0H.0 ← "1"

```


BXOR — Bit Exclusive OR

BXOR C,src.b

Operation:	Operand	Operation Summary	Bytes	Cycles
	C,mema.b	Exclusive-OR carry with memory bit	2	2
	C,memb.@L		2	2
	C,@H+DA.b		2	2

Description: The specified bit of the source is logically XORed with the carry bit value. The resultant bit is written to the carry flag. The source value is unaffected.

Operand	Binary Code								Operation Notation
C,mema.b *	1	1	1	1	0	1	1	1	C ← C XOR mema.b
C,memb.@L	1	1	1	1	0	1	1	1	C ← C XOR [memb.7-2 + L.3-2]. [L.1-0]
	0	1	0	0	a5	a4	a3	a2	
C,@H+DA.b	1	1	1	1	0	1	1	1	C ← C XOR [H + DA.3-0].b
	0	0	b1	b0	a3	a2	a1	a0	

	Second Byte								Bit Addresses
* mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
	1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

Examples: 1. The carry flag is logically XORed with the P1.0 value:

```
RCF          ; C ← "0"
BXOR    C,P1.0 ; If P1.0 = "1", then C ← "1"; if P1.0 = "0", then C ← "0"
```

2. The P1 address is FF1H and register L contains the value 1H (0001B). The address (memb.7-2) is 111100B and (L.3-2) = 00B. The resulting address is 11110000B or FF0H, specifying P0. The bit value for the BXOR instruction, (L.1-0) is 01B which specifies bit 1. Therefore, P1.@L = P0.1:

```
LD    L,#1H
BXOR  C,P0.@L ; P1.@L is specified as P0.1; C XOR P0.1
```

BXOR — Bit Exclusive OR

BXOR (Continued)

Examples: 3. Register H contains the value 2H and FLAG = 20H.3. The address of H is 0010B and FLAG(3-0) is 0000B. The resulting address is 00100000B or 20H. The bit value for the BOR instruction is 3. Therefore, @H+FLAG = 20H.3:

```
FLAG EQU 20H.3
LD H,#2H
BXOR C,@H+FLAG ; C XOR FLAG (20H.3)
```

CALL — Call Procedure

CALL dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	ADR	Call direct in page (14 bits)	3	4

Description: CALL calls a subroutine located at the destination address. The instruction adds three to the program counter to generate the return address and then pushes the result onto the stack, decreasing the stack pointer by six. The EMB and ERB are also pushed to the stack. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 16 K byte program memory address space.

Operand	Binary Code								Operation Notation
ADR	1	1	0	1	1	0	1	1	[(SP-1) (SP-2)] ← EMB, ERB
	0	1	a13	a12	a11	a10	a9	a8	[(SP-3) (SP-4)] ← PC7-0
	a7	a6	a5	a4	a3	a2	a1	a0	[(SP-5) (SP-6)] ← PC13-8

Example: The stack pointer value is 00H and the label 'PLAY' is assigned to program memory location 0E3FH. Executing the instruction

```
CALL  PLAY
```

at location 0123H will generate the following values:

```
SP    = 0FAH
0FFH  = 0H
0FEH  = EMB, ERB
0FDH  = 2H
0FCH  = 3H
0FBH  = 0H
0FAH  = 1H
PC    = 0E3FH
```

Data is written to stack locations 0FFH - 0FAH as follows:

SP - 6	(0FAH)	PC11 – PC8			
SP - 5	(0FBH)	0	0	PC13	PC12
SP - 4	(0FCH)	PC3 – PC0			
SP - 3	(0FDH)	PC7 – PC4			
SP - 2	(0FEH)	0	0	EMB	ERB
SP - 1	(0FFH)	0	0	0	0
SP →	(00H)				

CALLS — Call Procedure (Short)

CALLS dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	ADR	Call direct in page (11 bits)	2	3

Description: The CALLS instruction unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction. Then, it pushes the result onto the stack, decreasing the stack pointer six times. The higher bits of the PC, with the exception of the lower 11 bits, are cleared. The CALLS instruction can be used in the all range (0000H-3FFFH), but the subroutine must therefore be located within the 2 K byte block (0000H-07FFH) of program memory.

Operand	Binary Code								Operation Notation
ADR	1	1	1	0	1	a10	a9	a8	[(SP-1) (SP-2)] ← EMB, ERB
	a7	a6	a5	a4	a3	a2	a1	a0	[(SP-3) (SP-4)] ← PC7-0 [(SP-5) (SP-6)] ← PC14-8

Example: The stack pointer value is 00H and the label 'PLAY' is assigned to program memory location 0345H. Executing the instruction

CALLS PLAY

at location 0123H will generate the following values:

SP = 0FAH
 0FFH = 0H
 0FEH = EMB, ERB
 0FDH = 2H
 0FCH = 3H
 0FBH = 0H
 0FAH = 1H
 PC = 0345H

Data is written to stack locations 0FFH - 0FAH as follows:

SP - 6	(0FAH)	PC11 – PC8			
SP - 5	(0FBH)	0	PC14	PC13	PC12
SP - 4	(0FCH)	PC3 – PC0			
SP - 3	(0FDH)	PC7 – PC4			
SP - 2	(0FEH)	0	0	EMB	ERB
SP - 1	(0FFH)	0	0	0	0
SP →	(00H)				

CCF — Complement Carry Flag

CCF

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Complement carry flag	1	1

Description: The carry flag is complemented; if C = "1" it is changed to C = "0" and vice-versa.

Operand	Binary Code								Operation Notation
–	1	1	0	1	0	1	1	0	$C \leftarrow C$

Example: If the carry flag is logic zero, the instruction

CCF

changes the value to logic one.

COM — Complement Accumulator

COM A

Operation:	Operand	Operation Summary	Bytes	Cycles
	A	Complement accumulator (A)	2	2

Description: The accumulator value is complemented; if the bit value of A is "1", it is changed to "0" and vice versa.

Operand	Binary Code								Operation Notation
A	1	1	0	1	1	1	0	1	A ← A
	0	0	1	1	1	1	1	1	

Example: If the accumulator contains the value 4H (0100B), the instruction

COM A

leaves the value 0BH (1011B) in the accumulator.

CPSE — Compare and Skip If Equal

CPSE dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	R,#im	Compare and skip if register equals #im	2	2 + S
	@HL,#im	Compare and skip if indirect data memory equals #im	2	2 + S
	A,R	Compare and skip if A equals R	2	2 + S
	A,@HL	Compare and skip if A equals indirect data memory	1	1 + S
	EA,@HL	Compare and skip if EA equals indirect data memory	2	2 + S
	EA,RR	Compare and skip if EA equals RR	2	2 + S

Description: CPSE compares the source operand (subtracts it from) the destination operand, and skips the next instruction if the values are equal. Neither operand is affected by the comparison.

Operand	Binary Code								Operation Notation
R,#im	1	1	0	1	1	0	0	1	Skip if R = im
	d3	d2	d1	d0	0	r2	r1	r0	
@HL,#im	1	1	0	1	1	1	0	1	Skip if (HL) = im
	0	1	1	1	d3	d2	d1	d0	
A,R	1	1	0	1	1	1	0	1	Skip if A = R
	0	1	1	0	1	r2	r1	r0	
A,@HL	0	0	1	1	1	0	0	0	Skip if A = (HL)
EA,@HL	1	1	0	1	1	1	0	0	Skip if A = (HL), E = (HL+1)
	0	0	0	0	1	0	0	1	
EA,RR	1	1	0	1	1	1	0	0	Skip if EA = RR
	1	1	1	0	1	r2	r1	0	

Example: The extended accumulator contains the value 34H and register pair HL contains 56H. The second instruction (RET) in the instruction sequence

```
CPSE   EA,HL
RET
```

is not skipped. That is, the subroutine returns since the result of the comparison is 'not equal.'

DECS — Decrement and Skip on Borrow

DECS dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	R	Decrement register (R); skip on borrow	1	1 + S
	RR	Decrement register pair (RR); skip on borrow	2	2 + S

Description: The destination is decremented by one. An original value of 00H will underflow to 0FFH. If a borrow occurs, a skip is executed. The carry flag value is unaffected.

Operand	Binary Code								Operation Notation
R	0	1	0	0	1	r2	r1	r0	$R \leftarrow R-1$; skip on borrow
RR	1	1	0	1	1	1	0	0	$RR \leftarrow RR-1$; skip on borrow
	1	1	0	1	1	r2	r1	0	

- Examples:**
- Register pair HL contains the value 7FH (01111111B). The following instruction leaves the value 7EH in register pair HL:


```
DECS    HL
```
 - Register A contains the value 0H. The following instruction sequence leaves the value 0FFH in register A. Since a "borrow" occurs, the 'CALL PLAY1' instruction is skipped and the 'CALL PLAY2' instruction is executed:


```
DECS    A                ; "Borrow" occurs
CALL    PLAY1            ; Skipped
CALL    PLAY2            ; Executed
```


DI — Disable Interrupts

DI

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Disable all interrupts	2	2

Description: Bit 3 of the interrupt priority register IPR, IME, is cleared to logic zero, disabling all interrupts. Interrupts can still set their respective interrupt status latches, but the CPU will not directly service them.

Operand	Binary Code								Operation Notation
–	1	1	1	1	1	1	1	0	IME ← 0
	1	0	1	1	0	0	1	0	

Example: If the IME bit (bit 3 of the IPR) is logic one (e.g., all instructions are enabled), the instruction DI sets the IME bit to logic zero, disabling all interrupts.

EI — Enable Interrupts

EI

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Enable all interrupts	2	2

Description: Bit 3 of the interrupt priority register IPR (IME) is set to logic one. This allows all interrupts to be serviced when they occur, assuming they are enabled. If an interrupt's status latch was previously enabled by an interrupt, this interrupt can also be serviced.

Operand	Binary Code								Operation Notation
–	1	1	1	1	1	1	1	1	IME ← 1
	1	0	1	1	0	0	1	0	

Example: If the IME bit (bit 3 of the IPR) is logic zero (e.g., all instructions are disabled), the instruction

EI

sets the IME bit to logic one, enabling all interrupts.

IDLE — Idle Operation

IDLE

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Engage CPU idle mode	2	2

Description: IDLE causes the CPU clock to stop while the system clock continues oscillating by setting bit 2 of the power control register (PCON). After an IDLE instruction has been executed, peripheral hardware remains operative.

In application programs, an IDLE instruction must be immediately followed by at least three NOP instructions. This ensures an adequate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructions are not used after IDLE instruction, leakage current could be flown because of the floating state in the internal bus.

Operand	Binary Code								Operation Notation
–	1	1	1	1	1	1	1	1	PCON.2 ← 1
	1	0	1	0	0	0	1	1	

Example: The instruction sequence

```
IDLE
NOP
NOP
NOP
```

sets bit 2 of the PCON register to logic one, stopping the CPU clock. The three NOP instructions provide the necessary timing delay for clock stabilization before the next instruction in the program sequence is executed.

INCS — Increment and Skip on Carry

INCS dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	R	Increment register (R); skip on carry	1	1 + S
	DA	Increment direct data memory; skip on carry	2	2 + S
	@HL	Increment indirect data memory; skip on carry	2	2 + S
	RRb	Increment register pair (RRb); skip on carry	1	1 + S

Description: The instruction INCS increments the value of the destination operand by one. An original value of 0FH will, for example, overflow to 00H. If a carry occurs, the next instruction is skipped. The carry flag value is unaffected.

Operand	Binary Code								Operation Notation
R	0	1	0	1	1	r2	r1	r0	$R \leftarrow R + 1$; skip on carry
DA	1	1	0	0	1	0	1	0	$DA \leftarrow DA + 1$; skip on carry
	a7	a6	a5	a4	a3	a2	a1	a0	
@HL	1	1	0	1	1	1	0	1	$(HL) \leftarrow (HL) + 1$; skip on carry
	0	1	1	0	0	0	1	0	
RRb	1	0	0	0	0	r2	r1	0	$RRb \leftarrow RRb + 1$; skip on carry

Example: Register pair HL contains the value 7EH (01111110B). RAM location 7EH contains 0FH. The instruction sequence

```
INCS    @HL            ; 7EH ← "0"
INCS    HL              ; Skip
INCS    @HL            ; 7EH ← "1"
```

leaves the register pair HL with the value 7EH and RAM location 7EH with the value 1H. Since a carry occurred, the second instruction is skipped. The carry flag value remains unchanged.

IRET — Return From Interrupt

IRET

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Return from interrupt	1	3

Description: IRET is used at the end of an interrupt service routine. It pops the PC values successively from the stack and restores them to the program counter. The stack pointer is incremented by six and the PSW, enable memory bank (EMB) bit, and enable register bank (ERB) bit are also automatically restored to their pre-interrupt values. Program execution continues from the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower-level or same-level interrupt was pending when the IRET was executed, IRET will be executed before the pending interrupt is processed.

Since the 'a14' bit of an interrupt return address is not stored in the stack, this bit location is always interpreted as a logic zero. The starting address in the ROM must for this reason be located in 0000H-3FFFH.

Operand	Binary Code							Operation Notation	
–	1	1	0	1	0	1	0	1	PC13-8 ← (SP + 1) (SP) PC7-0 ← SP + 3) (SP + 2) PSW ← (SP + 5) (SP + 4) SP ← SP + 6

Example: The stack pointer contains the value 0FAH. An interrupt is detected in the instruction at location 0123H. RAM locations 0FDH, 0FCH, and 0FAH contain the values 2H, 3H, and 1H, respectively. The instruction

IRET

leaves the stack pointer with the value 00H and the program returns to continue execution at location 0123H.

During a return from interrupt, data is popped from the stack to the program counter. The data in stack locations 0FFH-0FAH is organized as follows:

SP →	(0FAH)	PC11 – PC8			
SP + 1	(0FBH)	0	0	PC13	PC12
SP + 2	(0FCH)	PC3 – PC0			
SP + 3	(0FDH)	PC7 – PC4			
SP + 4	(0FEH)	IS1	IS0	EMB	ERB
SP + 5	(0FFH)	C	SC2	SC1	SC0
SP + 6	(00H)				

JP — Jump

JP dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	ADR	Jump to direct address (14 bits)	3	3

Description: JP causes an unconditional branch to the indicated address by replacing the contents of the program counter with the address specified in the destination operand. The destination can be anywhere in the 16 K byte program memory address space.

Operand	Binary Code								Operation Notation
ADR	1	1	0	1	1	0	1	1	PC13-0 ← ADR13-0
	0	0	a13	a12	a11	a10	a9	a8	
	a7	a6	a5	a4	a3	a2	a1	a0	

Example: The label 'SYSICON' is assigned to the instruction at program location 07FFH. The instruction
 JP SYSICON
 at location 0123H will load the program counter with the value 07FFH.

JPS — Jump (Short)

JPS dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	ADR	Jump direct in page (12 bits)	2	2

Description: JPS causes an unconditional branch to the indicated address with the 4 K byte program memory address space. Bits 0-11 of the program counter are replaced with the directly specified address. The destination address for this jump is specified to the assembler by a label or by an actual address in program memory.

Operand	Binary Code								Operation Notation
ADR	1	0	0	1	a11	a10	a9	a8	PC13-0 ← PC13-12 + ADR11-0
	a7	a6	a5	a4	a3	a2	a1	a0	

Example: The label 'SUB' is assigned to the instruction at program memory location 00FFH. The instruction

```
JPS   SUB
```

at location 0EABH will load the program counter with the value 00FFH. Normally, the JPS instruction jumps to the address in the block in which the instruction is located. If the first byte of the instruction code is located at address xFFEh or xFFFh, the instruction will jump to the next block. If the instruction 'JPS SUB' were located instead at program memory address 0FFEh or 0FFFh, the instruction 'JPS SUB' would load the PC with the value 10FFh, causing a program malfunction.

JR — Jump Relative (Very Short)

JR dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	#im	Branch to relative immediate address	1	2
	@WX	Branch relative to contents of WX register	2	3
	@EA	Branch relative to contents of EA	2	3

Description: JR causes the relative address to be added to the program counter and passes control to the instruction whose address is now in the PC. The range of the relative address is current PC - 15 to current PC + 16. The destination address for this jump is specified to the assembler by a label, an actual address, or by immediate data using a plus sign (+) or a minus sign (-).

For immediate addressing, the (+) range is from 2 to 16 and the (-) range is from -1 to -15. If a 0, 1, or any other number that is outside these ranges are used, the assembler interprets it as an error.

For JR @WX and JR @EA branch relative instructions, the valid range for the relative address is 0H-0FFH. The destination address for these jumps can be specified to the assembler by a label that lies anywhere within the current 256-byte block.

Normally, the 'JR @WX' and 'JR @EA' instructions jump to the address in the page in which the instruction is located. However, if the first byte of the instruction code is located at address xxFEH or xxFFH, the instruction will jump to the next page.

Operand	Binary Code								Operation Notation
#im *									PC13-0 ← ADR (PC-15 to PC+16)
@WX	1	1	0	1	1	1	0	1	PC13-0 ← PC13-8 + (WX)
	0	1	1	0	0	1	0	0	
@EA	1	1	0	1	1	1	0	1	PC13-0 ← PC13-8 + (EA)
	0	1	1	0	0	0	0	0	

	First Byte								Condition
* JR #im	0	0	0	1	a3	a2	a1	a0	PC ← PC+2 to PC+16
	0	0	0	0	a3	a2	a1	a0	PC ← PC-1 to PC-15

JR — Jump Relative (Very Short)

JR (Continued)

Examples: 1. A short form for a relative jump to label 'KK' is the instruction

```
JR    KK
```

where 'KK' must be within the allowed range of current PC-15 to current PC+16. The JR instruction has in this case the effect of an unconditional JP instruction.

2. In the following instruction sequence, if the instruction 'LD WX, #02H' were to be executed in place of 'LD WX,#00H', the program would jump to 1004H and 'JPS CCC' would be executed. If 'LD WX,#03H' were to be executed, the jump would be to 1006H and 'JPS DDD' would be executed.

```

                ORG    1000H
                JPS    AAA
                JPS    BBB
                JPS    CCC
                JPS    DDD
XXX   LD        WX,#00H ; WX ← 00H
        LD        EA,WX
        ADS       WX,EA ; WX ← (WX) + (EA)
        JR        @WX  ; Current PC12-8 (10H) + WX (00H) = 1000H
                        ; Jump to address 1000H and execute JPS AAA

```

3. Here is another example:

```

ORG    1100H

LD     A,#0H
LD     A,#1H
LD     A,#2H
LD     A,#3H
LD     30H,A           ; Address 30H ← A
JPS    YYY
XXX   LD     EA,#00H ; EA ← 00H
JR     @EA           ; Jump to address 1100H
                        ; Address 30H ← 00H

```

If 'LD EA,#01H' were to be executed in place of 'LD EA,#00H', the program would jump to 1101H and address 30H would contain the value 1H. If 'LD EA,#02H' were to be executed, the jump would be to 1102H and address 30H would contain the value 2H.

LD — Load

LD dst,src

Operation:

Operand	Operation Summary	Bytes	Cycles
A,#im	Load 4-bit immediate data to A	1	1
A,@RRa	Load indirect data memory contents to A	1	1
A,DA	Load direct data memory contents to A	2	2
A,Ra	Load register contents to A	2	2
Ra,#im	Load 4-bit immediate data to register	2	2
RR,#imm	Load 8-bit immediate data to register	2	2
DA,A	Load contents of A to direct data memory	2	2
Ra,A	Load contents of A to register	2	2
EA,@HL	Load indirect data memory contents to EA	2	2
EA,DA	Load direct data memory contents to EA	2	2
EA,RRb	Load register contents to EA	2	2
@HL,A	Load contents of A to indirect data memory	1	1
DA,EA	Load contents of EA to data memory	2	2
RRb,EA	Load contents of EA to register	2	2
@HL,EA	Load contents of EA to indirect data memory	2	2

Description: The contents of the source are loaded into the destination. The source's contents are unaffected.

If an instruction such as 'LD A,#im' (LD EA,#imm) or 'LD HL,#imm' is written more than two times in succession, only the first LD will be executed; the other similar instructions that immediately follow the first LD will be treated like a NOP. This is called the 'redundancy effect' (see examples below).

Operand	Binary Code								Operation Notation
A,#im	1	0	1	1	d3	d2	d1	d0	A ← im
A,@RRa	1	0	0	0	1	i2	i1	i0	A ← (RRa)
A,DA	1	0	0	0	1	1	0	0	A ← DA
	a7	a6	a5	a4	a3	a2	a1	a0	
A,Ra	1	1	0	1	1	1	0	1	A ← Ra
	0	0	0	0	1	r2	r1	r0	
Ra,#im	1	1	0	1	1	0	0	1	Ra ← im
	d3	d2	d1	d0	1	r2	r1	r0	

LD – Load

LD (Continued)

Description:	Operand	Binary Code								Operation Notation
RR,#imm		1	0	0	0	0	r2	r1	1	RR ← imm
		d7	d6	d5	d4	d3	d2	d1	d0	
DA,A		1	0	0	0	1	0	0	1	DA ← A
		a7	a6	a5	a4	a3	a2	a1	a0	
Ra,A		1	1	0	1	1	1	0	1	Ra ← A
		0	0	0	0	0	r2	r1	r0	
EA,@HL		1	1	0	1	1	1	0	0	A ← (HL), E ← (HL + 1)
		0	0	0	0	1	0	0	0	
EA,DA		1	1	0	0	1	1	1	0	A ← DA, E ← DA + 1
		a7	a6	a5	a4	a3	a2	a1	a0	
EA,RRb		1	1	0	1	1	1	0	0	EA ← RRb
		1	1	1	1	1	r2	r1	0	
@HL,A		1	1	0	0	0	1	0	0	(HL) ← A
DA,EA		1	1	0	0	1	1	0	1	DA ← A, DA + 1 ← E
		a7	a6	a5	a4	a3	a2	a1	a0	
RRb,EA		1	1	0	1	1	1	0	0	RRb ← EA
		1	1	1	1	0	r2	r1	0	
@HL,EA		1	1	0	1	1	1	0	0	(HL) ← A, (HL + 1) ← E
		0	0	0	0	0	0	0	0	

Examples: 1. RAM location 30H contains the value 4H. The RAM location values are 40H, 41H and 0AH, 3H respectively. The following instruction sequence leaves the value 40H in point pair HL, 0AH in the accumulator and in RAM location 40H, and 3H in register E.

```
LD    HL,#30H      ; HL ← 30H
LD    A,@HL       ; A ← 4H
LD    HL,#40H     ; HL ← 40H
LD    EA,@HL      ; A ← 0AH, E ← 3H
LD    @HL,A       ; RAM (40H) ← 0AH
```

LD — Load

LD (Continued)

Examples: 2. If an instruction such as LD A,#im (LD EA,#imm) or LD HL,#imm is written more than two times in succession, only the first LD is executed; the next instructions are treated as NOPs. Here are two examples of this 'redundancy effect':

```
LD    A,#1H      ; A ← 1H
LD    EA,#2H     ; NOP
LD    A,#3H     ; NOP
LD    23H,A     ; (23H) ← 1H

LD    HL,#10H   ; HL ← 10H
LD    HL,#20H   ; NOP
LD    A,#3H     ; A ← 3H
LD    EA,#35    ; NOP
LD    @HL,A     ; (10H) ← 3H
```

The following table contains descriptions of special characteristics of the LD instruction when used in different addressing modes:

<u>Instruction</u>	<u>Operation Description and Guidelines</u>
LD A,#im	Since the 'redundancy effect' occurs with instructions like LD EA,#imm, if this instruction is used consecutively, the second and additional instructions of the same type will be treated like NOPs.
LD A,@RRa	Load the data memory contents pointed to by 8-bit RRa register pairs (HL, WX, WL) to the A register.
LD A,DA	Load direct data memory contents to the A register.
LD A,Ra	Load 4-bit register Ra (E, L, H, X, W, Z, Y) to the A register.
LD Ra,#im	Load 4-bit immediate data into the Ra register (E, L, H, X, W, Y, Z).
LD RR,#imm	Load 8-bit immediate data into the Ra register (EA, HL, WX, YZ). There is a redundancy effect if the operation addresses the HL or EA registers.
LD DA,A	Load contents of register A to direct data memory address.
LD Ra,A	Load contents of register A to 4-bit Ra register (E, L, H, X, W, Z, Y).

LD — Load

LD (Concluded)

Examples:	<u>Instruction</u>	<u>Operation Description and Guidelines</u>
	LD EA,@HL	Load data memory contents pointed to by 8-bit register HL to the A register, and the contents of HL+1 to the E register. The contents of register L must be an even number. If the number is odd, the LSB of register L is recognized as a logic zero (an even number), and it is not replaced with the true value. For example, 'LD HL,#36H' loads immediate 36H to HL and the next instruction 'LD EA,@HL' loads the contents of 36H to register A and the contents of 37H to register E.
	LD EA,DA	Load direct data memory contents of DA to the A register, and the next direct data memory contents of DA + 1 to the E register. The DA value must be an even number. If it is an odd number, the LSB of DA is recognized as a logic zero (an even number), and it is not replaced with the true value. For example, 'LD EA,37H' loads the contents of 36H to the A register and the contents of 37H to the E register.
	LD EA,RRb	Load 8-bit RRb register (HL, WX, YZ) to the EA register. H, W, and Y register values are loaded into the E register, and the L, X, and Z values into the A register.
	LD @HL,A	Load A register contents to data memory location pointed to by the 8-bit HL register value.
	LD DA,EA	Load the A register contents to direct data memory and the E register contents to the next direct data memory location. The DA value must be an even number. If it is an odd number, the LSB of the DA value is recognized as logic zero (an even number), and is not replaced with the true value.
	LD RRb,EA	Load contents of EA to the 8-bit RRb register (HL, WX, YZ). The E register is loaded into the H, W, and Y register and the A register into the L, X, and Z register.
	LD @HL,EA	Load the A register to data memory location pointed to by the 8-bit HL register, and the E register contents to the next location, HL + 1. The contents of the L register must be an even number. If the number is odd, the LSB of the L register is recognized as logic zero (an even number), and is not replaced with the true value. For example, 'LD HL,#36H' loads immediate 36H to register HL; the instruction 'LD @HL,EA' loads the contents of A into address 36H and the contents of E into address 37H.

LDB — Load Bit

LDB dst,src.b
LDB dst.b,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	mema.b,C	Load carry bit to a specified memory bit	2	2
	memb.@L,C	Load carry bit to a specified indirect memory bit	2	2
	@H+DA.b,C		2	2
	C,mema.b	Load memory bit to a specified carry bit	2	2
	C,memb.@L	Load indirect memory bit to a specified carry bit	2	2
	C,@H+DA.b		2	2

Description: The Boolean variable indicated by the first or second operand is copied into the location specified by the second or first operand. One of the operands must be the carry flag; the other may be any directly or indirectly addressable bit. The source is unaffected.

Operand	Binary Code								Operation Notation
mema.b,C *	1	1	1	1	1	1	0	0	mema.b ← C
memb.@L,C	1	1	1	1	1	1	0	0	memb.7-2 + [L.3-2]. [L.1-0] ← C
	0	1	0	0	a5	a4	a3	a2	
@H+DA.b,C	1	1	1	1	1	1	0	0	H + [DA.3-0].b ← (C)
	0	0	b1	b0	a3	a2	a1	a0	
C,mema.b*	1	1	1	1	0	1	0	0	C ← mema.b
C,memb.@L	1	1	1	1	0	1	0	0	C ← memb.7-2 + [L.3-2] . [L.1-0]
	0	1	0	0	a5	a4	a3	a2	
C,@H+DA.b	1	1	1	1	0	1	0	0	C ← [H + DA.3-0].b
	0	0	b1	b0	a3	a2	a1	a0	

		Second Byte						Bit Addresses		
*	mema.b	1	0	b1	b0	a3	a2	a1	a0	FB0H-FBFH
		1	1	b1	b0	a3	a2	a1	a0	FF0H-FFFH

LDB — Load Bit

LDB (Continued)

Examples: 1. The carry flag is set and the data value at input pin P1.0 is logic zero. The following instruction clears the carry flag to logic zero.

```
LDB    C,P1.0
```

2. The P1 address is FF1H and the L register contains the value 9H (1001B). The address (memb.7-2) is 111100B and (L.3-2) is 10B. The resulting address is 11110010B or FF2H and P2 is addressed. The bit value (L.1-0) is specified as 01B (bit 1).

```
LD     L,#9H
LDB    CO,P1.@L      ; P1.@L specifies P2.1 and C ← P2.1
```

3. The H register contains the value 2H and FLAG = 20H.3. The address for H is 0010B and for FLAG(3-0) the address is 0000B. The resulting address is 00100000B or 20H. The bit value is 3. Therefore, @H+FLAG = 20H.3.

```
FLAG   EQU     20H.3
LD     H,#2H
LDB    C,@H+FLAG    ; C ← FLAG (20H.3)
```

4. The following instruction sequence sets the carry flag and the loads the "1" data value to the output pin P2.0, setting it to output mode:

```
SCF                                ; C ← "1"
LDB    P2.0,C                       ; P2.0 ← "1"
```

5. The P1 address is FF1H and L = 9H (1001B). The address (memb.7-2) is 111100B and (L.3-2) is 10B. The resulting address, 11110010B specifies P2. The bit value (L.1-0) is specified as 01B (bit 1). Therefore, P1.@L = P2.1.

```
SCF                                ; C ← "1"
LD     L,#9H
LDB    P1.@L,C                       ; P1.@L specifies P2.1
                                           ; P2.1 ← "1"
```

6. In this example, H = 2H and FLAG = 20H.3 and the address 20H is specified. Since the bit value is 3, @H+FLAG = 20H.3:

```
FLAG   EQU     20H.3
RCF                                ; C ← "0"
LD     H,#2H
LDB    @H+FLAG,C                     ; FLAG(20H.3) ← "0"
```

NOTE: Port pin names used in examples 4 and 5 may vary with different SAM48 devices.

LDC — Load Code Byte

LDC dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	EA,@WX	Load code byte from WX to EA	1	3
	EA,@EA	Load code byte from EA to EA	1	3

Description: This instruction is used to load a byte from program memory into an extended accumulator. The address of the byte fetched is the six highest bit values in the program counter and the contents of an 8-bit working register (either WX or EA). The contents of the source are unaffected.

Operand	Binary Code								Operation Notation
EA,@WX	1	1	0	0	1	1	0	0	$EA \leftarrow [PC13-8 + (WX)]$
EA,@EA	1	1	0	0	1	0	0	0	$EA \leftarrow [PC13-8 + (EA)]$

Examples: 1. The following instructions will load one of four values defined by the define byte (DB) directive to the extended accumulator:

```
LD      EA,#00H
CALL   DISPLAY
JPS    MAIN

ORG    0500H

DB     66H
DB     77H
DB     88H
DB     99H
DISPLAY LDC      EA,@EA ; EA ← address 0500H = 66H
RET
```

If the instruction 'LD EA,#01H' is executed in place of 'LD EA,#00H', The content of 0501H (77H) is loaded to the EA register. If 'LD EA,#02H' is executed, the content of address 0502H (88H) is loaded to EA.

LDC — Load Code Byte

LDC (Continued)

Examples: 2. The following instructions will load one of four values defined by the define byte (DB) directive to the extended accumulator:

```

ORG      0500H

DB       66H
DB       77H
DB       88H
DB       99H
DISPLAY LD      WX,#00H
LDC      EA,@WX      ; EA ← address 0500H = 66H
RET

```

If the instruction 'LD WX,#01H' is executed in place of 'LD WX,#00H', then
EA ← address 0501H = 77H.

If the instruction 'LD WX,#02H' is executed in place of 'LD WX,#00H', then
EA ← address 0502H = 88H.

3. Normally, the LDC EA, @EA and the LDC EA, @WX instructions reference the table data on the page on which the instruction is located. If, however, the instruction is located at address xxFFH, it will reference table data on the next page. In this example, the upper 4 bits of the address at location 0200H is loaded into register E and the lower 4 bits into register A:

```

ORG      01FDH

01FDH   LD      WX,#00H
01FFH   LDC     EA,@WX ; E ← upper 4 bits of 0200H address
                          ; A ← lower 4 bits of 0200H address

```

4. Here is another example of page referencing with the LDC instruction:

```

ORG      0100H

DB       67H
SMB     0
LD      HL,#30H      ; Even number
LD      WX,#00H
LDC     EA,@WX      ; E ← upper 4 bits of 0100H address
                          ; A ← lower 4 bits of 0100H address
LD      @HL,EA      ; RAM (30H) ← 7, RAM (31H) ← 6

```

LDD — Load Data Memory and Decrement

LDD dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Load indirect data memory contents to A; decrement register L contents and skip on borrow	1	2 + S

Description: The contents of a data memory location are loaded into the accumulator, and the contents of the register L are decreased by one. If a "borrow" occurs (e.g., if the resulting value in register L is 0FH), the next instruction is skipped. The contents of data memory and the carry flag value are not affected.

Operand	Binary Code								Operation Notation
A,@HL	1	0	0	0	1	0	1	1	A ← (HL), then L ← L-1; skip if L = 0FH

Example: In this example, assume that register pair HL contains 20H and internal RAM location 20H contains the value 0FH:

```
LD      HL,#20H
LDD     A,@HL      ; A ← (HL) and L ← L-1
JPS     XXX        ; Skip
JPS     YYY        ; H ← 2H and L ← 0FH
```

The instruction 'JPS XXX' is skipped since a "borrow" occurred after the 'LDD A,@HL' and instruction 'JPS YYY' is executed.

LDI — Load Data Memory and Increment

LDI dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Load indirect data memory to A; increment register L contents and skip on overflow	1	2 + S

Description: The contents of a data memory location are loaded into the accumulator, and the contents of the register L are incremented by one. If an overflow occurs (e.g., if the resulting value in register L is 0H), the next instruction is skipped. The contents of data memory and the carry flag value are not affected.

Operand	Binary Code								Operation Notation
A,@HL	1	0	0	0	1	0	1	0	$A \leftarrow (HL)$, then $L \leftarrow L+1$; skip if $L = 0H$

Example: Assume that register pair HL contains the address 2FH and internal RAM location 2FH contains the value 0FH:

```
LD      HL,#2FH
LDI     A,@HL      ; A ← (HL) and L ← L+1
JPS     XXX        ; Skip
JPS     YYY        ; H ← 2H and L ← 0H
```

The instruction 'JPS XXX' is skipped since an overflow occurred after the 'LDI A,@HL' and the instruction 'JPS YYY' is executed.

NOP — No Operation

NOP

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	No operation	1	1

Description: No operation is performed by a NOP instruction. It is typically used for timing delays.

One NOP causes a 1-cycle delay: with a 1 μ s cycle time, five NOPs would therefore cause a 5 μ s delay. Program execution continues with the instruction immediately following the NOP. Only the PC is affected. At least three NOP instructions should follow a STOP or IDLE instruction.

Operand	Binary Code								Operation Notation
–	1	0	1	0	0	0	0	0	No operation

Example: Three NOP instructions follow the STOP instruction to provide a short interval for clock stabilization before power-down mode is initiated:

```
STOP
NOP
NOP
NOP
```

OR — Logical OR

OR dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A, #im	Logical-OR immediate data to A	2	2
	A, @HL	Logical-OR indirect data memory contents to A	1	1
	EA,RR	Logical-OR double register to EA	2	2
	RRb,EA	Logical-OR EA to double register	2	2

Description: The source operand is logically ORed with the destination operand. The result is stored in the destination. The contents of the source are unaffected.

Operand	Binary Code								Operation Notation
A, #im	1	1	0	1	1	1	0	1	A ← A OR im
	0	0	1	0	d3	d2	d1	d0	
A, @HL	0	0	1	1	1	0	1	0	A ← A OR (HL)
EA,RR	1	1	0	1	1	1	0	0	EA ← EA OR RR
	0	0	1	0	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	RRb ← RRb OR EA
	0	0	1	0	0	r2	r1	0	

Example: If the accumulator contains the value 0C3H (11000011B) and register pair HL the value 55H (01010101B), the instruction

OR EA,@HL

leaves the value 0D7H (11010111B) in the accumulator .

POP — POP From Stack

POP dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	RR	Pop to register pair from stack	1	1
	SB	Pop SMB and SRB values from stack	2	2

Description: The contents of the RAM location addressed by the stack pointer is read, and the SP is incremented by two. The value read is then transferred to the variable indicated by the destination operand.

Operand	Binary Code								Operation Notation
RR	0	0	1	0	1	r2	r1	0	$RR_L \leftarrow (SP), RR_H \leftarrow (SP+1)$ $SP \leftarrow SP+2$
SB	1	1	0	1	1	1	0	1	$(SRB) \leftarrow (SP), SMB \leftarrow (SP+1),$ $SP \leftarrow SP+2$
	0	1	1	0	0	1	1	0	

Example: The SP value is equal to 0EDH, and RAM locations 0EFH through 0EDH contain the values 2H, 3H, and 4H, respectively. The instruction

```
POP       HL
```

leaves the stack pointer set to 0EFH and the data pointer pair HL set to 34H.

PUSH — PUSH onto Stack

PUSH src

Operation:	Operand	Operation Summary	Bytes	Cycles
	RR	Push register pair onto stack	1	1
	SB	Push SMB and SRB values onto stack	2	2

Description: The SP is then decreased by two and the contents of the source operand are copied into the RAM location addressed by the stack pointer, thereby adding a new element to the top of the stack.

Operand	Binary Code								Operation Notation
RR	0	0	1	0	1	r2	r1	1	$(SP-1) \leftarrow RR_H, (SP-2) \leftarrow RR_L$ $SP \leftarrow SP-2$
SB	1	1	0	1	1	1	0	1	$(SP-1) \leftarrow SMB, (SP-2) \leftarrow SRB;$ $(SP) \leftarrow SP-2$
	0	1	1	0	0	1	1	1	

Example: As an interrupt service routine begins, the stack pointer contains the value 0FAH and the data pointer register pair HL contains the value 20H. The instruction

PUSH HL

leaves the stack pointer set to 0F8H and stores the values 2H and 0H in RAM locations 0F9H and 0F8H, respectively.

RCF — Reset Carry Flag

RCF

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Reset carry flag to logic zero	1	1

Description: The carry flag is cleared to logic zero, regardless of its previous value.

Operand	Binary Code								Operation Notation
–	1	1	1	0	0	1	1	0	$C \leftarrow 0$

Example: Assuming the carry flag is set to logic one, the instruction

RCF

resets (clears) the carry flag to logic zero.

REF — Reference Instruction

REF dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	memc	Reference code	1	1 ^(note)

NOTE: The instruction referenced by REF determines instruction cycles.

Description: The REF instruction is used to rewrite into 1-byte form, arbitrary 2-byte or 3-byte instructions (or two 1-byte instructions) stored in the REF instruction reference area in program memory. REF reduces the number of program memory accesses for a program.

Operand	Binary Code								Operation Notation
memc	t7	t6	t5	t4	t3	t2	t1	t0	PC13-0 ← memc5-0 + (memc+1).7-0

TJP and TCALL are 2-byte pseudo-instructions that are used only to specify the reference area:

- When the reference area is specified by the TJP instruction,
 $\text{memc.7-6} = 00$
 $\text{PC13-0} \leftarrow \text{memc.5-0} + (\text{memc}+1).7-0$
- When the reference area is specified by the TCALL instruction,
 $\text{memc.7-6} = 01$
 $[(\text{SP}-1) (\text{SP}-2)] \leftarrow \text{EMB, ERB}$
 $[(\text{SP}-3) (\text{SP}-4)] \leftarrow \text{PC7-0}$
 $[(\text{SP}-5) (\text{SP}-6)] \leftarrow \text{PC13-8}$
 $\text{SP} \leftarrow \text{SP}-6$
 $\text{PC-0} \leftarrow \text{memc.5-0} + (\text{memc}+1).7-0$

When the reference area is specified by any other instruction, the 'memc' and 'memc + 1' instructions are executed.

Instructions referenced by REF occupy 2 bytes of memory space (for two 1-byte instructions or one 2-byte instruction) and must be written as an even number from 0020H to 007FH in ROM. In addition, the destination address of the TJP and TCALL instructions must be located with the 3FFFH address. TJP and TCALL are reference instructions for JP/JPS and CALL/CALLS.

If the instruction following a REF is subject to the 'redundancy effect', the redundant instruction is skipped. If, however, the REF follows a redundant instruction, it is executed.

On the other hand, the binary code of a REF instruction is 1 byte. The upper 4 bits become the higher address bits of the referenced instruction, and the lower 4 bits of the referenced instruction becomes the lower address, producing a total of 8 bits or 1 byte (see Example 3 below).

NOTE: If the MSB value of the first one-byte binary code in instruction is "0", the instruction cannot be referenced by a REF instruction.

REF — Reference Instruction

REF (Continued)

Examples: 1. Instructions can be executed efficiently using REF, as shown in the following example:

```

                ORG    0020H
AAA    LD    HL,#00H
BBB    LD    EA,#FFH
CCC    TCALL SUB1
DDD    TJP   SUB2
      .
      .
      .
                ORG 0080H
REF    AAA    ; LD    HL,#00H
REF    BBB    ; LD    EA,#FFH
REF    CCC    ; CALL  SUB1
REF    DDD    ; JP    SUB2

```

2. The following example shows how the REF instruction is executed in relation to LD instructions that have a 'redundancy effect':

```

                ORG    0020H
AAA    LD    EA,#40H
      .
      .
      .
                ORG    0100H
LD     EA,#30H
REF    AAA                ;      Not skipped
      .
      .
      .
REF    AAA
LD     EA,#50H            ;      Skipped
SRB    2

```

REF — Reference Instruction

REF (Concluded)

Examples: 3. In this example the binary code of 'REF A1' at locations 20H-21H is 20H, for 'REF A2' at locations 22H-23H, it is 21H, and for 'REF A3' at 24H-25H, the binary code is 22H :

<u>Opcode</u>	<u>Symbol</u>	<u>Instruction</u>				
		ORG	0020H			
83 00	A1	LD	HL,#00H			
83 03	A2	LD	HL,#03H			
83 05	A3	LD	HL,#05H			
83 10	A4	LD	HL,#10H			
83 26	A5	LD	HL,#26H			
83 08	A6	LD	HL,#08H			
83 0F	A7	LD	HL,#0FH			
83 F0	A8	LD	HL,#0F0H			
83 67	A9	LD	HL,#067H			
41 0B	A10	TCALL	SUB1			
01 0D	A11	TJP	SUB2			
		•				
		•				
		•				
		ORG	0100H			
20		REF	A1	:	LD	HL,#00H
21		REF	A2	:	LD	HL,#03H
22		REF	A3	:	LD	HL,#05H
23		REF	A4	:	LD	HL,#10H
24		REF	A5	:	LD	HL,#26H
25		REF	A6	:	LD	HL,#08H
26		REF	A7	:	LD	HL,#0FH
27		REF	A8	:	LD	HL,#0F0H
30		REF	A9	:	LD	HL,#067H
31		REF	A10	:	CALL	SUB1
32		REF	A11	:	JP	SUB2

RET — Return From Subroutine

RET

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Return from subroutine	1	3

Description: RET pops the PC values successively from the stack, incrementing the stack pointer by six. Program execution continues from the resulting address, generally the instruction immediately following a CALL or CALLS.

Operand	Binary Code								Operation Notation
–	1	1	0	0	0	1	0	1	PC13-8 ← (SP + 1) (SP) PC7-0 ← (SP + 3) (SP + 2) EMB,ERB ← (SP + 4) SP ← SP + 6

Example: The stack pointer contains the value 0FAH. RAM locations 0FAH, 0FBH, 0FCH, and 0FDH contain 1H, 0H, 5H, and 2H, respectively. The instruction

RET

leaves the stack pointer with the new value of 00H and program execution continues from location 0125H.

During a return from subroutine, PC values are popped from stack locations as follows:

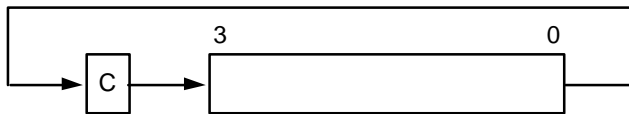
SP →	(0FAH)	PC11 – PC8			
SP + 1	(0FBH)	0	0	PC13	PC12
SP + 2	(0FCH)	PC3 – PC0			
SP + 3	(0FDH)	PC7 – PC4			
SP + 4	(0FEH)	0	0	EMB	ERB
SP + 5	(0FFH)	0	0	0	0
SP + 6	(000H)				

RRC — Rotate Accumulator Right through Carry

RRC A

Operation:	Operand	Operation Summary	Bytes	Cycles
	A	Rotate right through carry bit	1	1

Description: The four bits in the accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag and the original carry value moves into the bit 3 accumulator position.



Operand	Binary Code								Operation Notation
A	1	0	0	0	1	0	0	0	$C \leftarrow A.0, A3 \leftarrow C$ $A.n-1 \leftarrow A.n \quad (n = 1, 2, 3)$

Example: The accumulator contains the value 5H (0101B) and the carry flag is cleared to logic zero. The instruction

RRC A

leaves the accumulator with the value 2H (0010B) and the carry flag set to logic one.

NOTE

The number of memory bank selected by SMB may change for different devices in the SAM48 product family.

SBC — Subtract With Carry

SBC dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Subtract indirect data memory from A with carry	1	1
	EA,RR	Subtract register pair (RR) from EA with carry	2	2
	RRb,EA	Subtract EA from register pair (RRb) with carry	2	2

Description: SBC subtracts the source and carry flag value from the destination operand, leaving the result in the destination. SBC sets the carry flag if a borrow is needed for the most significant bit; otherwise it clears the carry flag. The contents of the source are unaffected.

If the carry flag was set before the SBC instruction was executed, a borrow was needed for the previous step in multiple precision subtraction. In this case, the carry bit is subtracted from the destination along with the source operand.

Operand	Binary Code								Operation Notation
A,@HL	0	0	1	1	1	1	0	0	$C, A \leftarrow A - (HL) - C$
EA,RR	1	1	0	1	1	1	0	0	$C, EA \leftarrow EA - RR - C$
	1	1	0	0	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	$C, RRb \leftarrow RRb - EA - C$
	1	1	0	0	0	r2	r1	0	

Examples: 1. The extended accumulator contains the value 0C3H, register pair HL the value 0AAH, and the carry flag is set to "1":

```
SCF                ; C ← "1"
SBC    EA,HL        ; EA ← 0C3H - 0AAH - 1H, C ← "0"
JPS    XXX          ; Jump to XXX; no skip after SBC
```

2. If the extended accumulator contains the value 0C3H, register pair HL the value 0AAH, and the carry flag is cleared to "0":

```
RCF                ; C ← "0"
SBC    EA,HL        ; EA ← 0C3H - 0AAH - 0H = 19H, C ← "0"
JPS    XXX          ; Jump to XXX; no skip after SBC
```

SBC — Subtract with Carry

SBC (Continued)

Examples: 3. If SBC A,@HL is followed by an ADS A,#im, the SBC skips on 'no borrow' to the instruction immediately after the ADS. An 'ADS A,#im' instruction immediately after the 'SBC A,@HL' instruction does not skip even if an overflow occurs. This function is useful for decimal adjustment operations.

- a. 8 - 6 decimal addition (the contents of the address specified by the HL register is 6H):

```
RCF                ; C ← "0"
LD      A,#8H      ; A ← 8H
SBC    A,@HL       ; A ← 8H - 6H - C(0) = 2H, C ← "0"
ADS    A,#0AH      ; Skip this instruction because no borrow after SBC result
JPS    XXX
```

- b. 3 - 4 decimal addition (the contents of the address specified by the HL register is 4H):

```
RCF                ; C ← "0"
LD      A,#3H      ; A ← 3H
SBC    A,@HL       ; A ← 3H - 4H - C(0) = 0FH, C ← "1"
ADS    A,#0AH      ; No skip. A ← 0FH + 0AH = 9H
                ; (The skip function of 'ADS A,#im' is inhibited after a
                ; 'SBC A,@HL' instruction even if an overflow occurs.)
JPS    XXX
```

SBS — Subtract

SBS dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Subtract indirect data memory from A; skip on borrow	1	1 + S
	EA,RR	Subtract register pair (RR) from EA; skip on borrow	2	2 + S
	RRb,EA	Subtract EA from register pair (RRb); skip on borrow	2	2 + S

Description: The source operand is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. A skip is executed if a borrow occurs. The value of the carry flag is not affected.

Operand	Binary Code								Operation Notation
A,@HL	0	0	1	1	1	1	0	1	$A \leftarrow A - (HL)$; skip on borrow
EA,RR	1	1	0	1	1	1	0	0	$EA \leftarrow EA - RR$; skip on borrow
	1	0	1	1	1	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	$RRb \leftarrow RRb - EA$; skip on borrow
	1	0	1	1	0	r2	r1	0	

Examples:

- The accumulator contains the value 0C3H, register pair HL contains the value 0C7H, and the carry flag is cleared to logic zero:

```

RCF                                ; C ← "0"
SBS    EA,HL                        ; EA ← 0C3H - 0C7H
                                           ; SBS instruction skips on borrow,
                                           ; but carry flag value is not affected
JPS    XXX                          ; Skip because a borrow occurred
JPS    YYY                          ; Jump to YYY is executed

```

- The accumulator contains the value 0AFH, register pair HL contains the value 0AAH, and the carry flag is set to logic one:

```

SCF                                ; C ← "1"
SBS    EA,HL                        ; EA ← 0AFH - 0AAH
JPS    XXX                          ; Jump to XXX
                                           ; JPS was not skipped since no "borrow" occurred after
                                           ; SBS

```


SCF — Set Carry Flag

SCF

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Set carry flag to logic one	1	1

Description: The SCF instruction sets the carry flag to logic one, regardless of its previous value.

Operand	Binary Code							Operation Notation	
–	1	1	1	0	0	1	1	1	$C \leftarrow 1$

Example: If the carry flag is cleared to logic zero, the instruction

SCF

sets the carry flag to logic one.

SMB — Select Memory Bank

SMB n

Operation:	Operand	Operation Summary	Bytes	Cycles
	n	Select memory bank	2	2

Description: The SMB instruction sets the upper four bits of a 12-bit data memory address to select a specific memory bank. The constants 0, 1, and 15 are usually used as the SMB operand to select the corresponding memory bank. All references to data memory addresses fall within the following address ranges:

Please note that since data memory spaces differ for various devices in the SAM4 product family, the 'n' value of the SMB instruction will also vary.

Addresses	Register Areas	Bank	SMB
000H-01FH	Working registers	0	0
020H-0FFH	Stack and general-purpose registers		
N00H-NFFH	General-purpose registers	n	n
	Display registers	(n = 1-14)	(n = 1-14)
F80H-FFFH	I/O-mapped hardware registers	15	15

The enable memory bank (EMB) flag must always be set to "1" in order for the SMB instruction to execute successfully for memory bank 0 - 15.

Format	Binary Code								Operation Notation
n	1	1	0	1	1	1	0	1	SMB ← n
	0	1	0	0	d3	d2	d1	d0	

Example: If the EMB flag is set, the instruction

SMB 0

selects the data memory address range for bank 0 (000H-0FFH) as the working memory bank.

NOTES:

1. Number of memory bank selected by SMB may change for different device in the SAM48 product family.
2. After RESET, the SMB value is zero.

SRB — Select Register Bank

SRB n

Operation:	Operand	Operation Summary	Bytes	Cycles
	n	Select register bank	2	2

Description: The SRB instruction selects one of four register banks in the working register memory area. The constant value used with SRB is 0, 1, 2, or 3. The following table shows the effect of SRB settings:

ERB Setting	SRB Settings				Selected Register Bank
	3	2	1	0	
0	0	0	x	x	Always set to bank 0
1	0	0	0	0	Bank 0
			0	1	Bank 1
			1	0	Bank 2
			1	1	Bank 3

NOTE: 'x' = not applicable.

The enable register bank flag (ERB) must always be set for the SRB instruction to execute successfully for register banks 0, 1, 2, and 3. In addition, if the ERB value is logic zero, register bank 0 is always selected, regardless of the SRB value.

Operand	Binary Code								Operation Notation
n	1	1	0	1	1	1	0	1	SRB ← n (n = 0, 1, 2, 3)
	0	1	0	1	0	0	d1	d0	

Example: If the ERB flag is set, the instruction

SRB 3

selects register bank 3 (018H-01FH) as the working memory register bank.

SRET — Return from Subroutine and Skip

SRET

Operation:	Operand	Operation Summary	Bytes	Cycles
	–	Return from subroutine and skip	1	3 + S

Description: SRET is normally used to return to the previously executing procedure at the end of a subroutine that was initiated by a CALL or CALLS instruction. SRET skips the resulting address, which is generally the instruction immediately after the point at which the subroutine was called. Then, program execution continues from the resulting address and the contents of the location addressed by the stack pointer are popped into the program counter.

Operand	Binary Code								Operation Notation
–	1	1	1	0	0	1	0	1	PC13-8 ← (SP + 1) (SP) PC7-0 ← (SP + 3) (SP + 2) EMB,ERB ← (SP + 4) SP ← SP + 6

Example: If the stack pointer contains the value 0FAH and RAM locations 0FAH, 0FBH, 0FCH, and 0FDH contain the values 1H, 0H, 5H, and 2H, respectively, the instruction

SRET

leaves the stack pointer with the value 00H and the program returns to continue execution at location 0125H, then skips unconditionally.

During a return from subroutine, data is popped from the stack to the PC as follows:

SP →	(0FAH)	PC11 – PC8			
SP + 1	(0FBH)	0	0	PC13	PC12
SP + 2	(0FCH)	PC3 – PC0			
SP + 3	(0FDH)	PC7 – PC4			
SP + 4	(0FEH)	0	0	EMB	ERB
SP + 5	(0FFH)	0	0	0	0
SP + 6	(000H)				

STOP — Stop Operation

STOP

Operation:	Operand	Operation Summary	Bytes	Cycles
	-	Engage CPU stop mode	2	2

Description: The STOP instruction stops the system clock by setting bit 3 of the power control register (PCON) to logic one. When STOP executes, all system operations are halted with the exception of some peripheral hardware with special power-down mode operating conditions.

In application programs, a STOP instruction must be immediately followed by at least three NOP instructions. This ensures an adequate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructions are not used after STOP instruction, leakage current could be flown because of the floating state in the internal bus.

Operand	Binary Code								Operation Notation
-	1	1	1	1	1	1	1	1	PCON.3 ← 1
	1	0	1	1	0	0	1	1	

Example: Given that bit 3 of the PCON register is cleared to logic zero, and all systems are operational, the instruction sequence

```
STOP
NOP
NOP
NOP
```

sets bit 3 of the PCON register to logic one, stopping all controller operations (with the exception of some peripheral hardware). The three NOP instructions provide the necessary timing delay for clock stabilization before the next instruction in the program sequence is executed.

VENT — Load EMB, ERB, and Vector Address

VENTn dst

Operation:	Operand	Operation Summary	Bytes	Cycles
	EMB (0,1) ERB (0,1) ADR	Load enable memory bank flag (EMB) and the enable register bank flag (ERB) and program counter to vector address, then branch to the corresponding location.	2	2

Description: The VENT instruction loads the contents of the enable memory bank flag (EMB) and enable register bank flag (ERB) into the respective vector addresses. It then points the interrupt service routine to the corresponding branching locations. The program counter is loaded automatically with the respective vector addresses which indicate the starting address of the respective vector interrupt service routines.

The EMB and ERB flags should be modified using VENT before the vector interrupts are acknowledged. Then, when an interrupt is generated, the EMB and ERB values of the previous routine are automatically pushed onto the stack and then popped back when the routine is completed.

After the return from interrupt (IRET) you do not need to set the EMB and ERB values again. Instead, use BTR and BITS to clear these values in your program routine.

The starting addresses for vector interrupts and reset operations are pointed to by the VENTn instruction. These starting addresses must be located in ROM ranges 0000H-3FFFH. Generally, the VENTn instructions are coded starting at location 0000H.

The format for VENT instructions is as follows:

VENTn d1,d2,ADDR

EMB ← d1 ("0" or "1")

ERB ← d2 ("0" or "1")

PC ← ADDR (address to branch)

n = device-specific module address code (n = 0-n)

Operand	Binary Code								Operation Notation
EMB (0,1) ERB (0,1) ADR	E	E	a13	a12	a11	a10	a9	a8	ROM (2 x n) 7-6 → EMB, ERB ROM (2 x n) 5-4 → PC13, PC12 ROM (2 x n) 3-0 → PC12-8 ROM (2 x n + 1) 7-0 → PC7-0 (n = 0, 1, 2, 3, 4, 5, 6, 7)
	M	R							
	B	B							
	a7	a6	a5	a4	a3	a2	a1	a0	

VENT — Load EMB, ERB, and Vector Address

VENTn (Continued)

Example: The instruction sequence

```
ORG      0000H
VENT0    1,0,RESET
VENT1    0,1,INTA
VENT2    0,1,INTB
VENT3    0,1,INTC
VENT4    0,1,INTD
VENT5    0,1,INTE
VENT6    0,1,INTF
VENT7    0,1,INTG
```

causes the program sequence to branch to the RESET routine labeled RESET, setting EMB to "1" and ERB to "0" when RESET is activated. When a basic timer interrupt is generated, VENT1 causes the program to branch to the basic timer's interrupt service routine, INTA, and to set the EMB value to "0" and the ERB value to "1". VENT2 then branches to INTB, VENT3 to INTC, and so on, setting the appropriate EMB and ERB values.

NOTE: The number of VENTn interrupt names used in the examples above may change for different devices in the SAM48 product family.

XCH — Exchange A or EA with Nibble or Byte

XCH dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,DA	Exchange A and data memory contents	2	2
	A,Ra	Exchange A and register (Ra) contents	1	1
	A,@RRa	Exchange A and indirect data memory	1	1
	EA,DA	Exchange EA and direct data memory contents	2	2
	EA,RRb	Exchange EA and register pair (RRb) contents	2	2
	EA,@HL	Exchange EA and indirect data memory contents	2	2

Description: The instruction XCH loads the accumulator with the contents of the indicated destination variable and writes the original contents of the accumulator to the source.

Operand	Binary Code								Operation Notation
A,DA	0	1	1	1	1	0	0	1	A ↔ DA
	a7	a6	a5	a4	a3	a2	a1	a0	
A,Ra	0	1	1	0	1	r2	r1	r0	A ↔ Ra
A,@RRa	0	1	1	1	1	i2	i1	i0	A ↔ (RRa)
EA,DA	1	1	0	0	1	1	1	1	A ↔ DA, E ↔ DA + 1
	a7	a6	a5	a4	a3	a2	a1	a0	
EA,RRb	1	1	0	1	1	1	0	0	EA ↔ RRb
	1	1	1	0	0	r2	r1	0	
EA,@HL	1	1	0	1	1	1	0	0	A ↔ (HL), E ↔ (HL + 1)
	0	0	0	0	0	0	0	1	

Example: Double register HL contains the address 20H. The accumulator contains the value 3FH (00111111B) and internal RAM location 20H the value 75H (01110101B). The instruction

```
XCH EA,@HL
```

leaves RAM location 20H with the value 3FH (00111111B) and the extended accumulator with the value 75H (01110101B).

XCHD — Exchange and Decrement

XCHD dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Exchange A and data memory contents; decrement contents of register L and skip on borrow	1	2 + S

Description: The instruction XCHD exchanges the contents of the accumulator with the RAM location addressed by register pair HL and then decrements the contents of register L. If the content of register L is 0FH, the next instruction is skipped. The value of the carry flag is not affected.

Operand	Binary Code								Operation Notation
A,@HL	0	1	1	1	1	0	1	1	A ↔ (HL), then L ← L-1; skip if L = 0FH

Example: Register pair HL contains the address 20H and internal RAM location 20H contains the value 0FH:

```

LD HL,#20H
LD A,#0H
XCHD    A,@HL    ; A ← 0FH and L ← L - 1, (HL) ← "0"
JPS     XXX      ; Skipped since a borrow occurred
JPS     YYY      ; H ← 2H, L ← 0FH

YYY     XCHD     A,@HL    ; (2FH) ← 0FH, A ← (2FH), L ← L - 1 = 0EH
      •
      •
      •

```

The 'JPS YYY' instruction is executed since a skip occurs after the XCHD instruction.

XCHI — Exchange and Increment

XCHI dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,@HL	Exchange A and data memory contents; increment contents of register L and skip on overflow	1	2 + S

Description: The instruction XCHI exchanges the contents of the accumulator with the RAM location addressed by register pair HL and then increments the contents of register L. If the content of register L is 0H, a skip is executed. The value of the carry flag is not affected.

Operand	Binary Code								Operation Notation
A,@HL	0	1	1	1	1	0	1	0	$A \leftrightarrow (HL)$, then $L \leftarrow L+1$; skip if $L = 0H$

Example: Register pair HL contains the address 2FH and internal RAM location 2FH contains 0FH:

```

LD    HL,#2FH
LD    A,#0H
XCHI  A,@HL    ;    A ← 0FH and L ← L + 1 = 0, (HL) ← "0"
JPS   XXX      ;    Skipped since an overflow occurred
JPS   YYY      ;    H ← 2H, L ← 0H

YYY   XCHI     A,@HL ; (20H) ← 0FH, A ← (20H), L ← L + 1 = 1H
      •
      •
      •

```

The 'JPS YYY' instruction is executed since a skip occurs after the XCHI instruction.

XOR — Logical Exclusive OR

XOR dst,src

Operation:	Operand	Operation Summary	Bytes	Cycles
	A,#im	Exclusive-OR immediate data to A	2	2
	A,@HL	Exclusive-OR indirect data memory to A	1	1
	EA,RR	Exclusive-OR register pair (RR) to EA	2	2
	RRb,EA	Exclusive-OR register pair (RRb) to EA	2	2

Description: XOR performs a bitwise logical XOR operation between the source and destination variables and stores the result in the destination. The source contents are unaffected.

Operand	Binary Code								Operation Notation
A,#im	1	1	0	1	1	1	0	1	A ← A XOR im
	0	0	1	1	d3	d2	d1	d0	
A,@HL	0	0	1	1	1	0	1	1	A ← A XOR (HL)
EA,RR	1	1	0	1	1	1	0	0	EA ← EA XOR (RR)
	0	0	1	1	0	r2	r1	0	
RRb,EA	1	1	0	1	1	1	0	0	RRb ← RRb XOR EA
	0	0	1	1	0	r2	r1	0	

Example: If the extended accumulator contains 0C3H (11000011B) and register pair HL contains 55H (01010101B), the instruction

```
XOR  EA,HL
```

leaves the value 96H (10010110B) in the extended accumulator.

NOTES

6

OSCILLATOR CIRCUITS

OVERVIEW

The S3C72Q5 microcontroller has two oscillator circuits: a main-system clock circuit, and a sub-system clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. Specifically, a clock pulse is required by the following peripheral modules:

- LCD controller
- Basic timer
- Timer/counter 0
- Timer/counter 1
- Watch timer
- Clock output circuit

CPU Clock Notation

In this document, the following notation is used for descriptions of the CPU clock:

- fx Main-system clock
- fxt Sub-system clock
- fxx Selected system clock

Clock Control Registers

When the system clock mode control register SCMOD and the power control register PCON registers are both cleared to zero after RESET, the normal CPU operating mode is enabled, a main-system clock of $fx/64$ is selected, and main-system clock oscillation is initiated.

The power control register, PCON, is used to select normal CPU operating mode or one of two power-down modes — stop or idle. Bits 3 and 2 of the PCON register can be manipulated by a STOP or IDLE instruction to engage stop or idle power-down mode.

The system clock mode control register, SCMOD, lets you select the *main-system clock (fx)* or a *sub-system clock (fxt)* as the CPU clock and to start (or stop) main-system clock oscillation. The resulting clock source, either main-system clock or sub-system clock, is referred to as the *selected system clock (fxx)*.

The main-system clock is selected and oscillation started when all SCMOD bits are cleared to logic zero. By setting SCMOD.3 and SCMOD.0 to different values, you can select a sub-system clock source and start or stop main-system clock oscillation. To stop main-system clock oscillation, you must use the STOP instruction (assuming the main-system clock is selected) or manipulate SCMOD.3 to “1” (assuming the sub-system clock is selected).

The main-system clock frequencies can be divided by 4, 8, or 64 and a sub-system clock frequencies can only be divided by 4. By manipulating PCON bits 1 and 0, you select one of the following frequencies as the CPU Clock, $fx/4$, $fxt/4$, $fx/8$, $fx/64$.

Using a Sub-system Clock

If a sub-system clock is being used as the selected system clock, the idle power-down mode can be initiated by executing an IDLE instruction. Since the sub-system clock source cannot be stopped internally, you cannot, however, use a STOP instruction to enable the stop power-down mode.

The watch timer, buzzer and LCD display operate normally with a sub-system clock source, since they operate at very low speed (as low as 122 μ s at 32.768 kHz) and with very low power consumption.

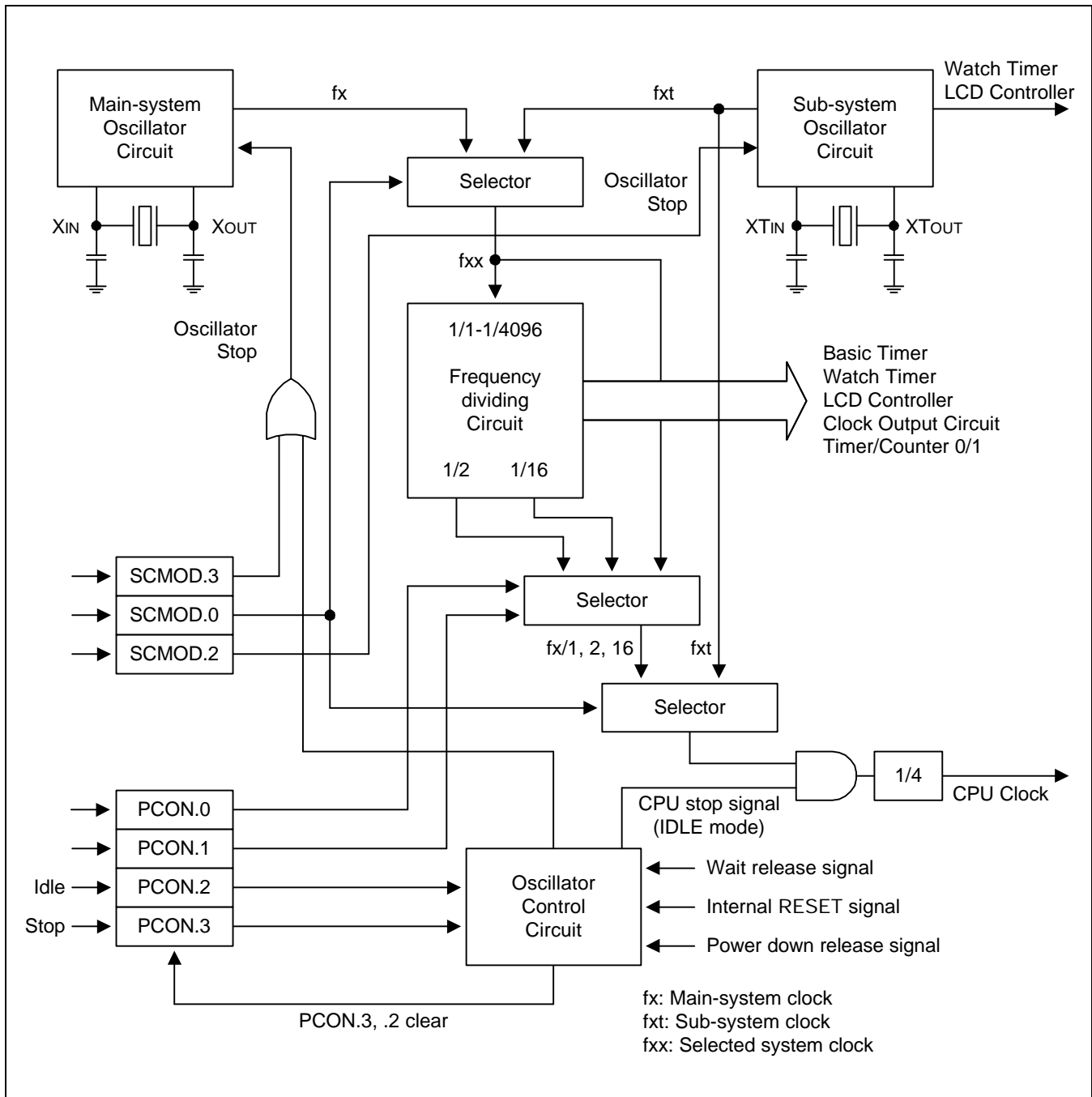


Figure 6-1. Clock Circuit Diagram

MAIN-SYSTEM OSCILLATOR CIRCUITS

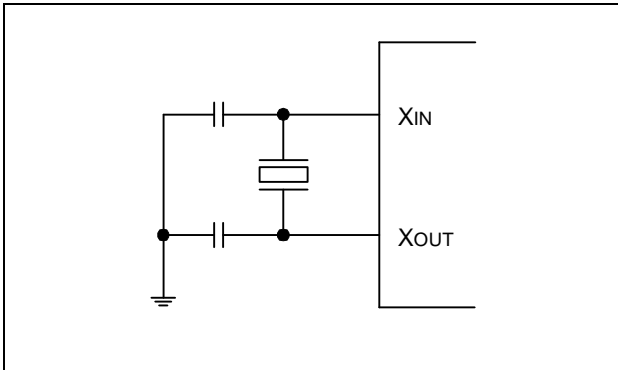


Figure 6-2. Crystal/Ceramic Oscillator

SUB-SYSTEM OSCILLATOR CIRCUITS

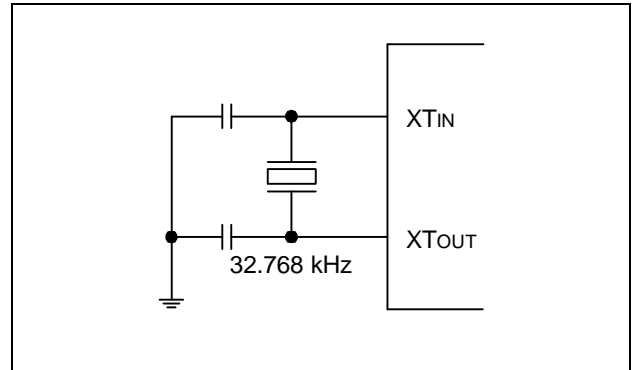


Figure 6-5. Crystal/Ceramic Oscillator

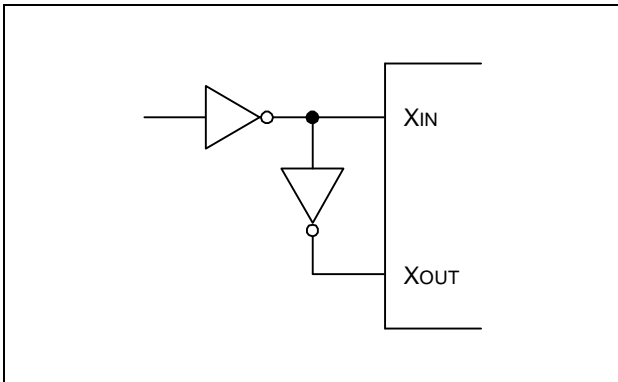


Figure 6-3. External Oscillator

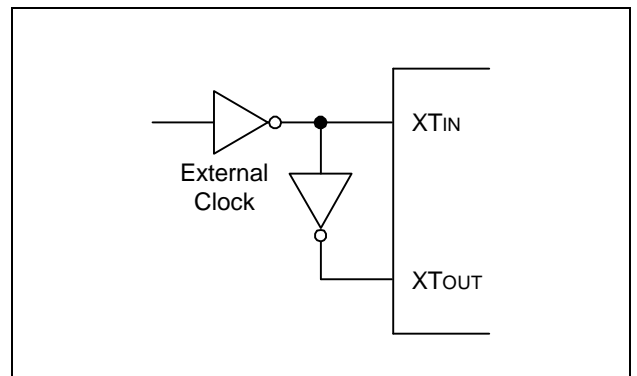


Figure 6-6. External Oscillator

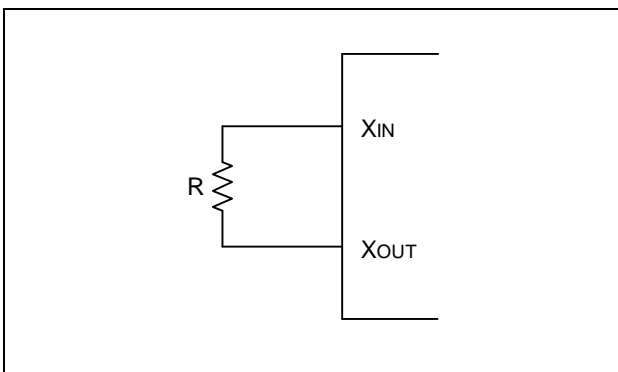


Figure 6-4. RC Oscillator

POWER CONTROL REGISTER (PCON)

The power control register, PCON, is a 4-bit register that is used to select the CPU clock frequency and to control CPU operating and power-down modes. PCON can be addressed directly by 4-bit write instructions or indirectly by the instructions IDLE and STOP.



PCON bits 3 and 2 are addressed by the STOP and IDLE instructions, respectively, to engage the idle and stop power-down modes. Idle and stop modes can be initiated by these instruction despite the current value of the enable memory bank flag (EMB). PCON bits 1 and 0 are used to select a specific system clock frequency. There are two basic choices:

- Main-system clock (fx) or sub-system clock (fxt);
- Divided fx/4, 8, 64 or fxt/4 clock frequency.

PCON.1 and PCON.0 settings are also connected with the system clock mode control register, SCMOD. If SCMOD.0 = "0" the main-system clock is always selected by the PCON.1 and PCON.0 setting; if SCMOD.0 = "1" the sub-system clock is selected.

RESET sets PCON register values (and SCMOD) to logic zero: SCMOD.3 and SCMOD.0 select the main-system clock (fx) and start clock oscillation; PCON.1 and PCON.0 divide the selected fx frequency by 64, and PCON.3 and PCON.2 enable normal CPU operating mode.

Table 6-1. Power Control Register (PCON) Organization

PCON Bit Settings		Resulting CPU Operating Mode	
PCON.3	PCON.2		
0	0	Normal CPU operating mode	
0	1	Idle power-down mode	
1	0	Stop power-down mode	

PCON Bit Settings		Resulting CPU Clock Frequency	
PCON.1	PCON.0	If SCMOD.0 = "0"	If SCMOD.0 = "1"
0	0	fx/64	fxt/4
1	0	fx/8	
1	1	fx/4	

PROGRAMMING TIP — Setting the CPU Clock

To set the CPU clock to 0.95 μs at 4.19 MHz:

```

BITS      EMB
SMB      15
LD        A,#3H
LD        PCON,A
    
```


INSTRUCTION CYCLE TIMES

The unit of time that equals one machine cycle varies depending on whether the main-system clock (f_x) or a sub-system clock (f_{xt}) is used, and on how the oscillator clock signal is divided (by 4, 8, or 64). Table 6-2 shows corresponding cycle times in microseconds.

Table 6-2. Instruction Cycle Times for CPU Clock Rates

Selected CPU Clock	Resulting Frequency	Oscillation Source	Cycle Time (μsec)
$f_x/64$	65.5 kHz	$f_x = 4.19 \text{ MHz}$	15.3
$f_x/8$	524.0 kHz		1.91
$f_x/4$	1.05 MHz		0.95
$f_{xt}/4$	8.19 kHz	$f_{xt} = 32.768 \text{ kHz}$	122.0

SYSTEM CLOCK MODE REGISTER (SCMOD)

The system clock mode register, SCMOD, is a 4-bit register that is used to select the CPU clock and to control main and sub-system clock oscillation. The SCMOD is mapped to the RAM address FB7H.

The main clock oscillation is stopped by setting SCMOD.3 when the clock source is subsystem clock and subsystem clock can be stopped by setting SCMOD.2 when the clock source is main system clock. SCMOD.0, SCMOD.3 cannot be simultaneously modified.

The subsystem clock is stopped only by setting SCMOD.2, and PCON which revokes stop mode cannot stop the subsystem clock. The stop of subsystem clock is released by RESET when the selected system clock is main system clock or subsystem clock and is released by setting SCMOD.2 when the selected system clock is main system clock.

RESET clears all SCMOD values to logic zero, selecting the main system clock (fx) as the CPU clock and starting clock oscillation. The reset value of the SCMOD is "0"

SCMOD.0, SCMOD.2, and SCMOD.3 bits can be manipulated by 1-bit write instructions (In other words, SCMOD.0, SCMOD.2, and SCMOD.3 cannot be modified simultaneously by a 4-bit write).

Bit 1 is always logic zero.

FB7H	SCMOD.3	SCMOD.2	"0"	SCMOD.0
------	---------	---------	-----	---------

A subsystem clock (fxt) can be selected as the system clock by manipulating the SCMOD.3 and SCMOD.0 bit settings. If SCMOD.3 = "0" and SCMOD.0 = "1", the subsystem clock is selected and main system clock oscillation continues. If SCMOD.3 = "1" and SCMOD.0 = "1", fxt is selected, but main system clock oscillation stops.

Even if you have selected fx as the CPU clock, setting SCMOD.3 to "1" will stop main system clock oscillation, and malfunction may be occurred. To operate safely, main system clock should be stopped by a stop instruction is main system clock mode.

Table 6-3. System Clock Mode Register (SCMOD) Organization

SCMOD Register Bit Settings		Resulting Clock Selection	
SCMOD.3	SCMOD.0	CPU Clock Source	fx Oscillation
0	0	fx	On
0	1	fxt	On
1	1	fxt	Off

SCMOD.2	Sub-oscillation on/off
0	Enable sub system clock
1	Disable sub system clock

NOTE: You can use SCMOD.2 as follows (ex; after data bank was used, a few minutes have passed):
 Main operation → sub-operation → sub-idle (LCD on, after a few minutes later without any external input) → sub-operation → main operation → SCMOD.2 = 1 → main stop mode (LCD off).

Table 6-4. Main Oscillation Stop Mode

Mode	Condition	Method to issue Osc Stop	Osc Stop Release Source ⁽²⁾
Main Oscillation STOP Mode	Main oscillator runs. Sub oscillator runs (stops). System clock is the main oscillation clock.	STOP instruction: Main oscillator stops. CPU is in idle mode. Sub oscillator still runs (stops).	Interrupt and RESET: After releasing stop mode, main oscillation starts and oscillation stabilization time is elapsed. And then the CPU operates. Oscillation stabilization time is $1 / \{256 \times \text{BT clock (fx)}\}$.
		When SCMOD.3 is set to "1" ⁽¹⁾ , main oscillator stops, halting the CPU operation. Sub oscillator still runs (stops).	RESET: Interrupt can't start the main oscillation. Therefore, the CPU operation can never be restarted.
	Main oscillator runs. Sub oscillator runs. System clock is the sub oscillation clock.	STOP instruction ⁽¹⁾ : Main oscillator stops. CPU is in idle mode. Sub oscillator still runs (stops). Sub oscillator still runs.	BT overflow, interrupt, and RESET: After the overflow of basic timer [$1 / \{256 \times \text{BT clock (fxt)}\}$], CPU operation and main oscillation automatically start.
		When SCMOD.3 is set to "1", main oscillator stops. The CPU, however, would still operate. Sub oscillator still runs.	Set SCMOD.3 to "0" or RESET
Sub Oscillation STOP Mode	Main oscillator runs. Sub oscillator runs. System clock is the main oscillation clock.	When SCMOD.2 to "1", sub oscillator stops, while main oscillator and the CPU would still operate.	Set SCMOD.2 to "0" or RESET
	Main oscillator runs (stops). Sub oscillator runs. System clock is the sub oscillation clock.	When SCMOD.2 to "1", sub oscillator stops, halting the CPU operation. Main oscillator still runs (stops).	RESET

NOTES:

1. This mode must not be used.
2. Oscillation stabilization time by interrupt is $1/(256 \times \text{BT clocks})$. Oscillation stabilization time by a reset is 31.3ms at 4.19Mhz, main oscillation clock.

SWITCHING THE CPU CLOCK

Together, bit settings in the power control register, PCON, and the system clock mode register, SCMOD, determine whether a main system or a subsystem clock is selected as the CPU clock, and also how this frequency is to be divided. This makes it possible to switch dynamically between main and subsystem clocks and to modify operating frequencies.

SCMOD.3, SCMOD.2, and SCMOD.0 select the main system clock (fx) or a subsystem clock (fxt) and start or stop main system and sub system clock oscillation. PCON.1 and PCON.0 control the frequency divider circuit, and divide the selected fx clock by 4, 8, or 64, or fxt clock by 4.

NOTE

A clock switch operation does not go into effect immediately when you make the SCMOD and PCON register modifications — the previously selected clock continues to run for a certain number of machine cycles.

For example, you are using the default CPU clock (normal operating mode and a main system clock of fx/64) and you want to switch from the fx clock to a subsystem clock and to stop the main system clock. To do this, you first need to set SCMOD.0 to "1". This switches the clock from fx to fxt but allows main system clock oscillation to continue. Before the switch actually goes into effect, a certain number of machine cycles must elapse. After this time interval, you can then disable main system clock oscillation by setting SCMOD.3 to "1".

This same 'stepped' approach must be taken to switch from a subsystem clock to the main system clock: first, clear SCMOD.3 to "0" to enable main system clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main system clock by clearing all SCMOD values to logic zero.

Following a RESET, CPU operation starts with the lowest main system clock frequency of 15.3 μs at 4.19 MHz after the standard oscillation stabilization interval of 31.3 ms has elapsed. Table 6-5 details the number of machine cycles that must elapse before a CPU clock switch modification goes into effect.

Table 6-5. Elapsed Machine Cycles During CPU Clock Switch

AFTER		SCMOD.0 = 0						SCMOD.0 = 1	
		PCON.1 = 0		PCON.0 = 0		PCON.1 = 1			PCON.0 = 1
BEFORE	PCON.1 = 0	N/A		1 MACHINE CYCLE		1 MACHINE CYCLE		N/A	
	PCON.0 = 0	8 MACHINE CYCLES		N/A		1 MACHINE CYCLES		N/A	
SCMOD.0 = 0	PCON.1 = 1	16 MACHINE CYCLES		1 MACHINE CYCLES		N/A		fx / 4fxt	
	PCON.0 = 1	N/A		N/A		1MACHINE CYCLES		N/A	
SCMOD.0 = 1		N/A		N/A		1MACHINE CYCLES		N/A	

NOTES:

1. Even if oscillation is stopped by setting SCMOD.3 during main system clock operation, the stop mode is not entered.
2. Since the X_{IN} input is connected internally to V_{SS} to avoid current leakage due to the crystal oscillator in stop mode, do not set SCMOD.3 to "1" or do not use stop instruction when an external clock is used as the main system clock.
3. When the system clock is switched to the subsystem clock, it is necessary to disable any interrupts which may occur during the time intervals shown in Table 6-5.
4. 'N/A' means 'not available'.
5. fx: Main-system clock, fxt: Sub-system clock. When fx is 4.19 MHz, and fxt is 32.768 kHz.

 **PROGRAMMING TIP — Switching Between Main-system and Sub-system Clock**

1. Switch from the main-system clock to the sub-system clock:

MA2SUB	BITS	SCMOD.0	; Switches to sub-system clock
	CALL	DLY80	; Delay 80 machine cycles
	BITS	SCMOD.3	; Stop the main-system clock
	RET		
DLY80	LD	A,#0FH	
DEL1	NOP		
	NOP		
	DECS	A	
	JR	DEL1	
	RET		

2. Switch from the sub-system clock to the main-system clock:

SUB2MA	BITR	SCMOD.3	; Start main-system clock oscillation
	CALL	DLY80	; Delay 80 machine cycles
	CALL	DLY80	
	BITR	SCMOD.0	; Switch to main-system clock
	RET		

CLOCK OUTPUT MODE REGISTER (CLMOD)

The clock output mode register, CLMOD, is a 4-bit register that is used to enable or disable clock output to the CLO pin and to select the CPU clock source and frequency. CLMOD is addressable by 4-bit write instructions only.

FD0H	CLMOD.3	"0"	CLMOD.1	CLMOD.0
------	---------	-----	---------	---------

RESET clears CLMOD to logic zero, which automatically selects the CPU clock as the clock source (without initiating clock oscillation), and disables clock output.

CLMOD.3 is the enable/disable clock output control bit; CLMOD.1 and CLMOD.0 are used to select one of four possible clock sources and frequencies: normal CPU clock, $f_{xx}/8$, $f_{xx}/16$, or $f_{xx}/64$.

Table 6-6. Clock Output Mode Register (CLMOD) Organization

CLMOD Bit Settings		Resulting Clock Output	
CLMOD.1	CLMOD.0	Clock Source	Frequency
0	0	CPU clock ($f_x/4$, $f_x/8$, $f_x/64$, $f_{xt}/4$)	1.05MHz, 524kHz, 65.5kHz or 8.19kHz
0	1	$f_{xx}/8$	523.8 kHz
1	0	$f_{xx}/16$	261.9 kHz
1	1	$f_{xx}/64$	65.5 kHz

CLMOD.3	Result of CLMOD.3 Setting
0	Disable clock output at the CLO pin.
1	Enable clock output at the CLO pin.

NOTES:

1. f_x : Main-system clock
2. f_{xt} : Sub-system clock
3. Frequencies assume that f_{xx} , $f_x = 4,19\text{MHz}$, and $f_{xt} = 32.768\text{kHz}$

CLOCK OUTPUT CIRCUIT

The clock output circuit, used to output clock pulses to the CLO pin, has the following components:

- 4-bit clock output mode register (CLMOD)
- Clock selector
- Port mode flag
- CLO output pin

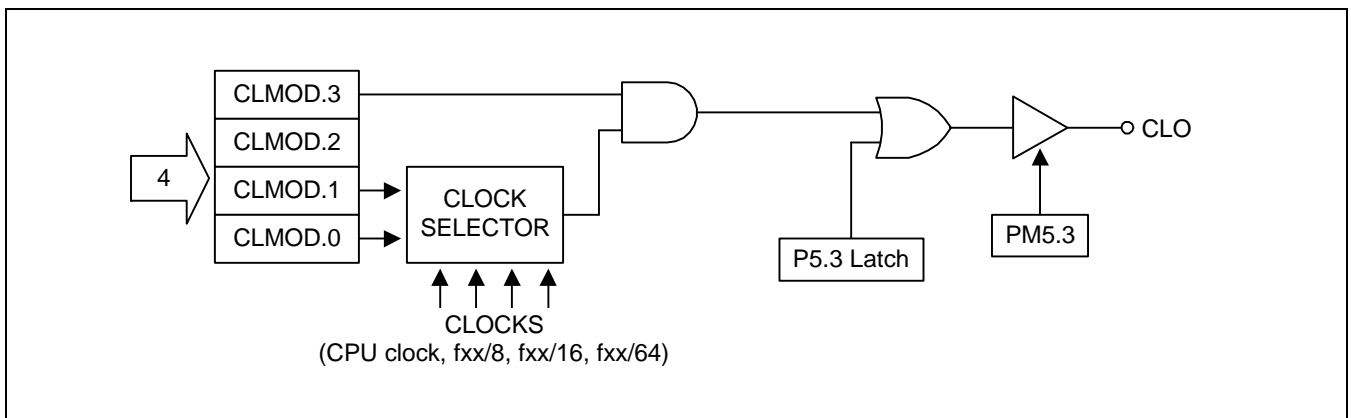


Figure 6-7. CLO Output Pin Circuit Diagram

CLOCK OUTPUT PROCEDURE

The procedure for outputting clock pulses to the CLO pin may be summarized as follows:

1. Disable clock output by clearing CLMOD.3 to logic zero.
2. Set the clock output frequency (CLMOD.1, CLMOD.0).
3. Load a "0" to the output latch of the CLO pin.
4. Set the port mode flag to output mode.
5. Enable clock output by setting CLMOD.3 to logic one.

PROGRAMMING TIP — CPU Clock Output to the CLO Pin

To output the CPU clock to the CLO pin:

BITS	EMB	
SMB	15	
LD	EA,#80H	
LD	PMG1,EA	; P5.3 ← Output mode
BITR	P5.3	; Clear the CLO pin output latch
LD	A,#8H	
LD	CLMOD,A	

NOTES

7 INTERRUPTS

OVERVIEW

The S3C72Q5's interrupt control circuit has five functional components:

- Interrupt enable flags (IEx)
- Interrupt request flags (IRQx)
- Interrupt master enable register (IME)
- Interrupt priority register (IPR)
- Power-down release signal circuit

Three kinds of interrupts are supported:

- Internal interrupts generated by on-chip processes
- External interrupts generated by external peripheral devices
- Quasi-interrupts used for edge detection and as clock sources

Table 7-1. Interrupt Types and Corresponding Port Pin(s)

Interrupt Type	Interrupt Name	Corresponding Port Pins
External interrupts	INT0, INT1, INTP0	P4.2, P4.3, P0(K0-K3)
Internal interrupts	INTB, INTT0, INTT1	Not applicable
Quasi-interrupts	INT2	P6,P7(KS0–KS7)
	INTW	Not applicable

VECTORED INTERRUPTS

Interrupt requests may be processed as vectored interrupts in hardware, or they can be generated by program software. A vectored interrupt is generated when the following flags and register settings, corresponding to the specific interrupt (INTn) are set to logic one:

- Interrupt enable flag (IEx)
- Interrupt master enable flag (IME)
- Interrupt request flag (IRQx)
- Interrupt status flags (IS0, IS1)
- Interrupt priority register (IPR)

If all conditions are satisfied for the execution of a requested service routine, the start address of the interrupt is loaded into the program counter and the program starts executing the service routine from this address.

EMB and ERB flags for RAM memory banks and registers are stored in the vector address area of the ROM during interrupt service routines. The flags are stored at the beginning of the program with the VENT instruction. The initial flag values determine the vectors for resets and interrupts. Enable flag values are saved during the main routine, as well as during service routines. Any changes that are made to enable flag values during a service routine are not stored in the vector address.

When an interrupt occurs, the EMB and ERB values before the interrupt is initiated are saved along with the program status word (PSW), and the EMB and the ERB flag for the interrupt are fetched from the respective vector address. Then, if necessary, you can modify the enable flags during the interrupt service routine. When the interrupt service routine is returned to the main routine by the IRET instruction, the original values saved in the stack are restored and the main program continues program execution with these values.

Software-Generated Interrupts

To generate an interrupt request from software, the program manipulates the appropriate IRQx flag. When the interrupt request flag value is set, it is retained until all other conditions for the vectored interrupt have been met, and the service routine can be initiated.

Multiple Interrupts

By manipulating the two interrupt status flags (IS0 and IS1), you can control service routine initialization and thereby process multiple interrupts simultaneously.

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have executed successfully, you can restore the register contents from the stack to working memory by using the POP instruction.

Power-Down Mode Release

An interrupt can be used to release power-down mode (stop or idle). Interrupts for power-down mode release are initiated by setting the corresponding interrupt enable flag. Even if the IME flag is cleared to zero, power-down mode will be released by an interrupt request signal when the interrupt enable flag has been set. In such cases, the interrupt routine will not be executed since IME = "0".

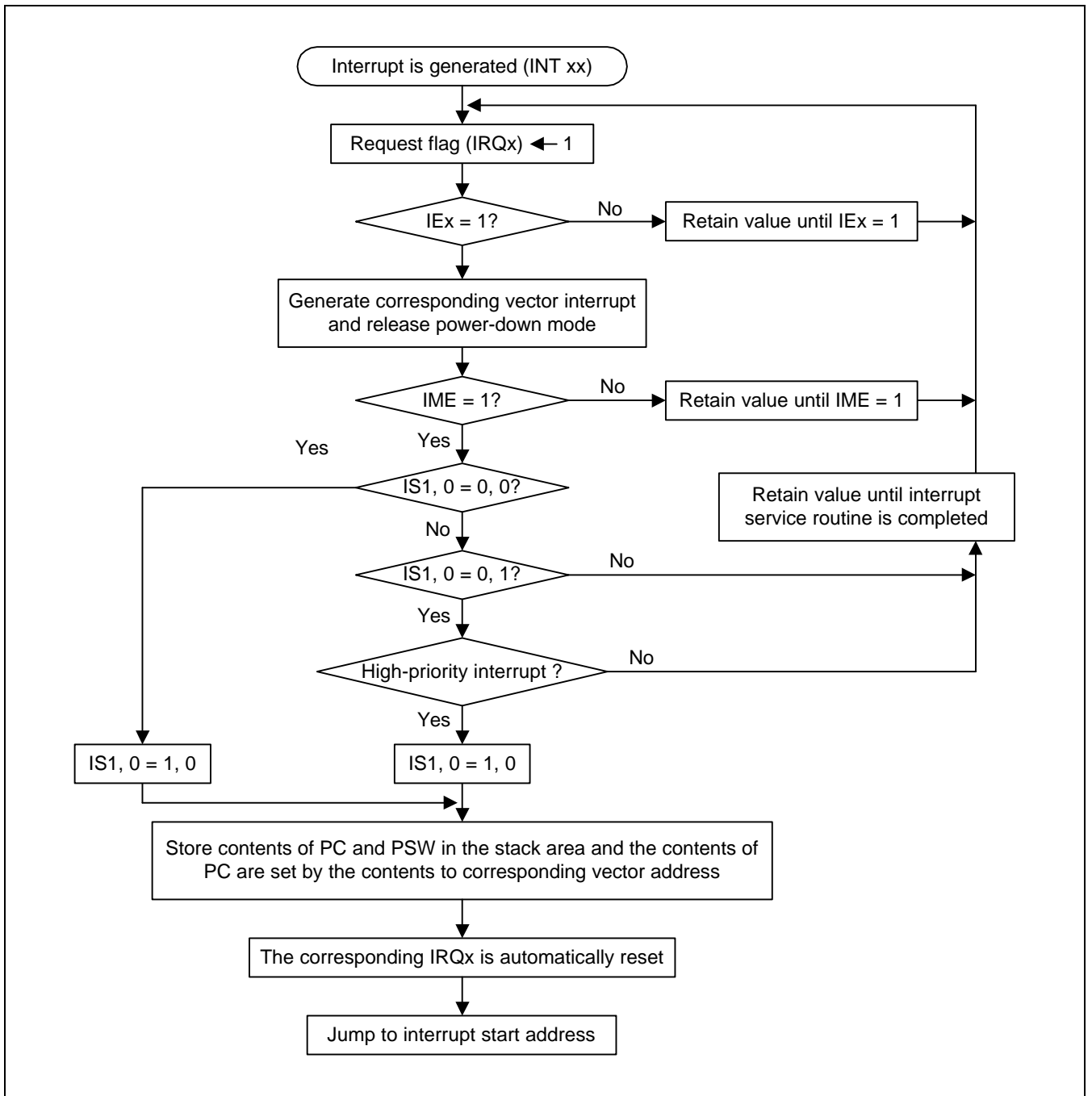


Figure 7-1. Interrupt Execution Flowchart

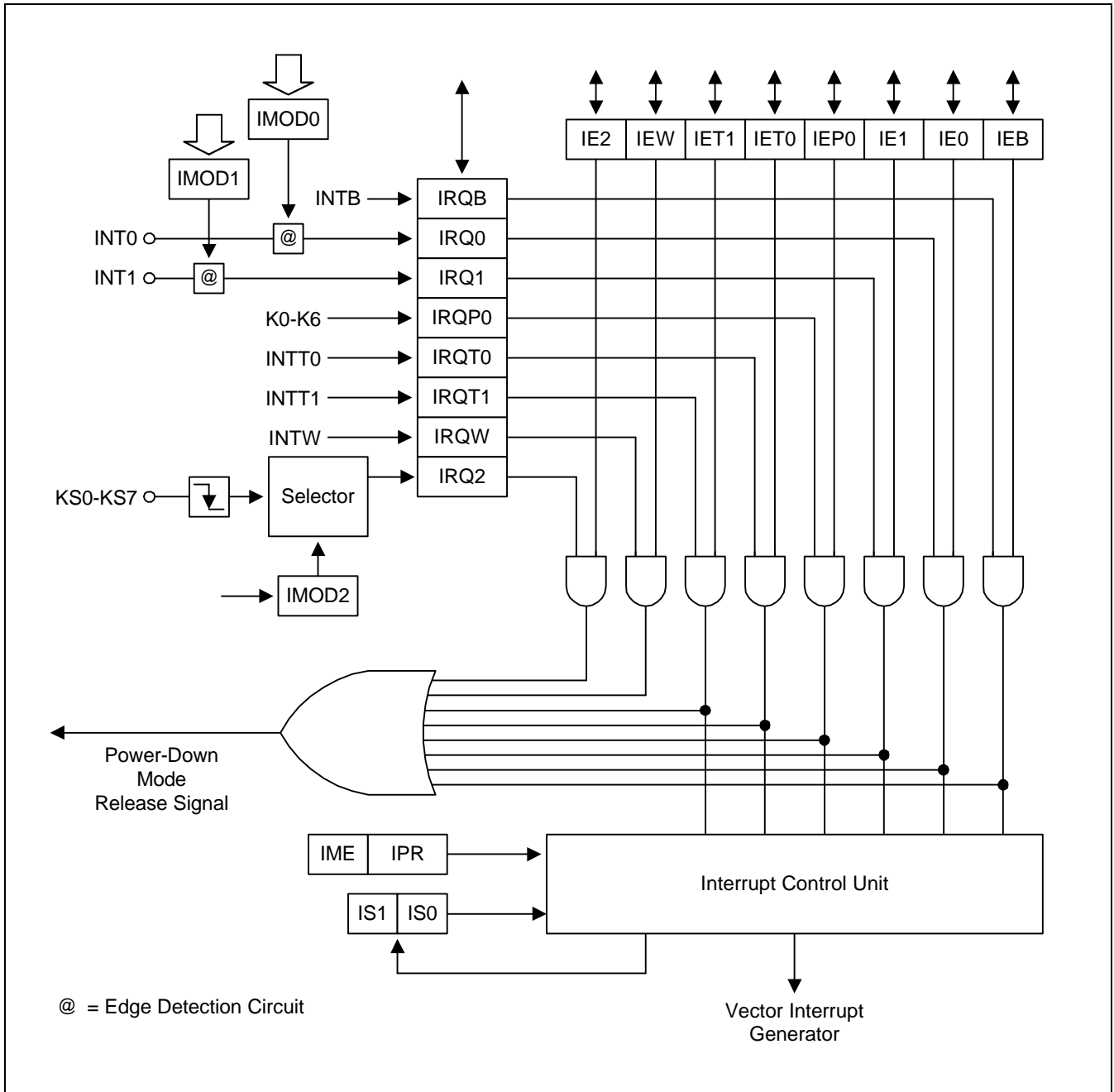


Figure 7-2. Interrupt Control Circuit Diagram

MULTIPLE INTERRUPTS

The interrupt controller can serve multiple interrupts in two ways: as two-level interrupts, where either all interrupt requests or only those of highest priority are serviced, or as multi-level interrupts, when the interrupt service routine for a lower-priority request is accepted during the execution of a higher priority routine.

Two-Level Interrupt Handling

Two-level interrupt handling is the standard method for processing multiple interrupts. When the IS1 and IS0 bits of the PSW (FB0H.3 and FB0H.2, respectively) are both logic zero, program execution mode is normal and all interrupt requests are serviced (see Figure 7-3).

Whenever an interrupt request is accepted, IS1 and IS0 are incremented by one, and the values are stored in the stack along with the other PSW bits. After the interrupt routine has been serviced, the modified IS1 and IS0 values are automatically restored from the stack by an IRET instruction.

IS0 and IS1 can be manipulated directly by 1-bit write instructions, regardless of the current value of the enable memory bank flag (EMB). Before you can modify an interrupt service flag, however, you must first disable interrupt processing with a DI instruction.

When IS1 = "0" and IS0 = "1", all interrupt service routines are inhibited except for the highest priority interrupt currently defined by the interrupt priority register (IPR).

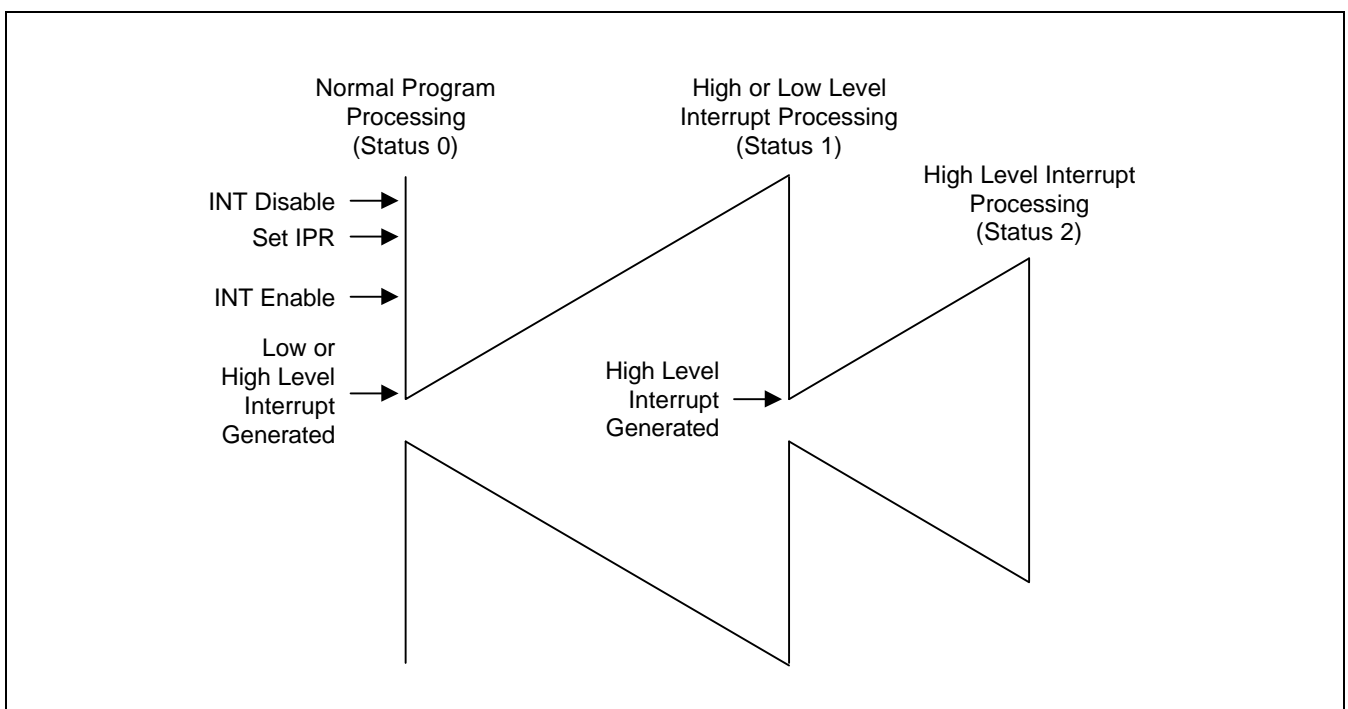


Figure 7-3. Two-Level Interrupt Handling

Multi-Level Interrupt Handling

With multi-level interrupt handling, a lower-priority interrupt request can be executed while a high-priority interrupt is being serviced. This is done by manipulating the interrupt status flags, IS0 and IS1 (see Table 7-2).

When an interrupt is requested during normal program execution, interrupt status flags IS0 and IS1 are set to "1" and "0", respectively. This setting allows only highest-priority interrupts to be serviced. When a high-priority request is accepted, both interrupt status flags are then cleared to "0" by software so that a request of any priority level can be serviced. In this way, the high- and low-priority requests can be serviced in parallel (see Figure 7-4).

Table 7-2. IS1 and IS0 Bit Manipulation for Multi-Level Interrupt Handling

Process Status	Before INT		Effect of ISx Bit Setting	After INT ACK	
	IS1	IS0		IS1	IS0
0	0	0	All interrupt requests are serviced.	0	1
1	0	1	Only high-priority interrupts as determined by the current settings in the IPR register are serviced.	1	0
2	1	0	No additional interrupt requests will be serviced.	–	–
–	1	1	Value undefined	–	–

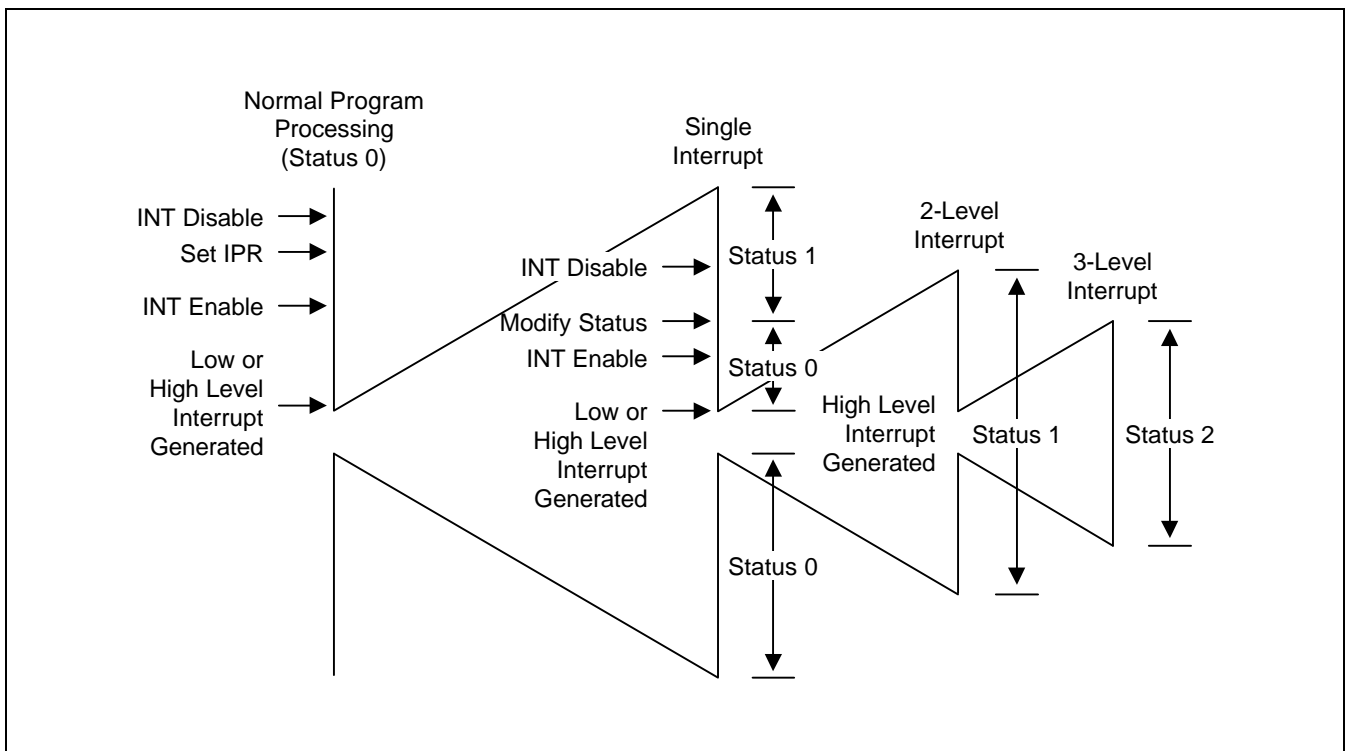


Figure 7-4. Multi-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

The 4-bit interrupt priority register (IPR) is used to control multi-level interrupt handling and its reset value is logic zero. Before the IPR can be modified by 4-bit write instructions, all interrupts must first be disabled by a DI instruction.

FB2H	IME	IPR.2	IPR.1	IPR.0
------	-----	-------	-------	-------

By manipulating the IPR settings, you can choose to process all interrupt requests with the same priority level, or you can select one type of interrupt for high-priority processing. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

Table 7-3. Standard Interrupt Priorities

Interrupt	Default Priority
INTB	1
INT0	2
INT1	3
INTP0	4
INTT0	5
INTT1	6

The MSB of the IPR, the interrupt master enable flag (IME), enables and disables all interrupt processing. Even if an interrupt request flag and its corresponding enable flag are set, a service routine cannot be executed until the IME flag is set to logic one. The IME flag can be directly manipulated by EI and DI instructions, regardless of the current enable memory bank (EMB) value.

Table 7-4. Interrupt Priority Register Settings

IPR.2	IPR.1	IPR.0	Result of IPR Bit Setting
0	0	0	Normal interrupt handling according to default priority settings.
0	0	1	Process INTB interrupt at highest priority
0	1	0	Process INT0 interrupt at highest priority
0	1	1	Process INT1 interrupt at highest priority
1	0	0	Process INTP0 interrupt at highest priority
1	0	1	Process INTT0 interrupt at highest priority
1	1	0	Process INTT1 interrupt at highest priority
1	1	1	N/A

NOTE: During normal interrupt processing, interrupts are processed in the order in which they occur. If two or more interrupt requests are received simultaneously, the priority level is determined according to the standard interrupt priorities in Table 7-3 (the default priority assigned by hardware when the lower three IPR bits = "0"). In this case, the higher-priority interrupt request is serviced and the other interrupt is inhibited. Then, when the high-priority interrupt is returned from its service routine by an IRET instruction, the inhibited service routine is started.

 **PROGRAMMING TIP — Setting The INT Interrupt Priority**

The following instruction sequence sets the INT1 interrupt to high priority:

```

BITS      EMB
SMB       15
DI                               ; IPR.3 (IME) ← 0
LD        A,#3H
LD        IPR,A
EI                               ; IPR.3 (IME) ← 1
    
```

EXTERNAL INTERRUPT 0 and 1 MODE REGISTERS (IMOD0, IMOD1)

The following components are used to process external interrupts at the INT0 and INT1 pin:

- Edge detection circuit
- Two mode registers, IMOD0 and IMOD1

The mode registers are used to control the triggering edge of the input signal. IMOD0 and IMOD1 settings let you choose either the rising or falling edge of the incoming signal as the interrupt request trigger.

FB4H	"0"	"0"	IMOD0.1	IMOD0.0
FB5H	"0"	"0"	IMOD1.1	IMOD1.0

IMOD0 and IMOD1 are addressable by 4-bit write instructions. RESET clears all IMOD values to logic zero, selecting rising edges as the trigger for incoming interrupt requests.

Table 7-5. IMOD0 and IMOD1 Register Organization

IMODx	"0"	"0"	IMODx.1	IMODx.0	Effect of IMOD Settings
			0	0	Rising edge detection
			0	1	Falling edge detection
			1	0	Both rising and falling edge detection
			1	1	IRQ0 flag cannot be set to "1"

NOTE: "x" means "0" or "1"

EXTERNAL INTERRUPT 0 AND 1 MODE REGISTERS (Continued)

— You can use INT0/INT1 to release power-down mode

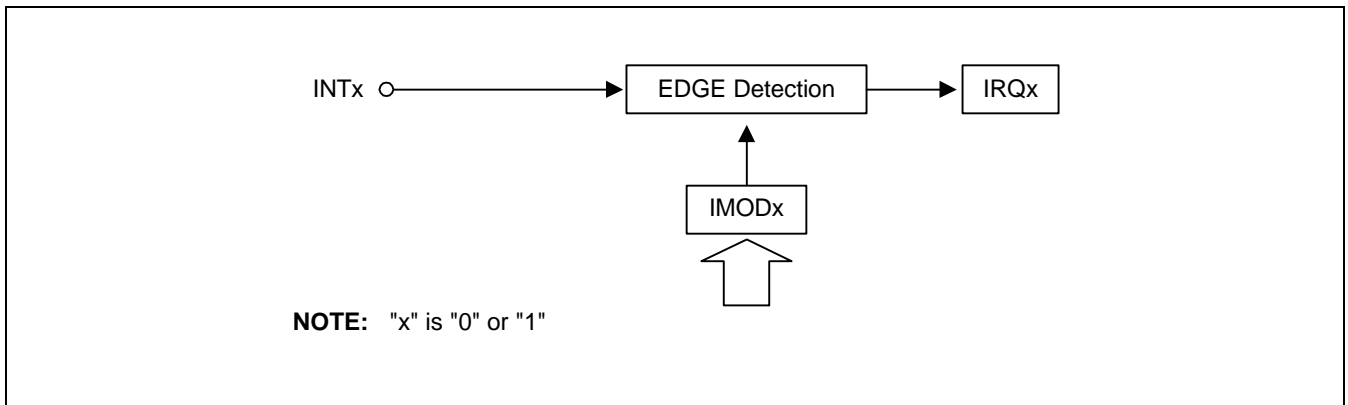


Figure 7-5. Circuit Diagram for INT0 and INT1 Pins

When modifying the IMOD0 and IMOD1 registers, it is possible to accidentally set an interrupt request flag. To avoid unwanted interrupts, take these precautions when writing your programs:

1. Disable all interrupts with a DI instruction.
2. Modify the IMOD0 or IMOD1 register.
3. Clear all relevant interrupt request flags.
4. Enable the interrupt by setting the appropriate IEx flag.
5. Enable all interrupts with an EI instruction.

NOTE: INT0 and INT1 are same in the function.

EXTERNAL INTERRUPT 2 MODE REGISTER (IMOD2)

To generate a key interrupt on a falling edge at KS0-KS7, all KS0-KS7 pins must be configured to input mode. IMOD2 is write-only register that can be written by 4-bit RAM control instruction only. It is mapped to the RAM address FB6H and the reset value of IMOD2 is 0.

FB6H	"0"	IMOD2.2	IMOD2.1	IMOD2.0
------	-----	---------	---------	---------

When a falling edge in any one of KS0-KS7 pins is detected, IRQ2 is set and the release signal of power down mode is generated. INT2, however, does not generate a vector interrupt. Among the pins which were selected as key interrupt, one or more pins which are in input low or output low don't execute a key interrupt function.

Table 7-6. IMOD2 Register Bit Settings

IMOD2	0	IMOD2.2	IMOD2.1	IMOD2.0	Effect of IMOD2 Settings
		0	0	0	Select falling edge of KS0-KS3
		0	0	1	Select falling edge of KS0-KS4
		0	1	0	Select falling edge of KS0-KS5
		0	1	1	Select falling edge of KS0-KS6
		1	0	0	Select falling edge of KS0-KS7

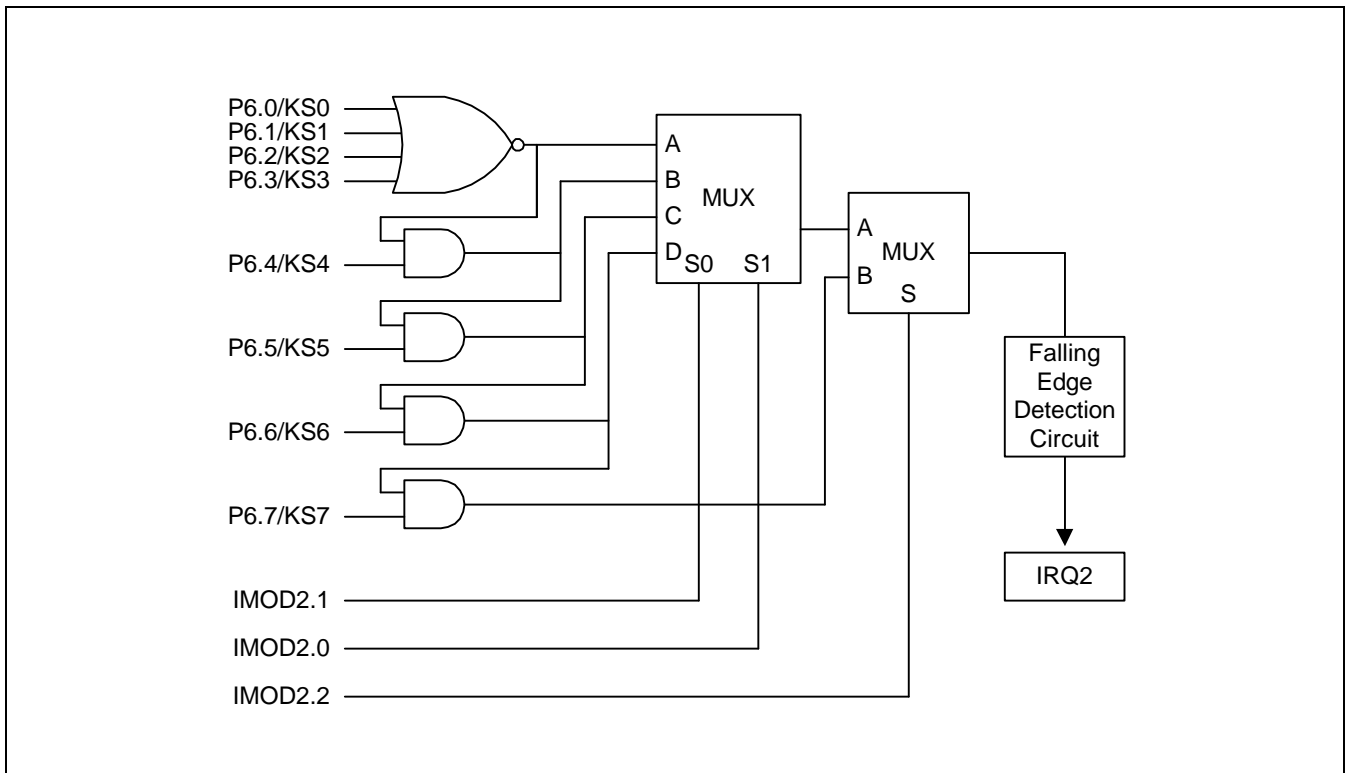


Figure 7-6. Circuit Diagram for INT2

PROGRAMMING TIP — Using INT2 as a Key Input Interrupt

When the INT2 interrupt is used as a key entry interrupt, the selected key interrupt source pin must be set to input:

1. When KS0–KS3 are selected (four pins):

BITS	EMB	
SMB	15	
LD	A,#0H	
LD	IMOD2,A	; (IMOD2) ← #0H, KS0-KS3 falling edge select
LD	EA,#00H	
LD	PMG2,EA	; P6 ← input mode
LD	A,#40H	
LD	PUMOD0,A	; Enable P6 pull-up resistors

INTERRUPT FLAGS

There are three types of interrupt flags: interrupt request and interrupt enable flags that correspond to each interrupt, the interrupt master enable flag, which enables or disables all interrupt processing.

Interrupt Master Enable Flag (IME)

The interrupt master enable flag, IME, enables or disables all interrupt processing. Therefore, even when an IRQx flag is set and its corresponding IEx flag is enabled, the interrupt service routine is not executed until the IME flag is set to logic one.

The IME flag is located in the IPR register (IPR.3). It can be directly be manipulated by EI and DI instructions, regardless of the current value of the enable memory bank flag (EMB).

IME	IPR.2	IPR.1	IPR.0	Effect of Bit Settings
0				Inhibit all interrupts
1				Enable all interrupts

Interrupt Enable Flags (IEx)

IEx flags, when set to logical one, enable specific interrupt requests to be serviced. When the interrupt request flag is set to logical one, an interrupt will not be serviced until its corresponding IEx flag is also enabled.

Interrupt enable flags can be read, written, or tested directly by 1-bit instructions. IEx flags can be addressed directly at their specific RAM addresses, despite the current value of the enable memory bank (EMB) flag.

Table 7-7. Interrupt Enable and Interrupt Request Flag Addresses

Address	Bit 3	Bit 2	Bit 1	Bit 0
FB8H	"U"	"U"	IEB	IRQB
FBAH	"U"	"U"	IEW	IRQW
FBBH	"U"	"U"	IET1	IRQT1
FBCH	"U"	"U"	IET0	IRQT0
FBDH	"U"	"U"	IEP0	IRQP0
FBEH	IE1	IRQ1	IE0	IRQ0
FBFH	"U"	"U"	IE2	IRQ2

NOTES:

1. IEx refers to all interrupt enable flags.
2. IRQx refers to all interrupt request flags.
3. IEx = 0 is interrupt disable mode.
4. IEx = 1 is interrupt enable mode.

Interrupt Request Flags (IRQx)

Interrupt request flags are read/write addressable by 1-bit or 4-bit instructions. IRQx flags can be addressed directly at their specific RAM addresses, regardless of the current value of the enable memory bank (EMB) flag.

When a specific IRQx flag is set to logic one, the corresponding interrupt request is generated. The flag is then automatically cleared to logic zero when the interrupt has been serviced. Exceptions are the watch timer interrupt request flags, IRQW, and the external interrupt 2 flag IRQ2, which must be cleared by software after the interrupt service routine has executed. IRQx flags are also used to execute interrupt requests from software. In summary, follow these guidelines for using IRQx flags:

1. IRQx is set to request an interrupt when an interrupt meets the set condition for interrupt generation.
2. IRQx is set to "1" by hardware and then cleared by hardware when the interrupt has been serviced (with the exception of IRQW and IRQ2).
3. When IRQx is set to "1" by software, an interrupt is generated.

Table 7-8. Interrupt Request Flag Conditions and Priorities

Interrupt Source	Internal/External	Pre-condition for IRQx Flag Setting	Interrupt Priority	IRQ Flag Name
INTB	I	Reference time interval signal from basic timer	1	IRQB
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INT1	E	Rising or falling edge detected at INT1 pin	3	IRQ1
INTP0	E	Falling edge detected at K0–K6 (P0.0–P1.2)	4	IRQP0
INTT0	I	Signals for TCNT0 and TREF0 registers match	5	IRQT0
INTT1	I	Signals for TCNT1 and TREF1 registers match	6	IRQT1
INT2 (note)	E	Falling edge is detected at any of the KS0–KS7 pins	–	IRQ2
INTW	I	Time interval of 0.5 s or 3.19 ms	–	IRQW

NOTE: Refer to page 7-10, 7-11.

NOTES

8 POWER-DOWN

OVERVIEW

The S3C72Q5 microcontroller has two power-down modes to reduce power consumption: idle and stop. Idle mode is initiated by the IDLE instruction and stop mode by the instruction STOP. (Several NOP instructions must always follow an IDLE or STOP instruction in a program.) In idle mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

When RESET occurs during normal operation or during a power-down mode, a reset operation is initiated and the CPU enters idle mode. When the standard oscillation stabilization time interval (31.3 ms at 4.19 MHz) has elapsed, normal CPU operation resumes.

In stop mode, main-system clock oscillation is halted (assuming it is currently operating), and peripheral hardware components are powered-down. The effect of stop mode on specific peripheral hardware components — CPU, basic timer, serial I/O, timer/ counters 0 and 1, watch timer, and LCD controller — and on external interrupt requests, is detailed in Table 8–1.

Idle or stop modes are terminated either by a RESET, or by an interrupt which is enabled by the corresponding interrupt enable flag, IEx. When power-down mode is terminated by RESET, a normal reset operation is executed. Assuming that both the interrupt enable flag and the interrupt request flag are set to "1", power-down mode is released immediately upon entering power-down mode.

When an interrupt is used to release power-down mode, the operation differs depending on the value of the interrupt master enable flag (IME):

- If the IME flag = "0"; If the power down mode release signal is generated, after releasing the power-down mode, program execution starts immediately under the instruction to enter power down mode without execution of interrupt service routine. The interrupt request flag remains set to logic one.
- If the IME flag = "1"; If the power down mode release signal is generated, after releasing the power down mode, two instructions following the instruction to enter power down mode are executed first and the interrupt service routine is executed, finally program is resumed. However, when the release signal is caused by INT2 or INTW, the operation is identical to the IME = "0" condition because INT2 and INTW are a quasi-interrupt.

NOTE

Do not use stop mode if you are using an external clock source because X_{IN} input must be restricted internally to V_{SS} to reduce current leakage.

Table 8-1. Hardware Operation During Power-Down Modes

Operation	Stop Mode	Idle Mode
Instruction	STOP	IDLE
System clock status	STOP mode can be used only if the main-system clock is selected as system clock (CPU clock)	IDLE mode can be used if the main-system clock or sub-system clock is selected as system clock (CPU clock)
Clock oscillator	Main-system clock oscillation stops	Only CPU clock oscillation stops (main and sub-system clock oscillation continues)
Basic timer	Basic timer stops	Basic timer operates (with IRQB set at each reference interval)
Timer/counter 0	Operates only if TCL0 is selected as the counter clock	Timer/counter 0 operates
Timer/counter1	Timer/counter1 stops	Timer/counter 1 operates
Watch timer	Operates only if sub-system clock (fxt) is selected as the counter clock	Watch timer operates
LCD controller	Operates only if a sub-system clock is selected as LCDCK	LCD controller operates
External interrupts	INT0,INT1,INT2 and INTP0 are acknowledged.	INT0,INT1,INT2 and INTP0 are acknowledged.
CPU	All CPU operations are disabled	All CPU operations are disabled
Mode release signal	Interrupt request signals are enable by an interrupt enable flag or by RESET input.	Interrupt request signals are enable by an interrupt enable flag or by RESET input.

Table 8-2. System Operating Mode Comparison

Mode	Condition	STOP/IDLE Mode Start Method	Current Consumption
Main operating mode	Main oscillator runs. Sub oscillator runs System clock is the main oscillation clock.	–	A
Main Idle mode	Main oscillator runs. Sub oscillator runs System clock is the main oscillation clock.	IDLE instruction	B
Main Stop mode	Main oscillator runs. Sub oscillator runs. System clock is the main oscillation clock.	STOP instruction	D
Sub operating mode	Main oscillator is stopped by SCMOD.3. Sub oscillator runs. System clock is the sub oscillation clock.	–	C
Sub Idle Mode	Main oscillator is stopped by SCMOD.3. Sub oscillator runs. System clock is the sub oscillation clock.	IDLE instruction	D

NOTE: The current consumption is: A > B > C > D

IDLE MODE TIMING DIAGRAMS

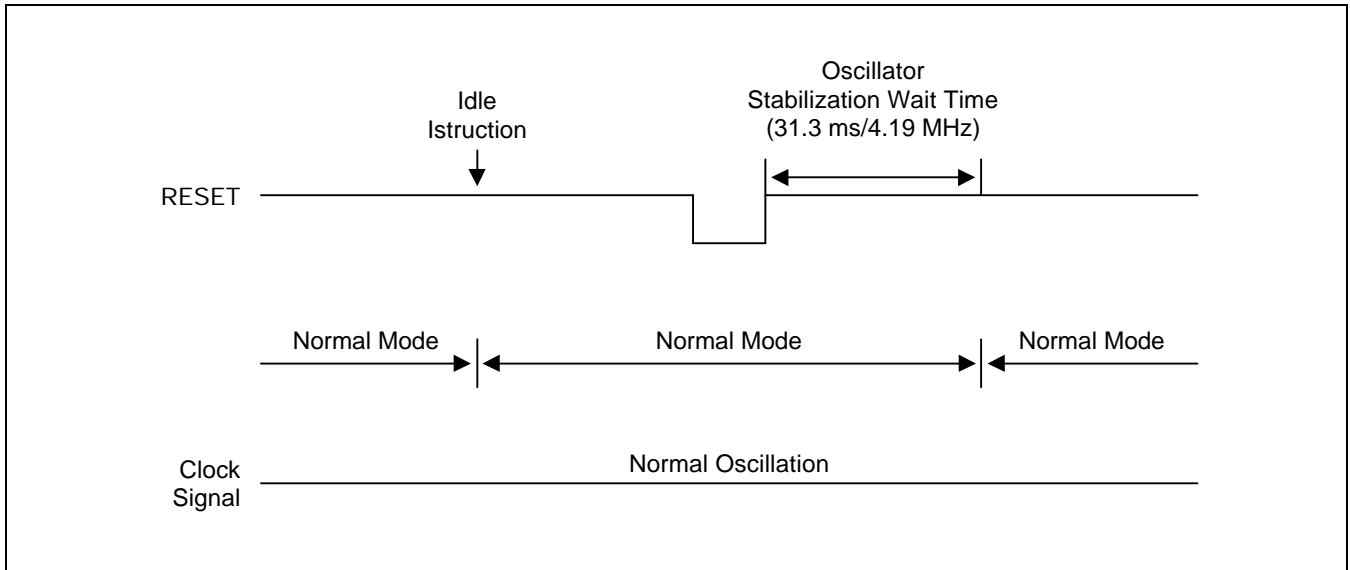


Figure 8-1. Timing When Idle Mode is Released by RESET

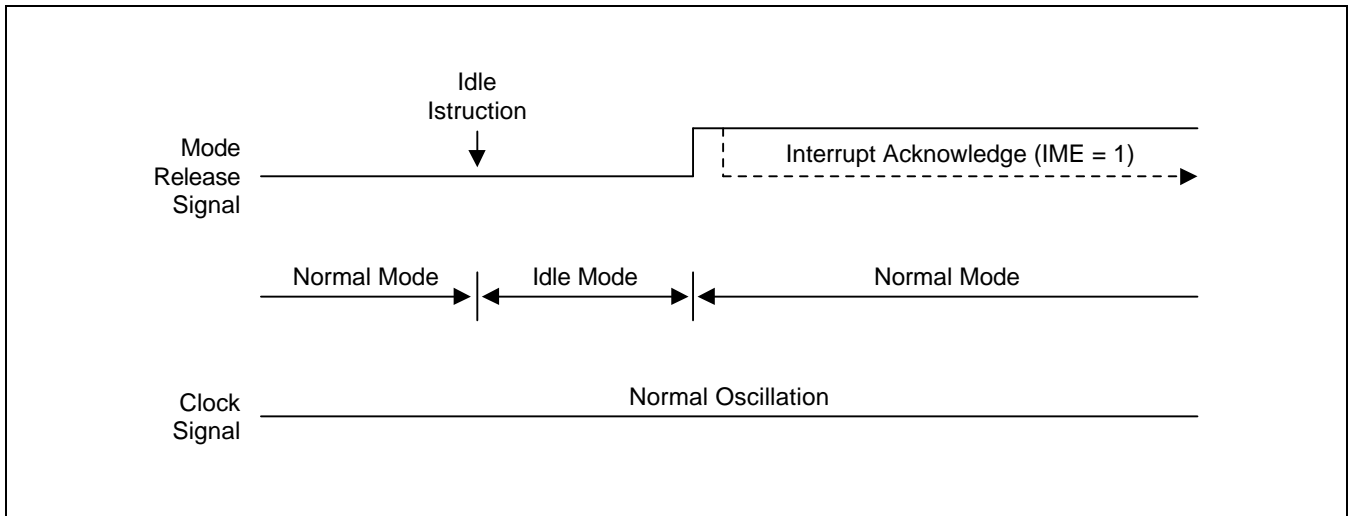


Figure 8-2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

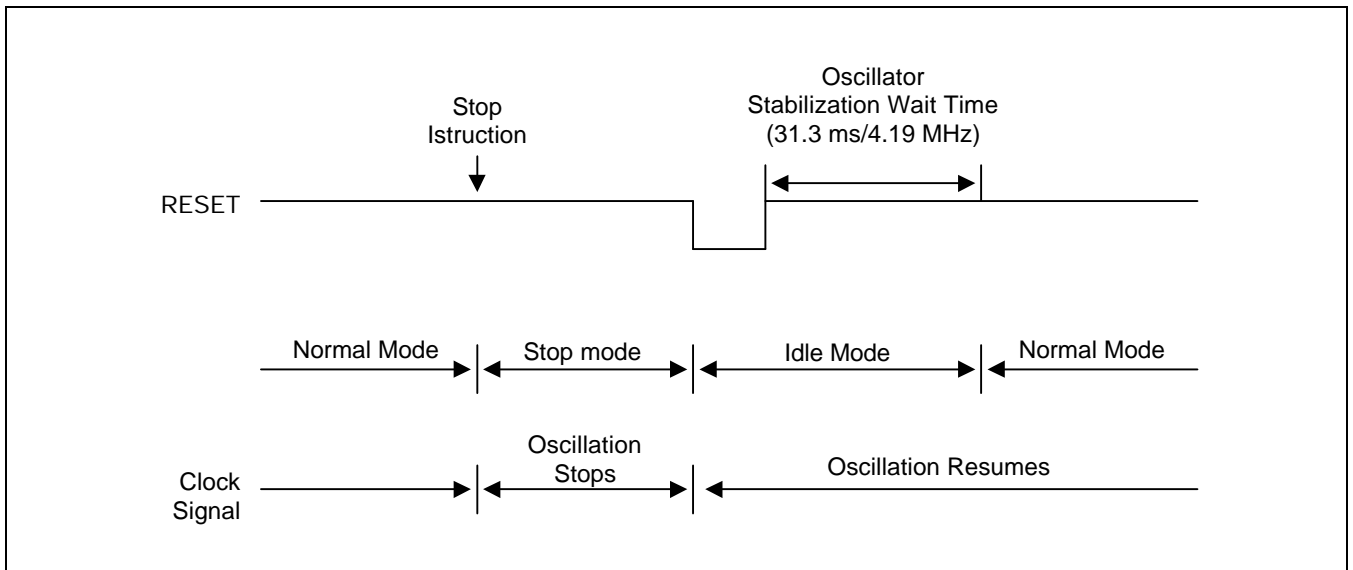


Figure 8-3. Timing When Stop Mode is Released by RESET

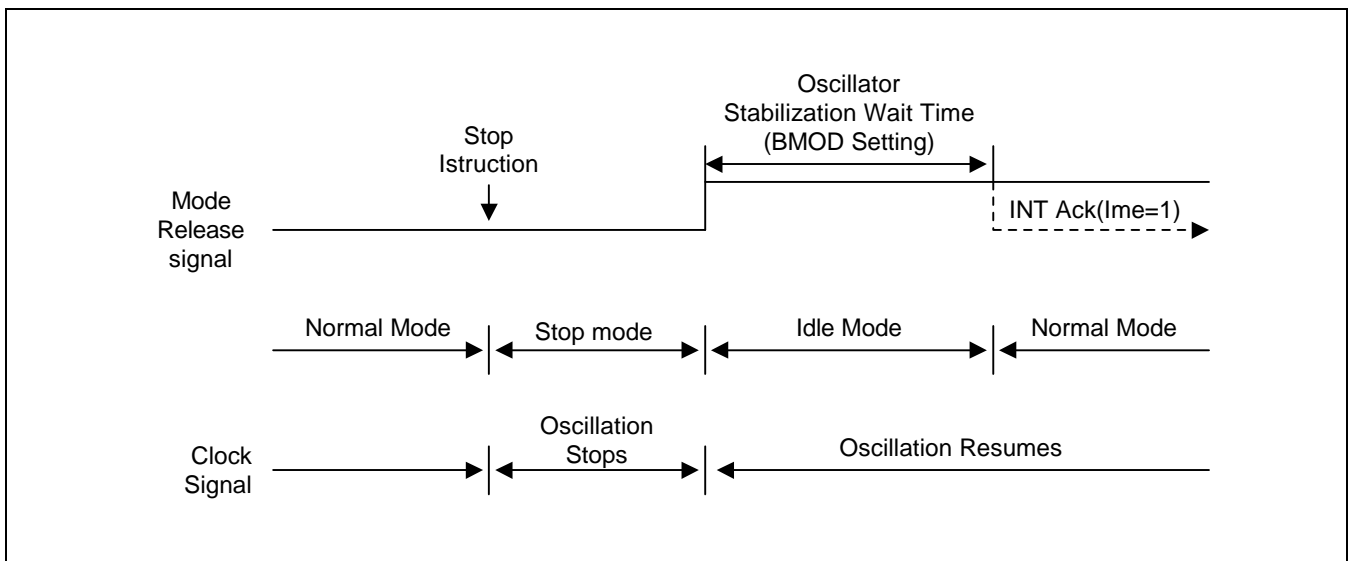


Figure 8-4. Timing When Stop Mode is Released by an Interrupt

PROGRAMMING TIP — Reducing Power Consumption for Key Input Interrupt Processing

The following code shows real-time clock and interrupt processing for key inputs to reduce power consumption. In this example, the system clock source is switched from the main-system clock to a sub-system clock and the LCD display is turned on:

```

KEYCLK  DI
        CALL    MA2SUB      ; Main-system clock → sub-system clock switch subroutine
        SMB     15
        LD      EA,#00H
        LD      P4,EA      ; All key strobe outputs to low level
        LD      A,#4H
        LD      IMOD2,A    ; Select KS0-KS7 enable
        SMB     0
        BITR    IRQW
        BITR    IRQ2
        BITS    IEW
        BITS    IE2

CLKS1   CALL    WATDIS     ; Execute clock and display changing subroutine
        BTSTZ   IRQ2
        JR      CIDLE
        CALL    SUB2MA     ; Sub-system clock → main-system clock switch subroutine
        EI
        RET

CIDLE   IDLE          ; Engage idle mode
        NOP
        NOP
        NOP
        JPS     CLKS1

```

RECOMMENDED CONNECTIONS FOR UNUSED PINS

To reduce overall power consumption, please configure unused pins according to the guidelines described in Table 8-3.

Table 8-3. Unused Pin Connections for Reducing Power Consumption

Pin/Share Pin Names	Recommended Connection
P0.0-P0.3/K0-K3 P1.0-P1.2/K4-K6 P4.0/TCL0 P4.1/TCLO0 P5.0-P5.1 P5.2/BUZ P6.0-P6.3/KS0-KS3/DM0-DM3 P7.0/KS4/DM4 P7.1/KS5/DM5/COM11 P7.2-P7.3/KS6-KS7/COM10-COM11	Input mode: Connect to V_{DD} Output mode: No connection
P4.2/ INT0-P4.3/INT1	Connect to V_{DD}
SEG0-SEG15/P8.0-P8.15 SEG16-SEG23/D0-D7 SEG24-SEG42/A0-A18 SEG43-SEG44/DR, DW SEG45-SEG59 COM0-COM11	No connection
TEST	Connect to V_{SS}

NOTES

9

RESET

OVERVIEW

When a **RESET** signal is input during normal operation or power-down mode, a hardware reset operation is initiated and the CPU enters idle mode. Then, when the standard oscillation stabilization interval of 31.3 ms at 4.19 MHz has elapsed, normal system operation resumes.

Regardless of when the **RESET** occurs — during normal operating mode or during a power-down mode — most hardware register values are set to the reset values described in Table 9–1. The current status of several register values is, however, always retained when a **RESET** occurs during idle or stop mode; If a **RESET** occurs during normal operating mode, their values are undefined. Current values that are retained in this case are as follows:

- Carry flag
- Data memory values
- General-purpose registers E, A, L, H, X, W, Z, and Y

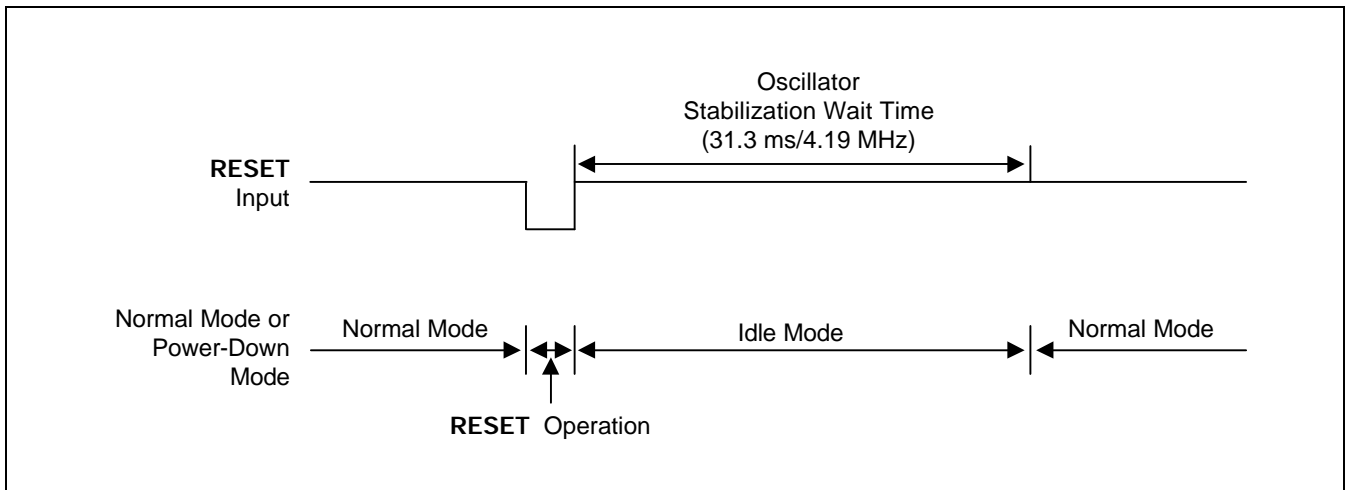


Figure 9-1. Timing for Oscillation Stabilization after RESET

HARDWARE REGISTER VALUES AFTER RESET

Table 9–1 gives you detailed information about hardware register values after a **RESET** occurs during power-down mode or during normal operation.

Table 9-1. Hardware Register Values After RESET

Hardware Component or Subcomponent	If RESET Occurs During Power down Mode	If RESET Occurs during normal operating
Program counter (PC)	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.
Program Status Word (PSW):		
Carry flag (C)	Retained	Undefined
Skip flag (SC0-SC2)	0	0
Interrupt status flags (IS0, IS1)	0	0
Bank enable flags (EMB, ERB)	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.
Stack pointer (SP)	Undefined	Undefined
Data Memory (RAM):		
Registers E, A, L, H, X, W, Z, Y	Values retained	Undefined
General-purpose registers	Values retained ^(note)	Undefined
Bank selection registers (SMB, SRB)	15, 0	15, 0
BSC register (BSC0-BSC3)	0	0
Bank1 page select register (PASR)	0	0
Key scan register (KSR0-KSR3)	0	0
Clocks:		
Power control register (PCON)	0	0
Clock output mode register (CLMOD)	0	0
System clock mode register (SCMOD)	0	0
Interrupts:		
Interrupt request flags (IRQx)	0	0
Interrupt enable flags (IE _x)	0	0
Interrupt priority flag (IPR)	0	0
Interrupt master enable flag (IME)	0	0
INT0 mode register (IMOD0)	0	0
INT1 mode register (IMOD1)	0	0
INT2 mode register (IMOD2)	0	0

NOTE: The values of the 0F8H–0FDH are not retained when a RESET signal is input.

Table 9-1. Hardware Register Values After RESET (Continued)

Hardware Component or Subcomponent	If RESET Occurs During Power down Mode	If RESET Occurs during normal operation
I/O Ports:		
Output buffers	Off	Off
Output latches	0	0
Port mode flags (PM)	0	0
Pull-up resistor mode reg (PUMOD0)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Watch-dog Timer:		
Watch-dog timer mode selection (WDMOD)	A5	A5
Watch-dog timer counter clear flag(WDFLAG)	0	0
Timer/Counter 0:		
Count register (TCNT0)	0	0
Reference register (TREF0)	FFH	FFH
Mode register (TMOD0)	0	0
Output enable flag (TOE0)	0	0
Timer/Counter 1:		
Count register (TCNT1)	0	0
Reference register (TREF1)	FFH	FFH
Mode register (TMOD1)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
LCD Driver/Controller:		
LCD mode register (LMOD0/1)	0	0
Display data memory	Values retained	Undefined
Output buffers	Off	Off

NOTES

10 I/O PORTS

OVERVIEW

The S3C72Q5 has 39 I/O Lines. There are total of 16 output pins, 23 configurable I/O pins, for a total number of 39 pins.

Port Mode Flags

Port mode flags (PM) are used to configure I/O ports to input or output mode by setting or clearing the corresponding I/O buffer. PM flags are stored in one 8-bit, 4-bit register and are addressable by 8-bit, 4-bit write instructions respectively.

Output Ports 8

Output ports 8 consists of 16 pins that can be used either for LCD segment data output or for normal 1-bit output. When LCD display is off, P8 can be used to normal output. The value of P8 is determined by KSR0-KSR3 regardless of LMOD.0. (refer to P12-17)

Pull-up Resistor Mode Register (PUMOD0)

The pull-up resistor mode register (PUMOD0) is 8-bit register used to assign internal pull-up resistors by software to specific I/O ports.

When a configurable I/O port pin is used as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up is enabled by a corresponding PUMOD0 bit setting.

PUMOD0 is addressable by 8-bit write instructions only. RESET clears PUMOD0 register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

N-channel Open-drain Mode Register (PNE0)

The n-channel open-drain mode register (PNE0) is used to configure outputs as n-channel open-drain outputs or as push-pull outputs.

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
0	I/O	4	P0.0-P0.3 (K0 - K3)	FF0H	4-bit I/O port. 1,4, and 8-bit read/write, and test are possible.
1		3	P1.0-P1.2 (K4 - K6)	FF1H	Individual pins can be specified as input or output. 7-bit pull-up resistors are assignable by software. Pull-up resistors are automatically disabled for output pins.
4		4	P4.0-P4.3	FF4H	4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. 4-bit unit pins are software configurable as input or output. Individual pins are software configurable as open-drain or push-pull output. 4-bit pull-up resistors are assignable by software and pull-up resistors are automatically disabled for output pins.
5		4	P5.0-P5.3	FF5H	
6		O	4	P6.0-P6.3	FF6H
7	4		P7.0-P7.3	FF7H	
8	O	16	P8.0-P8.15	FA2H- FA5H	4-bit controllable output

Table 10-2. Port Pin Status During Instruction Execution

Instruction Type	Example	Input Mode Status	Output Mode Status
1-bit test 1-bit input 4-bit input 8-bit input	BTST P0.1 LDB C,P1.3 LD A,P6 LD EA,P4	Input or test data at each pin	Input or test data at output latch
1-bit output	BITR P4.0	Output latch contents undefined	Output pin status is modified
4-bit output 8-bit output	LD P5,A LD P6,EA	Transfer accumulator data to the output latch	Transfer accumulator data to the output pin

PORT MODE FLAGS (PM FLAGS)

Port mode flags (PM) are used to configure I/O ports to input or output mode by setting or clearing the corresponding I/O buffer. PM flags are stored in one 8-bit registers and are addressable by 8-bit write instructions reflectively.

For convenient program reference, PM flags are organized into three groups — PMG0, PMG1, and PMG2 as shown in Table 10-3.

When a PM flag is "0", the port is set to input mode; when it is "1", the port is enabled for output. RESET clears all port mode flags to logical zero, automatically configuring the corresponding I/O ports to input mode.

Table 10-3. Port Mode Group Flags

PM Group ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PMG0	FEAH	PM 0.3	PM 0.2	PM 0.1	PM 0.0
	FEBH	"0"	PM 1.2	PM 1.1	PM 1.0
PMG1	FECH	PM 4.3	PM 4.2	PM 4.1	PM 4.0
	FEDH	PM 5.3	PM 5.2	PM 5.1	PM 5.0
PMG2	FEEH	PM 6.3	PM 6.2	PM 6.1	PM 6.0
	FEEH	PM 7.3	PM 7.2	PM 7.1	PM 7.0

NOTE: If bit = "0", the corresponding I/O pin is set to input mode. If bit = "1", the pin is set to output mode. All flags are cleared to "0" following RESET.

To use INTP0 interrupt, P0 and P1 must be set to external interrupt pins by LMOD.6-LMOD.4, input mode by PMG0 and pull-up resistor enable by PUMOD0.

PROGRAMMING TIP — Configuring I/O Ports to Input or Output

Configure P4 as an output port:

```

BITS      EMB
SMB      15
LD      EA,#0FH
LD      PMG1,EA      ; P4 ← Output

```

PULL-UP RESISTOR MODE REGISTER (PUMOD0)

The pull-up resistor mode register (PUMOD0) is an 8-bit register used to assign internal pull-up resistors by software to specific I/O ports.

When a configurable I/O port pin is used as an output pin, its assigned pull-up resistor is automatically disabled even though the pin's pull-up is enabled by a corresponding PUMOD0 bit setting.

RESET clears PUMOD0 register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-4. Pull-Up Resistor Mode Register (PUMOD0) Organization

Bit Name	PUMOD0 function	
PUMOD0.7	0	Disconnect port 7 pull-up resistor
	1	Connect port 7 pull-up resistor
PUMOD0.6	0	Disconnect port 6 pull-up resistor
	1	Connect port 6 pull-up resistor
PUMOD0.5	0	Disconnect port 5 pull-up resistor
	1	Connect port 5 pull-up resistor
PUMOD0.4	0	Disconnect port 4 pull-up resistor
	1	Connect port 4 pull-up resistor
PUMOD0.3	0	Always logic zero
PUMOD0.2	0	Always logic zero
PUMOD0.1	0	Always logic zero
PUMOD0.0	0	Disconnect port 0,1 pull-up resistor
	1	Connect port 0,1 pull-up resistor

NOTE: When P0, P1 are used to external interrupt pins, the pull-up resistors of input mode are determined by key strobe signal (refer to P12-7).

 **PROGRAMMING TIP — Enabling and Disabling I/O Port Pull-Up Resistors**

P6 enable pull-up resistors.

```

BITS      EMB
SMB       15
LD        EA,#40H
LD        PUMOD0,EA      ; P6 pull-up resistor enable

```

N-channel Open-drain Mode Register (PNE0)

PNE0	Address	Bit3	Bit2	Bit1	Bit0
	FE6H	PNE4.3	PNE4.2	PNE4.1	PNE4.0
	FE7H	PNE5.3	PNE5.2	PNE5.1	PNE5.0

The n-channel open-drain mode register, PNE0, is used to configure port4 and 5 to n-channel open-drain outputs or as push-pull outputs. When a bit in the PNE0 register is set to one, the corresponding output pin is configured to n-channel open-drain; when set to "0", the output pin is configured to push-pull.

The PNE0 register consists of an 8-bit, as shown above. PNE0 can be addressed by 8-bit write instructions only.

PORT 0,1 CIRCUIT DIAGRAM

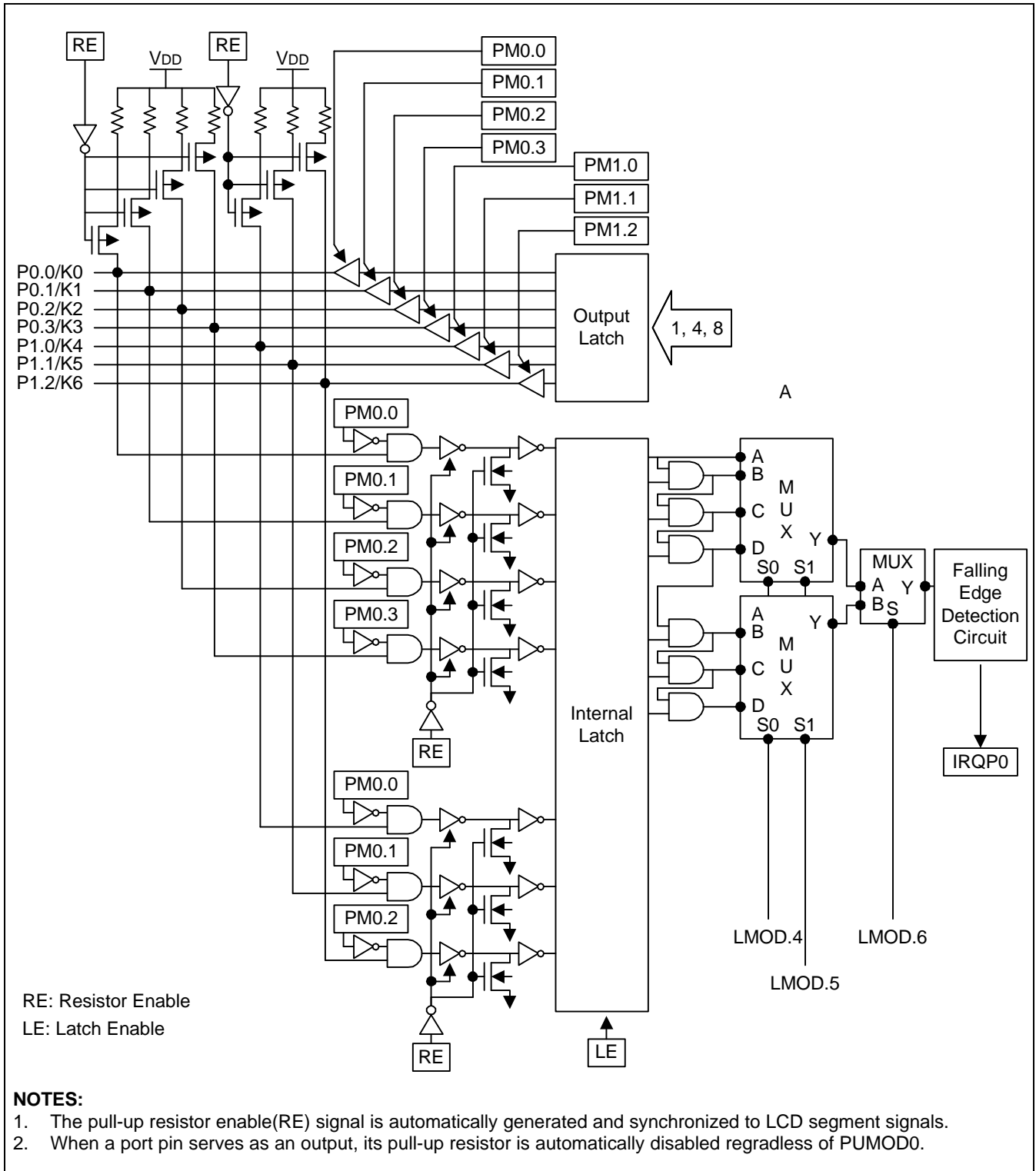


Figure 10-1. Port 0,1 Circuit Diagram

PORT 4 CIRCUIT DIAGRAM

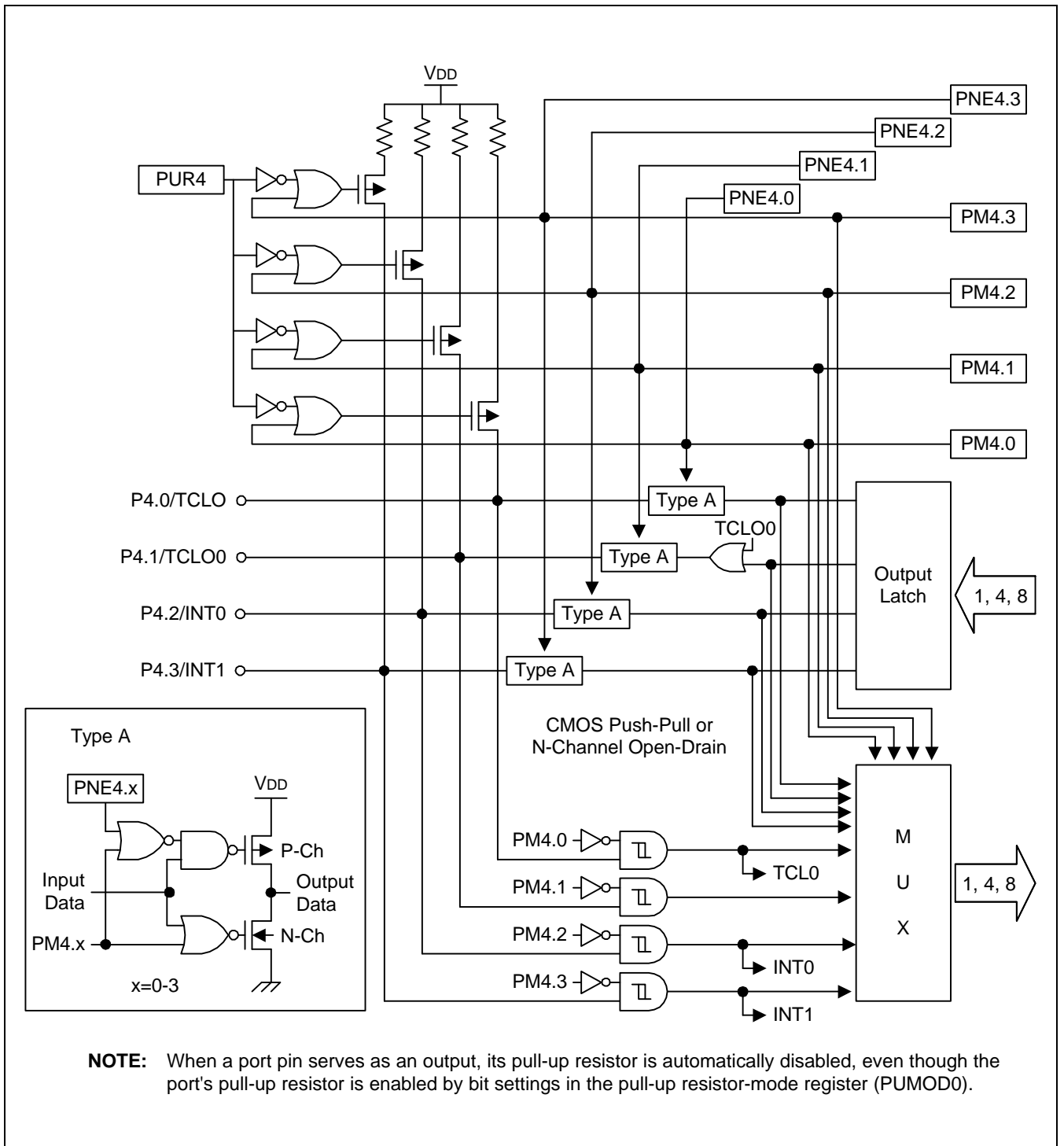


Figure 10-2. Port 4 Circuit Diagram

PORT 5 CIRCUIT DIAGRAM

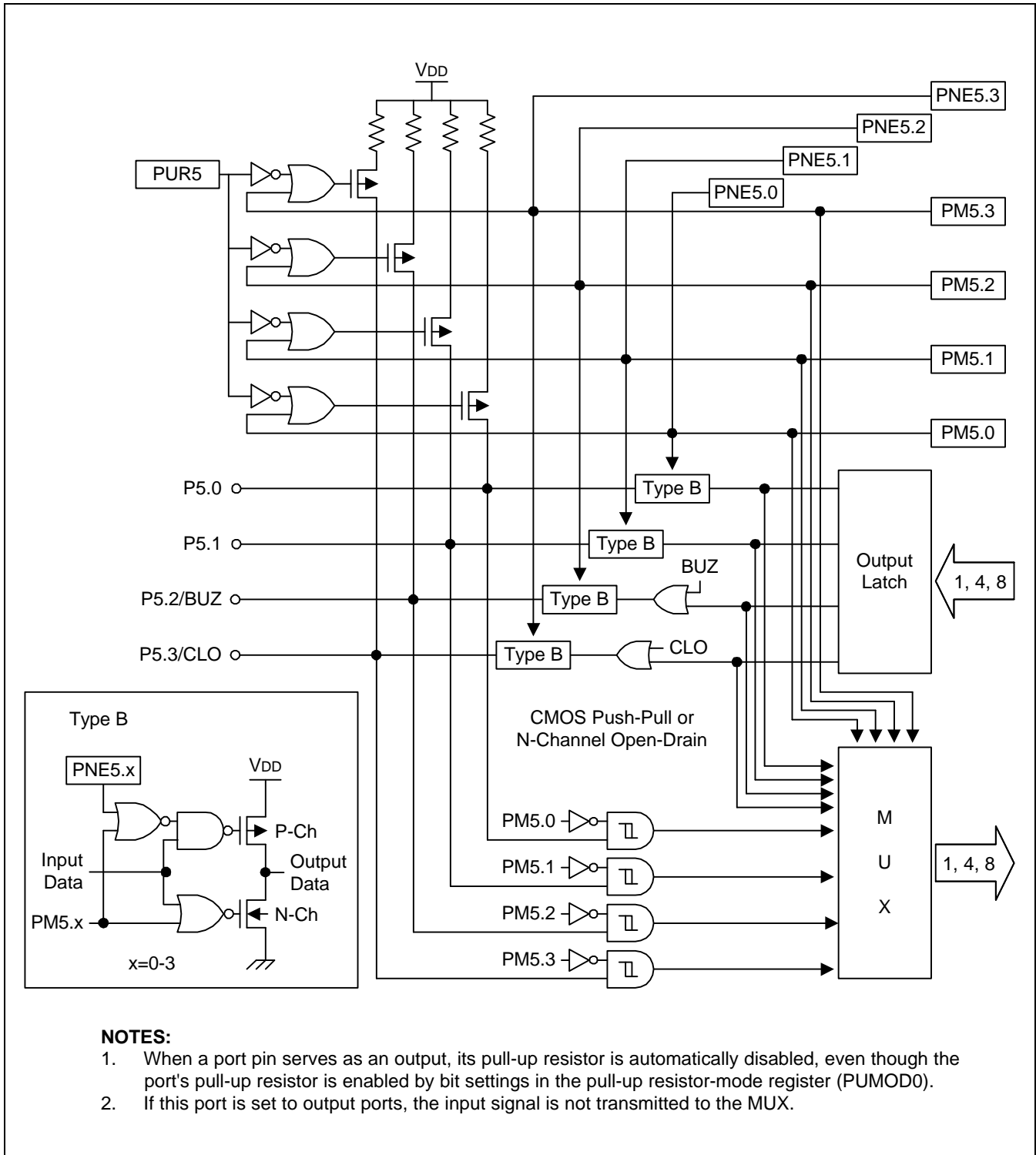


Figure 10-3. Port 5 Circuit Diagram

PORT 6 CIRCUIT DIAGRAM

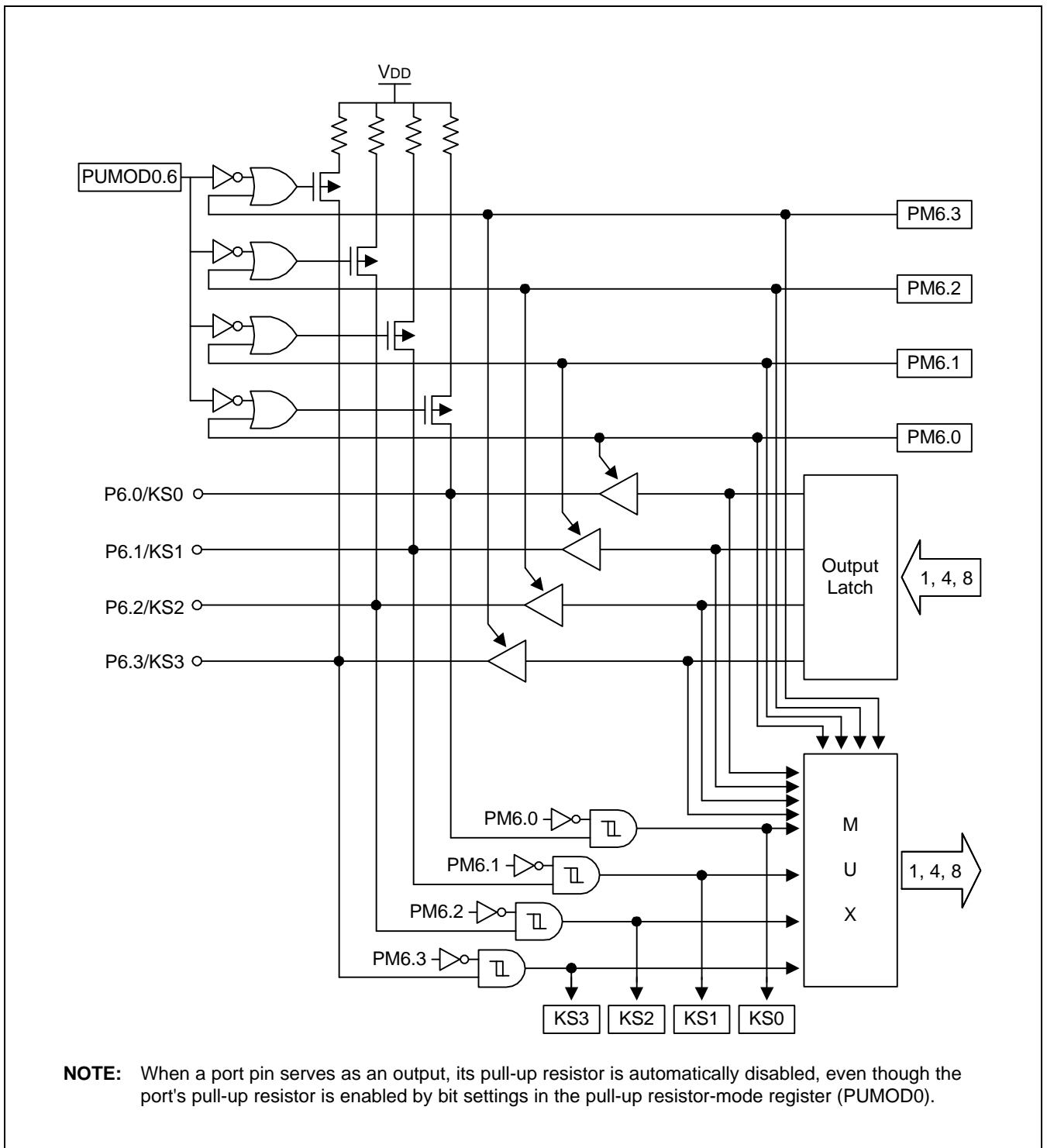
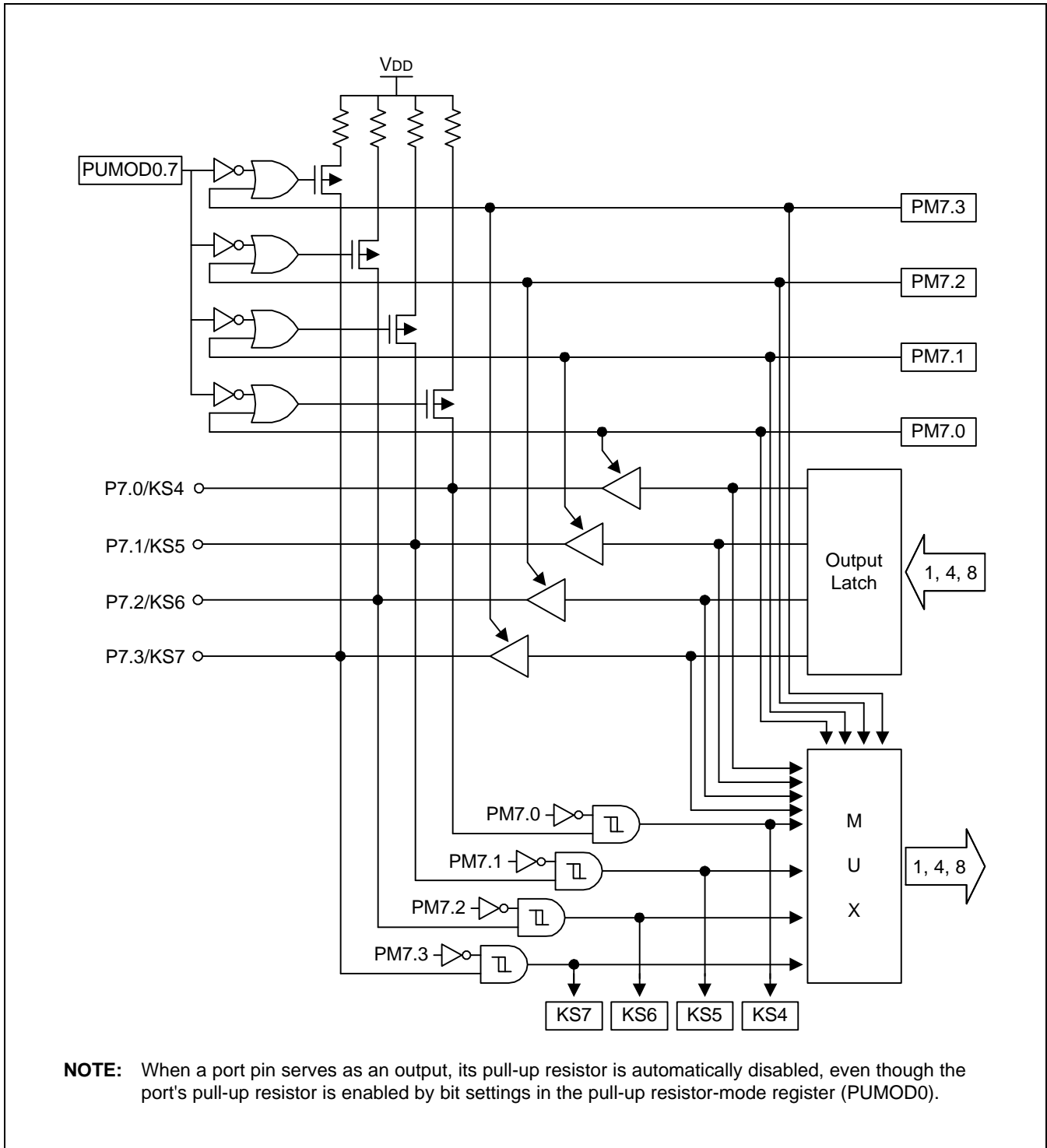


Figure 10-4. Port 6 Circuit Diagram

PORT 7 CIRCUIT DIAGRAM



NOTE: When a port pin serves as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings in the pull-up resistor-mode register (PUMOD0).

Figure 10-5. Port 7 Circuit Diagram

11

TIMERS and TIMER/COUNTERS

OVERVIEW

The S3C72Q5 microcontroller has two timers, and two timer/counters modules:

- 8-bit basic timer (BT)
- 8-bit timer/counter 0 (TC0)
- 8-bit timer/counter 1 (TC1)
- Watch timer (WT)

The 8-bit basic timer (BT) is the microcontroller's main interval timer and watch-dog timer. It generates an interrupt request at a fixed time interval when the appropriate modification is made to its mode register. The basic timer also is used to determine clock oscillation stabilization time when stop mode is released by an interrupt and after a RESET.

The 8-bit timer/counter 0 (TC0) is a programmable timer/counter that is used primarily for event counting and for clock frequency modification and output.

The 8-bit timer/counter 1 (TC1) is a programmable timer/counter that is used primarily for event counting and for clock frequency modification.

The watch timer (WT) module consists of an 8-bit watch timer mode register, a clock selector, and a frequency divider circuit. Watch timer functions include real-time and watch-time measurement, main and subsystem clock interval timing, buzzer output generation. It also generates a clock signal for the LCD controller.

BASIC TIMER (BT)

OVERVIEW

The 8-bit basic timer (BT) has six functional components:

- Clock selector logic
- 4-bit mode register (BMOD)
- 8-bit counter register (BCNT)
- 8-bit watchdog timer mode register (WDMOD)
- Watchdog timer counter clear flag (WDTCF)
- 3-bit watchdog timer counter register(WDCNT)

The basic timer generates interrupt requests at precise intervals, based on the frequency of the system clock. You can use the basic timer as a "watchdog" timer for monitoring system events or use BT output to stabilize clock oscillation when stop mode is released by an interrupt and following RESET. Bit settings in the basic timer mode register BMOD turns the BT module on and off, selects the input clock frequency, and controls interrupt or stabilization intervals.

Interval Timer Function

The basic timer's primary function is to measure elapsed time intervals. The standard time interval is equal to 256 basic timer clock pulses.

To restart the basic timer, one bit setting is required: bit 3 of the mode register BMOD should be set to logic one. The input clock frequency and the interrupt and stabilization interval are selected by loading the appropriate bit values to BMOD.2-BMOD.0.

The 8-bit counter register, BCNT, is incremented each time a clock signal is detected that corresponds to the frequency selected by BMOD. BCNT continues incrementing as it counts BT clocks until an overflow occurs (≥ 255). An overflow causes the BT interrupt request flag (IRQB) to be set to logic one to signal that the designated time interval has elapsed. An interrupt request is then generated, BCNT is cleared to logic zero, and counting continues from 00H.

Watchdog Timer Function

The basic timer can also be used as a "watchdog" timer to signal the occurrence of system or program operation error. For this purpose, instruction that clear the watchdog timer (BITS WDTCF) should be executed at proper points in a program within given period. If an instruction that clears the watchdog timer is not executed within the given period and the watchdog timer overflows, reset signal is generated and the system restarts with reset status. An operation of watchdog timer is as follows:

- Write some values (except #5AH) to watchdog timer mode register, WDMOD.
- If WDCNT overflows, system reset is generated.

Oscillation Stabilization Interval Control

Bits 2-0 of the BMOD register are used to select the input clock frequency for the basic timer. This setting also determines the time interval (also referred to as 'wait time') required to stabilize clock signal oscillation when stop mode is released by an interrupt. When a RESET signal is inputted, the standard stabilization interval for system clock oscillation following the RESET is 31.3 ms at 4.19 MHz.

Table 11-1. Basic Timer Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
BMOD	Control	Controls the clock frequency (mode) of the basic timer; also, the oscillation stabilization interval after stop mode release or RESET	4-bit	F85H	4-bit write-only; BMOD.3: 1-bit writeable	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H-F87H	8-bit read-only	U (note)
WDMOD	Control	Controls watchdog timer operation.	8-bit	F98H-F99H	8-bit write-only	A5H
WDTCF	Control	Clears the watchdog timer's counter.	1-bit	F9AH.3	1-, 4-bit write	"0"

NOTE: 'U' means the value is undetermined after a RESET.

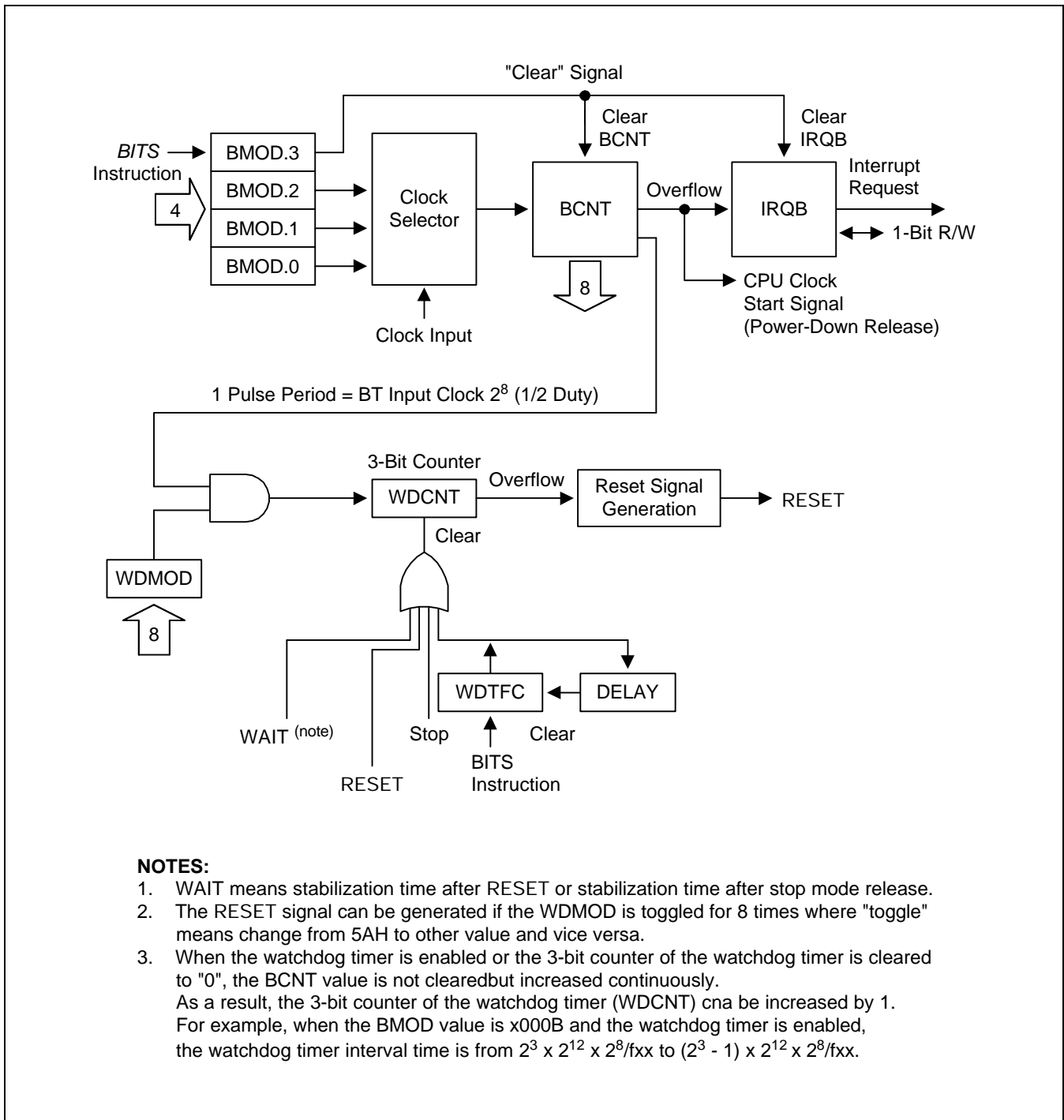


Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

The basic timer mode register, BMOD, is a 4-bit write-only register. Bit 3, the basic timer start control bit, is also 1-bit addressable. All BMOD values are set to logic zero following RESET and interrupt request signal generation is set to the longest interval. (BT counter operation cannot be stopped.) BMOD settings have the following effects:

- Restart the basic timer;
- Control the frequency of clock signal input to the basic timer;
- Determine time interval required for clock oscillation to stabilize following the release of stop mode by an interrupt.

By loading different values into the BMOD register, you can dynamically modify the basic timer clock frequency during program execution. Four BT frequencies, ranging from $f_{xx}/2^{12}$ to $f_{xx}/2^5$, are selectable. Since BMOD's reset value is logic zero, the default clock frequency setting is $f_{xx}/2^{12}$.

The most significant bit of the BMOD register, BMOD.3, is used to restart the basic timer. When BMOD.3 is set to logic one (enabled) by a 1-bit write instruction, the contents of the BT counter register (BCNT) and the BT interrupt request flag (IRQB) are both cleared to logic zero, and timer operation is restarted.

The combination of bit settings in the remaining three registers — BMOD.2, BMOD.1, and BMOD.0 — determine the clock input frequency and oscillation stabilization interval.

Table 11-2. Basic Timer Mode Register (BMOD) Organization

BMOD.3			Basic Timer Enable/Disable Control Bit	
1			Restart basic timer; clear IRQB, BCNT, and BMOD.3 to "0"	

BMOD.2	BMOD.1	BMOD.0	Basic Timer Input Clock	Interrupt Interval Time (Wait Time)
0	0	0	$f_{xx}/2^{12}$ (1.02 kHz)	$2^{20}/f_{xx}$ (250 ms)
0	1	1	$f_{xx}/2^9$ (8.18 kHz)	$2^{17}/f_{xx}$ (31.3 ms)
1	0	1	$f_{xx}/2^7$ (32.7 kHz)	$2^{15}/f_{xx}$ (7.82 ms)
1	1	1	$f_{xx}/2^5$ (131 kHz)	$2^{13}/f_{xx}$ (1.95 ms)

NOTES:

1. Clock frequencies and interrupt interval time assume a system oscillator clock frequency (f_{xx}) of 4.19 MHz.
2. f_{xx} = selected system clock frequency.
3. Wait time is the time required to stabilize clock signal oscillation after stop mode is released. The data in the table column 'Interrupt Interval Time' can also be interpreted as "Oscillation Stabilization."
4. The standard stabilization time for system clock oscillation following a RESET is 31.3 ms at 4.19 MHz.

BASIC TIMER COUNTER (BCNT)

BCNT is an 8-bit counter for the basic timer. It can be addressed by 8-bit read instructions. RESET leaves the BCNT counter value undetermined. BCNT is automatically cleared to logic zero whenever the BMOD register control bit (BMOD.3) is set to "1" to restart the basic timer. It is incremented each time a clock pulse of the frequency determined by the current BMOD bit settings is detected.

When BCNT has incremented to hexadecimal 'FFH' (≥ 255 clock pulses), it is cleared to '00H' and an overflow is generated. The overflow causes the interrupt request flag, IRQB, to be set to logic one. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

NOTE

Always execute a BCNT read operation twice to eliminate the possibility of reading unstable data while the counter is incrementing. If, after two consecutive reads, the BCNT values match, you can select the latter value as valid data. Until the results of the consecutive reads match, however, the read operation must be repeated until the validation condition is met.

BASIC TIMER OPERATION SEQUENCE

The basic timer's sequence of operations may be summarized as follows:

1. Set BMOD.3 to logic one to restart the basic timer
2. BCNT is then incremented by one after each clock pulse corresponding to BMOD selection
3. BCNT overflows if $BCNT \geq 255$ (BCNT = FFH)
4. When an overflow occurs, the IRQB flag is set by hardware to logic one
5. The interrupt request is generated
6. BCNT is then cleared by hardware to logic zero
7. Basic timer resumes counting clock pulses

PROGRAMMING TIP — Using the Basic Timer

1. To read the basic timer count register (BCNT):

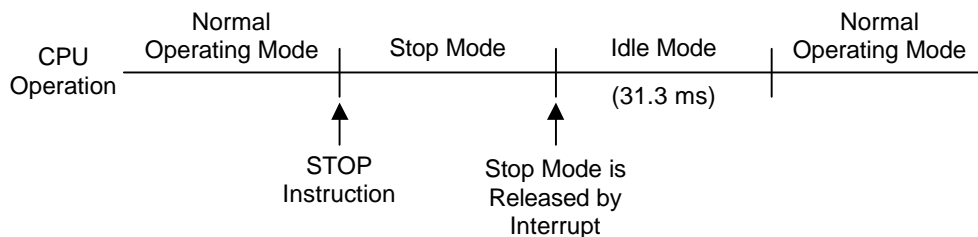
```

BCNTR          BITS      EMB
               SMB      15
               LD      EA,BCNT
               LD      YZ,EA
               LD      EA,BCNT
               CPSE   EA,YZ
               JR      BCNTR
  
```

2. When stop mode is released by an interrupt, set the oscillation stabilization interval to 31.3ms (at 4.19MHz):

```

BITS      EMB
SMB      15
LD      A,#0BH
LD      BMOD,A           ; Wait time is 31.3ms
NOP
STOP                    ; Set stop power-down mode
NOP
NOP
NOP
  
```



3. To set the basic timer interrupt interval time to 1.95 ms (at 4.19 MHz):

```

BITS      EMB
SMB      15
LD      A,#0FH
LD      BMOD,A
EI
BITS      IEB           ; Basic timer interrupt enable flag is set to "1"
  
```

4. Clear BCNT and the IRQB flag and restart the basic timer:

```

BITS      EMB
SMB      15
BITS      BMOD.3
  
```

WATCHDOG TIMER MODE REGISTER (WDMOD)

The watchdog timer mode register, WDMOD, is a 8-bit write-only register. WDMOD register controls to enable or disable the watchdog function. WDMOD values are set to logic "A5H" following RESET and this value enables the watchdog timer. Watchdog timer is set to the longest interval because BT overflow signal is generated with the longest interval.

WDMOD	Watchdog Timer Enable/Disable Control
5AH	Disable watchdog timer function
Any other value	Enable watchdog timer function

WATCHDOG TIMER COUNTER (WDCNT)

The watchdog timer counter, WDCNT, is a 3-bit counter. WDCNT is automatically cleared to logic zero, and restarts whenever the WDTCF register control bit is set to "1". RESET, stop, and wait signal clears the WDCNT to logic zero also.

WDCNT increments each time a clock pulse of the overflow frequency determined by the current BMOD bit setting is generated. When WDCNT has incremented to hexadecimal "07H", it is cleared to "00H" and an overflow is generated. The overflow causes the system RESET. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

WATCHDOG TIMER COUNTER CLEAR FLAG (WDTCF)


The watchdog timer counter clear flag, WDTCF, is a 1-bit write instruction. When WDTCF is set to one, it clears the WDCNT to zero and restarts the WDCNT. WDTCF register bits 2–0 are always logic zero.

Table 11-3. Watchdog Timer Interval Time

BMOD	BT Input Clock	WDCNT Input Clock	WDT Interval Time
x000b	$f_{xx}/2^{12}$	$f_{xx}/(2^{12} \times 2^8)$	$(7 \text{ or } 8)^3 \times (2^{12} \times 2^8)/f_{xx} = 1.75\text{--}2 \text{ sec}$
x011b	$f_{xx}/2^9$	$f_{xx}/(2^9 \times 2^8)$	$(7 \text{ or } 8)^3 \times (2^9 \times 2^8)/f_{xx} = 218.7\text{--}250 \text{ ms}$
x101b	$f_{xx}/2^7$	$f_{xx}/(2^7 \times 2^8)$	$(7 \text{ or } 8)^3 \times (2^7 \times 2^8)/f_{xx} = 54.6\text{--}62.5 \text{ ms}$
x111b	$f_{xx}/2^5$	$f_{xx}/(2^5 \times 2^8)$	$(7 \text{ or } 8)^3 \times (2^5 \times 2^8)/f_{xx} = 13.6\text{--}15.6 \text{ ms}$

NOTES:

1. Clock frequencies assume a system oscillator clock frequency (f_{xx}) of 4.19 MHz.
2. f_{xx} = system clock frequency.

 **PROGRAMMING TIP — Using the Watchdog Timer**

```

RESET      DI
           LD      EA,#00H
           LD      SP,EA
           .
           .
           .
           LD      A,#0DH          ; WDCNT input clock is 7.82 ms
           LD      BMOD,A
           .
           .
           .
MAIN       BITS      WDTCF          ; Main routine operation period must be shorter than
           .                    ; watchdog-timer's period
           .
           .
           JP      MAIN

```

8-BIT TIMER/COUNTER 0 (TC0)

OVERVIEW

Timer/counter 0 (TC0) is used to count system 'events' by identifying the transition (high-to-low or low-to-high) of incoming square wave signals. To indicate that an event has occurred, or that a specified time interval has elapsed, TC0 generates an interrupt request. By counting signal transitions and comparing the current counter value with the reference register value, TC0 can be used to measure specific time intervals.

TC0 has a reloadable counter that consists of two parts: an 8-bit reference register (TREF0) into which you write the counter reference value, and an 8-bit counter register (TCNT0) whose value is automatically incremented by counter logic.

An 8-bit mode register, TMOD0, is used to activate the timer/counter and to select the basic clock frequency to be used for timer/counter operations. To dynamically modify the basic frequency, new values can be loaded into the TMOD0 register during program execution.

TC0 FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
External event counter	Counts various system "events" based on edge detection of external clock signals at the TC0 input pin, TCL0. To start the event counting operation, TMOD0.2 is set to "1" and TMOD0.6 is cleared to "0".
Arbitrary frequency output	Outputs selectable clock frequencies to the TC0 output pin, TCLO0.
External signal divider	Divides the frequency of an incoming external clock signal according to a modifiable reference value (TREF0), and outputs the modified frequency to the TCLO0 pin.

TC0 COMPONENT SUMMARY

Mode register (TMOD0)	Activates the timer/counter and selects the internal clock frequency or the external clock source at the TCLK pin.
Reference register (TREF0)	Stores the reference value for the desired number of clock pulses between interrupt requests.
Counter register (TCNT0)	Counts internal or external clock pulses based on the bit settings in TMOD0 and TREF0.
Clock selector circuit	Together with the mode register (TMOD0), lets you select one of four internal clock frequencies or an external clock.
8-bit comparator	Determines when to generate an interrupt by comparing the current value of the counter register (TCNT0) with the reference value previously programmed into the reference register (TREF0).
Output enable flag (TOE0)	Must be set to logic one before the contents of the TOL0 latch can be output to TCLK0.
Interrupt request flag (IRQT0)	Cleared when TC0 operation starts and the TC0 interrupt service routine is executed and enabled whenever the counter value and reference value coincide.
Interrupt enable flag (IET0)	Must be set to logic one before the interrupt requests generated by timer/counter 0 can be processed.

Table 11-4. TC0 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0	Control	Controls TC0 enable/disable (bit 2); clears and resumes counting operation (bit 3); sets input clock and clock frequency (bits 6-4)	8-bit	F90H-F91H	8-bit write-only; (TMOD0.3 is also 1-bit writeable)	"0"
TCNT0	Counter	Counts clock pulses matching the TMOD0 frequency setting	8-bit	F94H-F95H	8-bit read-only	"0"
TREF0	Reference	Stores reference value for the timer/counter 0 interval setting	8-bit	F96H-F97H	8-bit write-only	FFH
TOE0	Flag	Controls timer/counter 0 output to the TCLK0 pin	1-bit	F92H.2	1-bit and 4-bit read/write	"0"

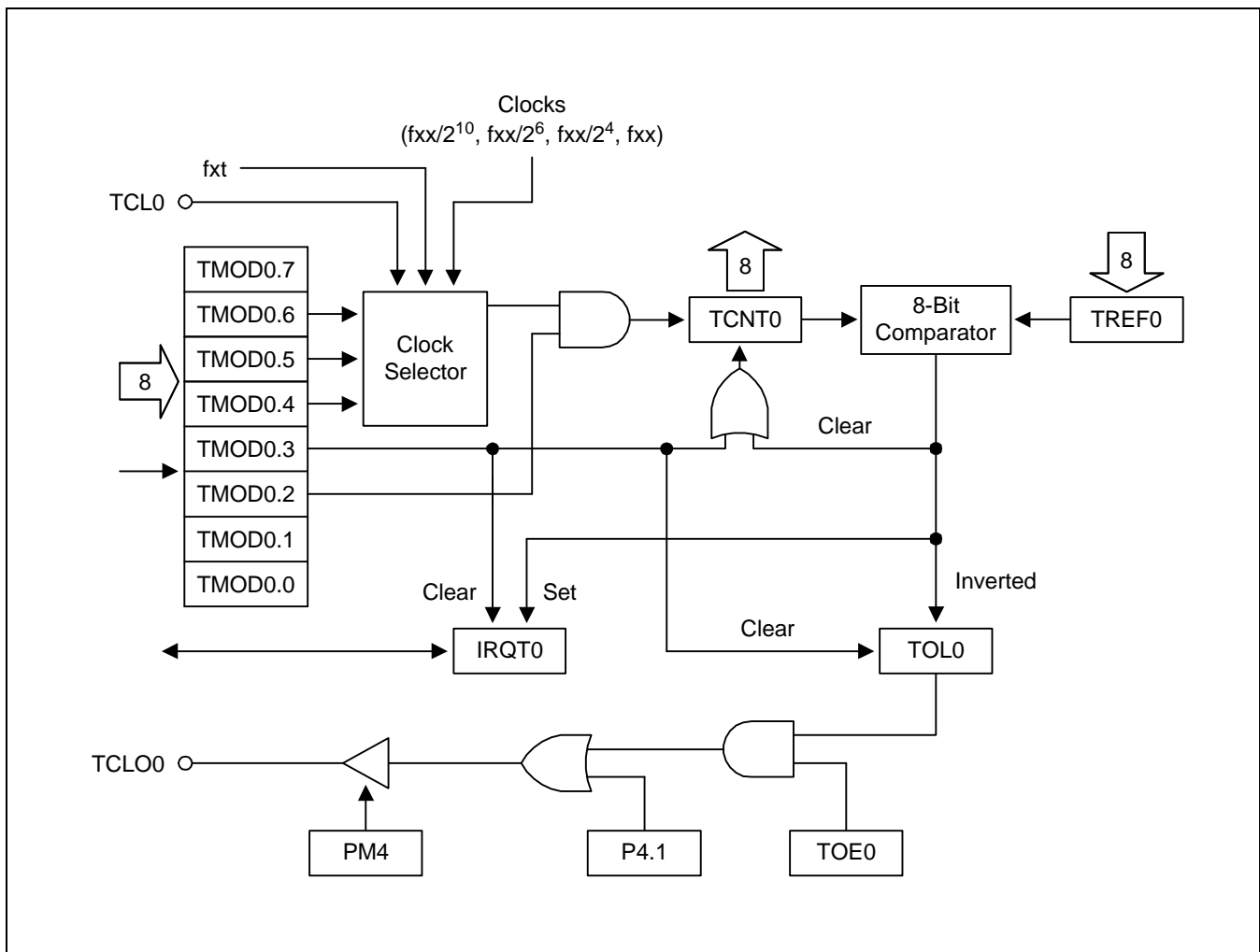


Figure 11-2. TC0 Circuit Diagram

TC0 ENABLE/DISABLE PROCEDURE

Enable Timer/Counter 0

- Set TMOD0.2 to logic one
- Set the TC0 interrupt enable flag IET0 to logic one
- Set TMOD0.3 to logic one

TCNT0, IRQT0, and TOL0 are cleared to logic zero, and timer/counter operation starts.

Disable Timer/Counter 0

- Set TMOD0.2 to logic zero

Clock signal input to the counter register TCNT0 is halted. The current TCNT0 value is retained and can be read if necessary.

TC0 PROGRAMMABLE TIMER/COUNTER FUNCTION

Timer/counter 0 can be programmed to generate interrupt requests at various intervals based on the selected system clock frequency. Its 8-bit TC0 mode register TMOD0 is used to activate the timer/counter and to select the clock frequency. The reference register TREF0 stores the value for the number of clock pulses to be generated between interrupt requests. The counter register, TCNT0, counts the incoming clock pulses, which are compared to the TREF0 value as TCNT0 is incremented. When there is a match ($TREF0 = TCNT0$), an interrupt request is generated.

To program timer/counter 0 to generate interrupt requests at specific intervals, choose one of five internal clock frequencies (divisions of the system clock, f_{xx} , f_{xt}) and load a counter reference value into the TREF0 register. TCNT0 is incremented each time an internal counter pulse is detected with the reference clock frequency specified by TMOD0.4-TMOD0.6 settings. To generate an interrupt request, the TC0 interrupt request flag (IRQT0) is set to logic one, the status of TOL0 is inverted, and the interrupt is generated. The content of TCNT0 is then cleared to 00H and TC0 continues counting. The interrupt request mechanism for TC0 includes an interrupt enable flag (IET0) and an interrupt request flag (IRQT0).

TC0 OPERATION SEQUENCE

The general sequence of operations for using TC0 can be summarized as follows:

1. Set TMOD0.2 to "1" to enable TC0
2. Set TMOD0.6 to "1" to enable the system clock (f_{xx}) input
3. Set TMOD0.5 and TMOD0.4 bits to desired internal frequency ($f_{xx}/2^n$)
4. Load a value to TREF0 to specify the interval between interrupt requests
5. Set the TC0 interrupt enable flag (IET0) to "1"
6. Set TMOD0.3 bit to "1" to clear TCNT0, IRQT0, and TOL0 and start counting
7. TCNT0 increments with each internal clock pulse
8. When the comparator shows $TCNT0 = TREF0$, the IRQT0 flag is set to "1", and an interrupt request is generated.
9. Output latch (TOL0) logic toggles high or low
10. TCNT0 is cleared to 00H and counting resumes
11. Programmable timer/counter operation continues until TMOD0.2 is cleared to "0".

TC0 EVENT COUNTER FUNCTION

Timer/counter 0 can monitor or detect system 'events' by using the external clock input at the TCL0 pin (I/O port 4.0) as the counter source. The TC0 mode register selects rising or falling edge detection for incoming clock signals. The counter register TCNT0 is incremented each time the selected state transition of the external clock signal occurs.

With the exception of the different TMOD0.4-TMOD0.6 settings, the operation sequence for TC0's event counter function is identical to its programmable timer/counter function. To activate the TC0 event counter function,

- Set TMOD0.2 to "1" to enable TC0;
- Clear TMOD0.6 to "0" to select the external clock source at the TCL0 pin;
- Select TCL0 edge detection for rising or falling signal edges by loading the appropriate values to TMOD0.5 and TMOD0.4.
- P4.0 must be set to input mode.

Table 11-5. TMOD0 Settings for TCL0 Edge Detection

TMOD0.5	TMOD0.4	TCL0 Edge Detection
0	0	Rising edges
0	1	Falling edges

NOTE: If you set P4.0 to a open-drain, you can use P4.0 as TCL0 pin for external TCO clock, even if P4.0 is set to output mode.

TC0 CLOCK FREQUENCY OUTPUT

Using timer/counter 0, a modifiable clock frequency can be output to the TC0 clock output pin, TCLO0. To select the clock frequency, load the appropriate values to the TC0 mode register, TMOD0. The clock interval is selected by loading the desired reference value into the reference register TREF0. To enable the output to the TCLO0 pin at I/O port 4.1, the following conditions must be met:

- TC0 output enable flag TOE0 must be set to "1"
- I/O mode flag for P4.1 (PM4) must be set to output mode ("1")
- Output latch value for P4.1 must be set to "0"

In summary, the operational sequence required to output a TC0-generated clock signal to the TCLO0 pin is as follows:

1. Load a reference value to TREF0.
2. Set the internal clock frequency in TMOD0.
3. Initiate TC0 clock output to TCLO0 (TMOD0.2 = "1").
4. Set port 4 mode flag (PM4) to "1".
5. Set P4.1 output latch to "0".
6. Set TOE0 flag to "1".

Each time TCNT0 overflows and an interrupt request is generated, the state of the output latch TOL0 is inverted and the TC0-generated clock signal is output to the TCLO0 pin.

PROGRAMMING TIP — TC0 Signal Output to the TCLO0 Pin

Output a 30 ms pulse width signal to the TCLO0 pin (at 4.19 MHz):

```

BITS      EMB
SMB       15
LD        EA,#79H
LD        TREF0,EA
LD        EA,#4CH
LD        TMOD0,EA
LD        A,#1H
LD        PMG2,A           ; P4.1← output mode
BITR      P4.1             ; P4.1clear
BITS      TOE0

```

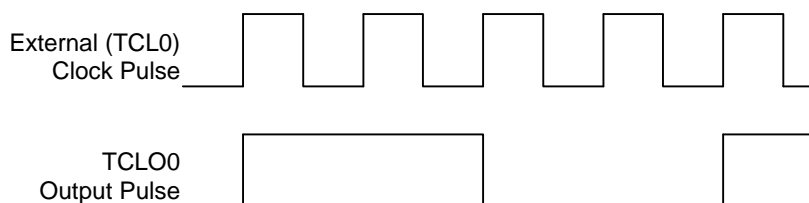
TC0 EXTERNAL INPUT SIGNAL DIVIDER

By selecting an external clock source and loading a reference value into the TC0 reference register, TREF0, you can divide the incoming clock signal by the TREF0 value and then output this modified clock frequency to the TCLO0 pin. The sequence of operations used to divide external clock input can be summarized as follows:

1. Load a signal divider value to the TREF0 register
2. Clear TMOD0.6 to "0" to enable external clock input at the TCL0 pin
3. Set TMOD0.5 and TMOD0.4 to desired TCL0 signal edge detection
4. Set port 4 mode flag (PM4) to output ("1")
5. Set P4.1 output latch to "0"
6. Set TOE0 flag to "1" to enable output of the divided frequency to the TCLO0 pin

 **PROGRAMMING TIP — External TCL0 Clock Output to the TCLO0 Pin**

Output external TCL0 clock pulse to the TCLO0 pin (divided by four):



```

BITS      EMB
SMB      15
LD        EA,#01H
LD        TREF0,EA
LD        EA,#0CH
LD        TMOD0,EA
LD        A,#1H
LD        PMG2,A           ; P4.1 ← output mode
BITR      P4.1             ; P4.1 clear
BITS      TOE0

```

NOTE: The Port 4.0 must be a open-drain pin for external TC0 clock input to the TCL0 pin, when the Port 4 is set to output mode.

TC0 MODE REGISTER (TMOD0)

TMOD0 is the 8-bit mode control register for timer/counter 0. It is addressable by 8-bit write instructions. One bit, TMOD0.3, is also 1-bit writeable. RESET clears all TMOD0 bits to logic zero and disables TC0 operations.

F90H	TMOD0.3	TMOD0.2	"0"	"0"
F91H	"0"	TMOD0.6	TMOD0.5	TMOD0.4

TMOD0.2 is the enable/disable bit for timer/counter 0. When TMOD0.3 is set to "1", the contents of TCNT0, IRQT0, and TOL0 are cleared, counting starts from 00H, and TMOD0.3 is automatically reset to "0" for normal TC0 operation. When TC0 operation stops (TMOD0.2 = "0"), the contents of the TC0 counter register TCNT0 are retained until TC0 is re-enabled.

The TMOD0.6, TMOD0.5, and TMOD0.4 bit settings are used together to select the TC0 clock source. This selection involves two variables:

- Synchronization of timer/counter operations with either the rising edge or the falling edge of the clock signal input at the TCLK pin, and
- Selection of one of four frequencies, based on division of the incoming system clock frequency, for use in internal TC0 operation.

Table 11-6. TC0 Mode Register (TMOD0) Organization

Bit Name	Setting	Resulting TC0 Function	Address
TMOD0.7	0	Always logic zero	F91H
TMOD0.6 TMOD0.5 TMOD0.4	0,1	Specify input clock edge and internal frequency	
TMOD0.3	1	Clear TCNT0, IRQT0, and TOL0 and resume counting immediately (This bit is automatically cleared to logic zero immediately after counting resumes.)	
TMOD0.2	0	Disable timer/counter 0; retain TCNT0 contents	F90H
	1	Enable timer/counter 0	
TMOD0.1	0	Always logic zero	
TMOD0.0	0	Always logic zero	

Table 11-7. TMOD0.6, TMOD0.5, and TMOD0.4 Bit Settings

TMOD0.6	TMOD0.5	TMOD0.4	Resulting Counter Source and Clock Frequency
0	0	0	External clock input (TCL0) on rising edges
0	0	1	External clock input (TCL0) on falling edges
0	1	x	fxt (Subsystem clock: 32.768 kHz)
1	0	0	$f_{xx}/2^{10}$ (4.09 kHz)
1	0	1	$f_{xx}/2^6$ (65.5 kHz)
1	1	0	$f_{xx}/2^4$ (262 kHz)
1	1	1	f_{xx} (4.19 MHz)

NOTE: 'fxx' = selected system clock of 4.19 MHz.

PROGRAMMING TIP — Restarting TC0 Counting Operation

1. Set TC0 timer interval to 4.09 kHz:

```

BITS    EMB
SMB     15
LD      EA,#4CH
LD      TMOD0,EA
EI
BITS    IET0

```

2. Clear TCNT0, IRQT0, and TOL0 and restart TC0 counting operation:

```

BITS    EMB
SMB     15
BITS    TMOD0.3

```

TC0 COUNTER REGISTER (TCNT0)

The 8-bit counter register for timer/counter 0, TCNT0, is read-only and can be addressed by 8-bit RAM control instructions. RESET sets all TCNT0 register values to logic zero (00H).

Whenever TMODE.3 is enabled, TCNT0 is cleared to logic zero and counting resumes. The TCNT0 register value is incremented each time an incoming clock signal is detected that matches the signal edge and frequency setting of the TMODE register (specifically, TMODE.6, TMODE.5, and TMODE.4).

Each time TCNT0 is incremented, the new value is compared to the reference value stored in the TC0 reference buffer, TREF0. When TCNT0 = TREF0, an overflow occurs in the TCNT0 register, the interrupt request flag, IRQT0, is set to logic one, and an interrupt request is generated to indicate that the specified timer/counter interval has elapsed.

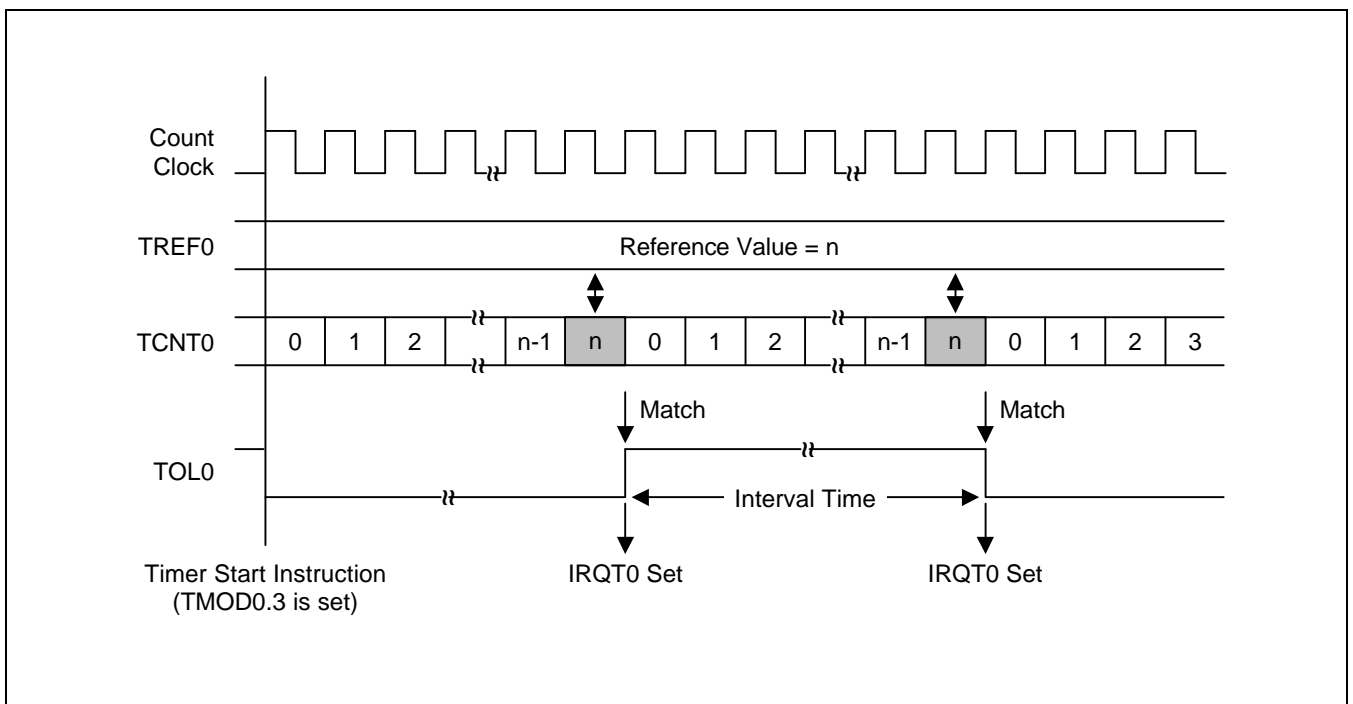


Figure 11-3. TC0 Timing Diagram

TC0 REFERENCE REGISTER (TREF0)

The TC0 reference register TREF0 is an 8-bit write-only register. It is addressable by 8-bit RAM control instructions. RESET initializes the TREF0 value to 'FFH'.

TREF0 is used to store a reference value to be compared to the incrementing TCNT0 register in order to identify an elapsed time interval. Reference values will differ depending upon the specific function that TC0 is being used to perform — as a programmable timer/counter, event counter, clock signal divider, or arbitrary frequency output source.

During timer/counter operation, the value loaded into the reference register is compared to the TCNT0 value. When TCNT0 = TREF0, the TC0 output latch (TOL0) is inverted and an interrupt request is generated to signal the interval or event. The TREF0 value, together with the TMOD0 clock frequency selection, determines the specific TC0 timer interval. Use the following formula to calculate the correct value to load to the TREF0 reference register:

$$\text{TC0 timer interval} = (\text{TREF0 value} + 1) \times \frac{1}{\text{TMOD0 frequency setting}}$$

(TREF0 value ≠ 0)

TC0 OUTPUT ENABLE FLAG (TOE0)

The 1-bit timer/counter 0 output enable flag TOE0 controls output from timer/counter 0 to the TCLO0 pin. TOE0 is addressable by 1-bit and 4-bit read/write instruction.

	(MSB)		TOE0	"U"	"0"	(LSB)	
F92H							

NOTE: The "U" means a undefined register bit.

When you set the TOE0 flag to "1", the contents of TOL0 can be output to the TCLO0 pin. Whenever a RESET occurs, TOE0 is automatically set to logic zero, disabling all TC0 output. Even when the TOE0 flag is disabled, timer/counter 0 can continue to output an internally generated clock frequency, via TOL0.

TC0 OUTPUT LATCH (TOL0)

TOL0 is the output latch for timer/counter 0. When the 8-bit comparator detects a correspondence between the value of the counter register TCNT0 and the reference value stored in the TREF0 register, the TOL0 value is inverted — the latch toggles high-to-low or low-to-high. Whenever the state of TOL0 is switched, the TC0 signal is output. TC0 output may be directed to the TCLO0 pin.

Assuming TC0 is enabled, when bit 3 of the TMOD0 register is set to "1", the TOL0 latch is cleared to logic zero, along with the counter register TCNT0 and the interrupt request flag, IRQT0, and counting resumes immediately. When TC0 is disabled (TMOD0.2 = "0"), the contents of the TOL0 latch are retained and can be read, if necessary.

PROGRAMMING TIP — Setting a TC0 Timer Interval

To set a 30 ms timer interval for TC0, given $f_{xx} = 4.19$ MHz, follow these steps.

1. Select the timer/counter 0 mode register with a maximum setup time of 62.5ms (assume the TC0 counter clock = $f_{xx}/2^{10}$, and TREF0 is set to FFH):
2. Calculate the TREF0 value:

$$30 \text{ ms} = \frac{\text{TREF0 value} + 1}{4.09 \text{ kHz}}$$

$$\text{TREF0} + 1 = \frac{30 \text{ ms}}{244 \mu\text{s}} = 122.9 = 7AH$$

$$\text{TREF0 value} = 7AH - 1 = 79H$$

3. Load the value 79H to the TREF0 register:

BITS	EMB
SMB	15
LD	EA,#79H
LD	TREF0,EA
LD	EA,#4CH
LD	TMOD0,EA

8-BIT TIMER/COUNTER 1 (TC1)

OVERVIEW

Timer/counter 1 (TC1) is used to count system 'events' by identifying the transition (high-to-low or low-to-high) of incoming square wave signals. To indicate that an event has occurred, or that a specified time interval has elapsed, TC1 generates an interrupt request. By counting signal transitions and comparing the current counter value with the reference register value, TC1 can be used to measure specific time intervals.

TC1 has a reloadable counter that consists of two parts: an 8-bit reference register (TREF1) into which you write the counter reference value, and an 8-bit counter register (TCNT1) whose value is automatically incremented by counter logic.

An 8-bit mode register, TMOD1, is used to activate the timer/counter and to select the basic clock frequency to be used for timer/counter operations. To dynamically modify the basic frequency, new values can be loaded into the TMOD1 register during program execution.

TC1 FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
--------------------------	--

TC1 COMPONENT SUMMARY

Mode register (TMOD1)	Activates the timer/counter and selects the internal clock frequency.
Reference register (TREF1)	Stores the reference value for the desired number of clock pulses between interrupt requests.
Counter register (TCNT1)	Counts internal or external clock pulses based on the bit settings in TMOD1 and TREF1.
Clock selector circuit	Together with the mode register (TMOD1), lets you select one of four internal clock frequencies or an external clock.
8-bit comparator	Determines when to generate an interrupt by comparing the current value of the counter register (TCNT1) with the reference value previously programmed into the reference register (TREF1).
Interrupt request flag (IRQT1)	Cleared when TC1 operation starts and the TC1 interrupt service routine is executed and enabled whenever the counter value and reference value coincide.
Interrupt enable flag (IET1)	Must be set to logic one before the interrupt requests generated by timer/counter 1 can be processed.

Table 11-8. TC1 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD1	Control	Controls TC1 enable/disable (bit 2); clears and resumes counting operation (bit 3); sets input clock and clock frequency (bits 6-4)	8-bit	FA6H-FA7H	8-bit write-only; (TMOD1.3 is also 1-bit writeable)	"0"
TCNT1	Counter	Counts clock pulses matching the TMOD1 frequency setting	8-bit	FA8H-FA9H	8-bit read-only	"0"
TREF1	Reference	Stores reference value for the timer/counter 1 interval setting	8-bit	FAAH-FABH	8-bit write-only	FFH

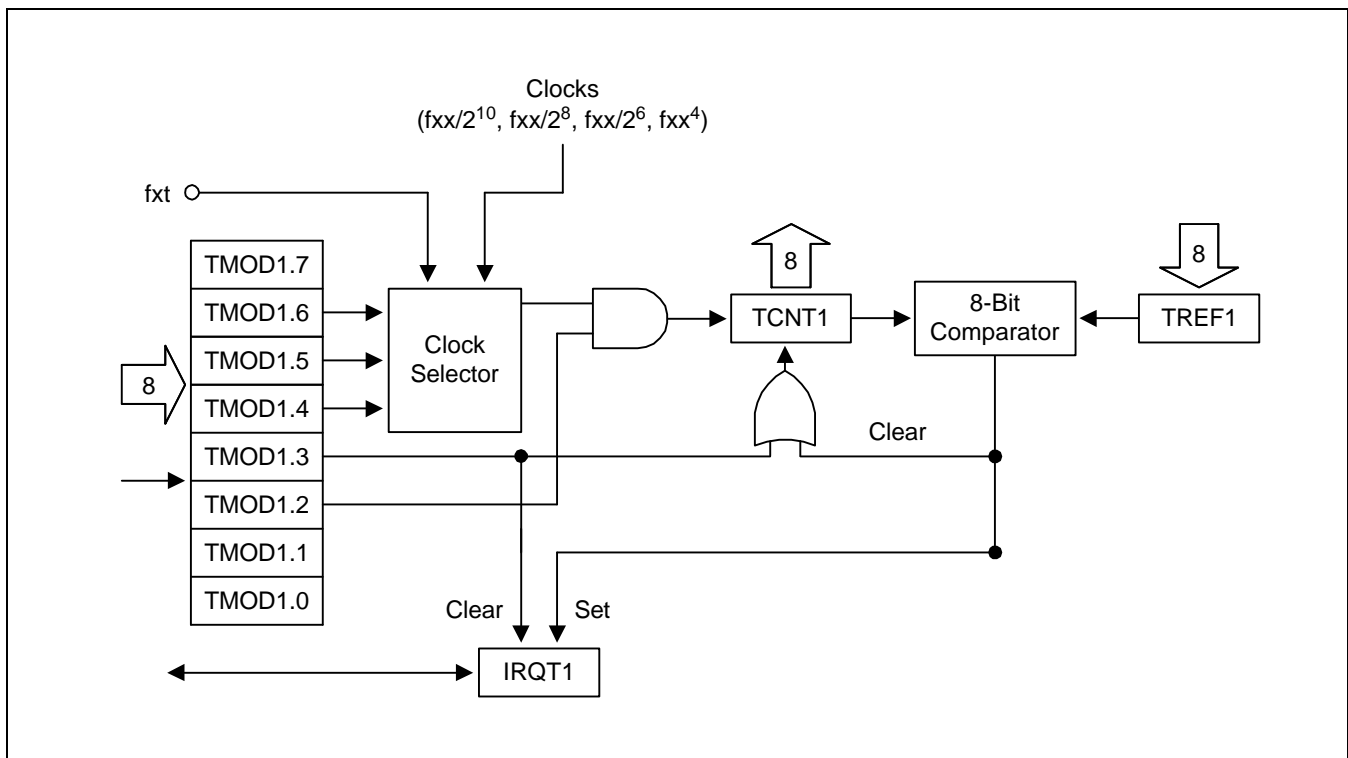


Figure 11-4. TC1 Circuit Diagram

TC1 ENABLE/DISABLE PROCEDURE

Enable Timer/Counter 1

- Set TMOD1.2 to logic one
- Set the TC1 interrupt enable flag IET1 to logic one
- Set TMOD1.3 to logic one

TCNT1, IRQT1, and TOL1 are cleared to logic zero, and timer/counter operation starts.

Disable Timer/Counter 1

- Set TMOD1.2 to logic zero

Clock signal input to the counter register TCNT1 is halted. The current TCNT1 value is retained and can be read if necessary.

TC1 PROGRAMMABLE TIMER/COUNTER FUNCTION

Timer/counter 1 can be programmed to generate interrupt requests at various intervals based on the selected system clock frequency. Its 8-bit TC1 mode register TMOD1 is used to activate the timer/counter and to select the clock frequency. The reference register TREF1 stores the value for the number of clock pulses to be generated between interrupt requests. The counter register, TCNT1, counts the incoming clock pulses, which are compared to the TREF1 value as TCNT1 is incremented. When there is a match ($TREF1 = TCNT1$), an interrupt request is generated.

To program timer/counter 1 to generate interrupt requests at specific intervals, choose one of five internal clock frequencies (divisions of the system clock, fxx,fxt) and load a counter reference value into the TREF1 register. TCNT1 is incremented each time an internal counter pulse is detected with the reference clock frequency specified by TMOD1.4-TMOD1.6 settings. To generate an interrupt request, the TC1 interrupt request flag (IRQT1) is set to logic one, and the interrupt is generated. The content of TCNT1 is then cleared to 00H and TC1 continues counting. The interrupt request mechanism for TC1 includes an interrupt enable flag (IET1) and an interrupt request flag (IRQT1).

TC1 OPERATION SEQUENCE

The general sequence of operations for using TC1 can be summarized as follows:

1. Set TMOD1.2 to "1" to enable TC1
2. Set TMOD1.6 to "1" to enable the system clock (fxx) input.
3. Set TMOD1.5 and TMOD1.4 bits to desired internal frequency ($f_{xx}/2^n$) or TMOD1.6 and TMOD1.5 to "1" to enable the system clock fxt input.
4. Load a value to TREF1 to specify the interval between interrupt requests
5. Set the TC1 interrupt enable flag (IET1) to "1"
6. Set TMOD1.3 bit to "1" to clear TCNT1, IRQT1, and TOL1 and start counting
7. TCNT1 increments with each internal clock pulse
8. When the comparator shows $TCNT1 = TREF1$, the IRQT1 flag is set to "1", and an interrupt request is generated.
9. TCNT1 is cleared to 00H and counting resumes
10. Programmable timer/counter operation continues until TMOD1.2 is cleared to "0".

TC1 MODE REGISTER (TMOD1)

TMOD1 is the 8-bit mode control register for timer/counter 1. It is addressable by 8-bit write instructions. One bit, TMOD1.3, is also 1-bit writeable. RESET clears all TMOD1 bits to logic zero and disables TC1 operations.

FA6H	TMOD1.3	TMOD1.2	"0"	"0"
FA7H	"0"	TMOD1.6	TMOD1.5	TMOD1.4

TMOD1.2 is the enable/disable bit for timer/counter 1. When TMOD1.3 is set to "1", the contents of TCNT1, and IRQT1, are cleared, counting starts from 00H, and TMOD1.3 is automatically reset to "0" for normal TC1 operation. When TC1 operation stops (TMOD1.2 = "0"), the contents of the TC1 counter register TCNT1 are retained until TC1 is re-enabled.

The TMOD1.6, TMOD1.5, and TMOD1.4 bit settings are used together to select the TC1 clock source. This selection involves:

- Selection of one of five frequencies, based on division of the incoming system clock frequency, or subsystem clock frequency, for use in internal TC1 operation.

Table 11-9. TC1 Mode Register (TMOD1) Organization

Bit Name	Setting	Resulting TC1 Function	Address
TMOD1.7	0	Always logic zero	FA7H
TMOD1.6 TMOD1.5 TMOD1.4	0,1	Specify input clock edge and internal frequency	
TMOD1.3	1	Clear TCNT1, and IRQT1, and resume counting immediately (This bit is automatically cleared to logic zero immediately after counting resumes.)	
TMOD1.2	0	Disable timer/counter 1; retain TCNT1 contents	FA6H
	1	Enable timer/counter 1	
TMOD1.1	0	Always logic zero	
TMOD1.1	0	Always logic zero	

Table 11-10. TMOD1.6, TMOD1.5, and TMOD1.4 Bit Settings

TMOD1.6	TMOD1.5	TMOD1.4	Resulting Counter Source and Clock Frequency
0	1	x	fxt (Subsystem clock: 32.768 kHz)
1	0	0	$f_{xx}/2^{10}$ (4.09 kHz)
1	0	1	$f_{xx}/2^6$ (65.5 kHz)
1	1	0	$f_{xx}/2^4$ (262 kHz)
1	1	1	f_{xx} (4.19 MHz)

NOTE: 'fxx' = selected system clock of 4.19 MHz.

PROGRAMMING TIP — Restarting TC1 Counting Operation

1. Set TC1 timer interval to 4.09 kHz:

```

BITS    EMB
SMB     15
LD      EA,#4CH
LD      TMOD1,EA
EI
BITS    IET1

```

2. Clear TCNT1, and IRQT1, and restart TC1 counting operation:

```

BITS    EMB
SMB     15
BITS    TMOD1.3

```

TC1 COUNTER REGISTER (TCNT1)

The 8-bit counter register for timer/counter 1, TCNT1, is read-only and can be addressed by 8-bit RAM control instructions. RESET sets all TCNT1 register values to logic zero (00H).

Whenever TMOD1.3 is enabled, TCNT1 is cleared to logic zero and counting resumes. The TCNT1 register value is incremented each time an incoming clock signal is detected that matches the signal edge and frequency setting of the TMOD1 register (specifically, TMOD1.6, TMOD1.5, and TMOD1.4).

Each time TCNT1 is incremented, the new value is compared to the reference value stored in the TC1 reference buffer, TREF1. When TCNT1 = TREF1, an overflow occurs in the TCNT1 register, the interrupt request flag, IRQT1, is set to logic one, and an interrupt request is generated to indicate that the specified timer/counter interval has elapsed.

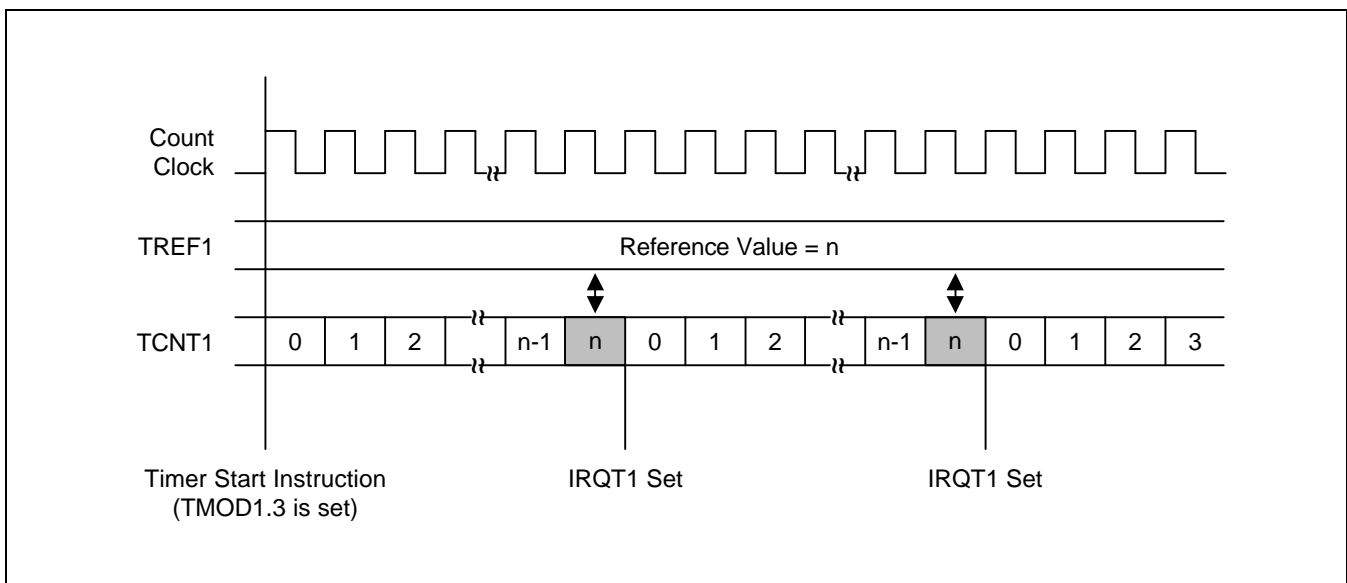


Figure 11-5. TC1 Timing Diagram

TC1 REFERENCE REGISTER (TREF1)

The TC1 reference register TREF1 is an 8-bit write-only register. It is addressable by 8-bit RAM control instructions. RESET initializes the TREF1 value to 'FFH'.

TREF1 is used to store a reference value to be compared to the incrementing TCNT1 register in order to identify an elapsed time interval. Reference values will differ depending upon the specific function that TC1 is being used to perform — as a programmable timer/counter.

During timer/counter operation, the value loaded into the reference register is compared to the TCNT1 value. When TCNT1 = TREF1 an interrupt request is generated to signal the interval. The TREF1 value, together with the TMOD1 clock frequency selection, determines the specific TC1 timer interval. Use the following formula to calculate the correct value to load to the TREF1 reference register:

$$\text{TC1 timer interval} = (\text{TREF1 value} + 1) \times \frac{1}{\text{TMOD1 frequency setting}}$$

(TREF1 value \neq 1)

PROGRAMMING TIP — Setting a TC1 Timer Interval

To set a 30 ms timer interval for TC1, given $f_{\text{xx}} = 4.19 \text{ MHz}$, follow these steps.

1. Select the timer/counter 1 mode register with a maximum setup time of 62.5ms (assume the TC1 counter clock = $f_{\text{xx}}/2^{10}$, and TREF1 is set to FFH):
2. Calculate the TREF0 value:

$$30 \text{ ms} = \frac{\text{TREF1 value} + 1}{4.09 \text{ kHz}}$$

$$\text{TREF1} + 1 = \frac{30 \text{ ms}}{244 \mu\text{s}} = 122.9 = 7\text{AH}$$

$$\text{TREF1 value} = 7\text{AH} - 1 = 79\text{H}$$

3. Load the value 79H to the TREF1 register:

BITS	EMB
SMB	15
LD	EA,#79H
LD	TREF1,EA
LD	EA,#4CH
LD	TMOD1,EA

WATCH TIMER

OVERVIEW

The watch timer is a multi-purpose timer which consists of three basic components:

- 8-bit watch timer mode register (WMOD)
- Clock selector
- Frequency divider circuit

Watch timer functions include real-time and watch-time measurement and interval timing for the main and sub-system clock. It is also used as a clock source for the LCD controller and for generating buzzer (BUZ) output.

Real-Time and Watch-Time Measurement

To start watch timer operation, set bit 2 of the watch timer mode register (WMOD.2) to logic one. The watch timer starts, the interrupt request flag IRQW is automatically set to logic one, and interrupt requests commence in 0.5-second intervals.

Since the watch timer functions as a quasi-interrupt instead of a vectored interrupt, the IRQW flag should be cleared to logic zero by program software as soon as a requested interrupt service routine has been executed.

Using a Main System or Subsystem Clock Source

The watch timer can generate interrupts based on the main system clock frequency or on the subsystem clock. When the zero bit of the WMOD register is set to "1", the watch timer uses the subsystem clock signal (fxt) as its source; if WMOD.0 = "0", the main system clock (fx) is used as the signal source, according to the following formula:

$$\text{Watch timer clock (fw)} = \frac{\text{Main system clock (fx)}}{128} = 32.768 \text{ kHz (fx = 4.19 MHz)}$$

This feature is useful for controlling timer-related operations during stop mode. When stop mode is engaged, the main system clock (fx) is halted, but the subsystem clock continues to oscillate. By using the subsystem clock as the oscillation source during stop mode, the watch timer can set the interrupt request flag IRQW to "1", thereby releasing stop mode.

Buzzer Output Frequency Generator

The watch timer can generate a steady 2 kHz, 4 kHz, 8 kHz, or 16 kHz signal to the BUZ pin at selected clock for watch timer. To select the desired BUZ frequency, load the appropriate value to the WMOD register. This output can then be used to actuate an external buzzer sound. To generate a BUZ signal, three conditions must be met:

- The WMOD.7 register bit is set to "1"
- The output latch for I/O port 5.2 is cleared to "0"
- The port 5.2 output mode flag (PM5.2) set to 'output' mode

Timing Tests in High-Speed Mode

By setting WMOD.1 to "1", the watch timer will operate in high-speed mode, generating an interrupt every 3.91 ms at oscillation clock of 4.19 MHz. At its normal speed (WMOD.1 = '0'), the watch timer generates an interrupt request every 0.5 seconds. High-speed mode is useful for timing events for program debugging sequences.

Check Subsystem Clock Level Feature

The watch timer can also check the input level of the subsystem clock by testing WMOD.3. If WMOD.3 is "1", the input level at the XT_{in} pin is high; if WMOD.3 is "0", the input level at the XT_{in} pin is low.

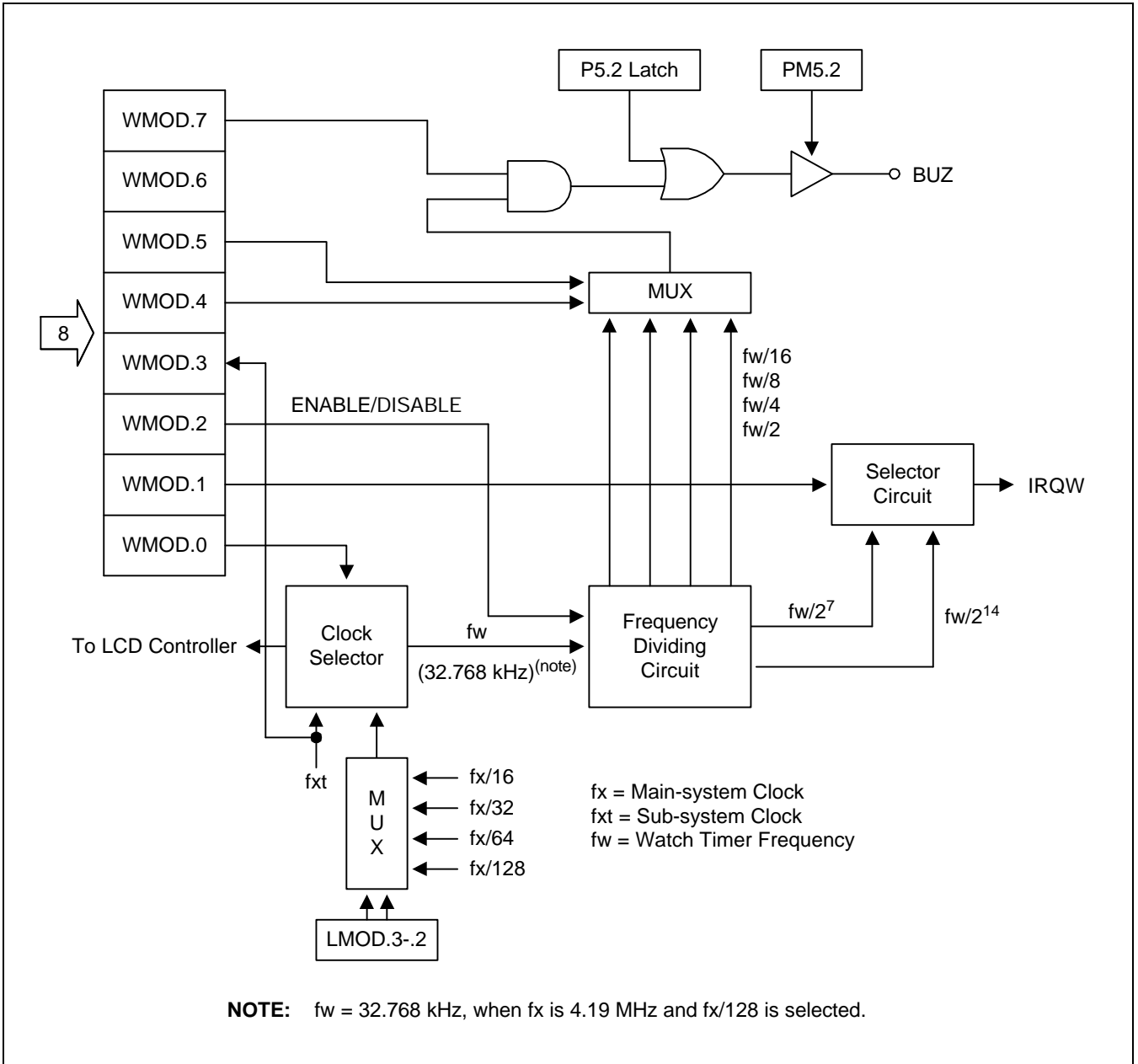


Figure 11-6. Watch Timer Circuit Diagram

WATCH TIMER MODE REGISTER (WMOD)

The watch timer mode register WMOD is used to select specific watch timer operations. It is 8-bit write-only addressable. An exception is WMOD bit 3 (the XT_{IN} input level control bit) which is 1-bit read-only addressable. A RESET automatically sets WMOD.3 to the current input level of the subsystem clock, XT_{IN} (high, if logic one; low, if logic zero), and all other WMOD bits to logic zero.

F88H	WMOD.3	WMOD.2	WMOD.1	WMOD.0
F89H	WMOD.7	"0"	WMOD.5	WMOD.4

In summary, WMOD settings control the following watch timer functions:

- Watch timer clock and LCD clock selection (WMOD.0)
- Watch timer speed control (WMOD.1)
- Enable/disable watch timer (WMOD.2)
- XT_{IN} input level control (WMOD.3)
- Buzzer frequency selection (WMOD.4 and WMOD.5)
- Enable/disable buzzer output (WMOD.7)

Table 11-11. Watch Timer Mode Register (WMOD) Organization

Bit Name	Values	Function	Address	
WMOD.7	0	Disable buzzer (BUZ) signal output at the BUZ pin	F89H	
	1	Enable buzzer (BUZ) signal output at the BUZ pin		
WMOD.6	0	Always logic zero	F88H	
WMOD.5 – .4	0	0		2 kHz buzzer (BUZ) signal output
	0	1		4 kHz buzzer (BUZ) signal output
	1	0		8 kHz buzzer (BUZ) signal output
	1	1		16 kHz buzzer (BUZ) signal output
WMOD.3	0	Input level to XT _{in} pin is low	F88H	
	1	Input level to XT _{in} pin is high		
WMOD.2	0	Disable watch timer; clear frequency dividing circuits	F88H	
	1	Enable watch timer		
WMOD.1	0	Normal mode; sets IRQW to 0.5 second		F88H
	1	High-speed mode; sets IRQW to 3.91 ms		
WMOD.0	0	Select (fx/128) as the watch timer clock (fw) Select a LCD clock source as main system clock		F88H
	1	Select subsystem clock as watch timer clock (fw) Select a LCD clock source as sub system clock		

NOTE: Main system clock frequency (fx) is assumed to be 4.19 MHz; subsystem clock (fxx) is assumed to be 32.768 kHz.

 **PROGRAMMING TIP — Using the Watch Timer**

1. Select a subsystem clock as the LCD display clock, a 0.5 second interrupt, and 2 kHz buzzer enable:

```

BITS      EMB
SMB       15
LD        EA,#40H
LD        PMG1,EA      ; P5.2 ← output mode
BITR      P5.2
LD        EA,#85H
LD        WMOD,EA
BITS      IEW

```

2. Sample real-time clock processing method:

```

CLOCK     BTSTZ      IRQW      ; 0.5 second check
          RET        ; No, return
          •          ; Yes, 0.5 second interrupt generation
          •
          •          ; Increment HOUR, MINUTE, SECOND

```

NOTES

12 LCD CONTROLLER/DRIVER

OVERVIEW

The S3C72Q5 microcontroller can directly drive an up-to-12-characters (5 x 12 dots) LCD panel. Its LCD block has the following components:

- LCD controller/driver
- Display RAM (100H-1BBH) for storing display data (13H page)
- 60 segment output pins (SEG0-SEG59)
- 12 common output pins (COM0-COM11)
- LCD contrast control circuit by software (16 steps)

The frame frequency, LCD divide resistors, key strobe signal output key, and check signal output are determined by bit settings in the LCD mode register, LMOD.

The LCD duty and normal LCD display are determined by bit settings in the LCD control registers, LCON0 and LCON1.

When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even during main stop and idle modes.

LCD CIRCUIT DIAGRAM

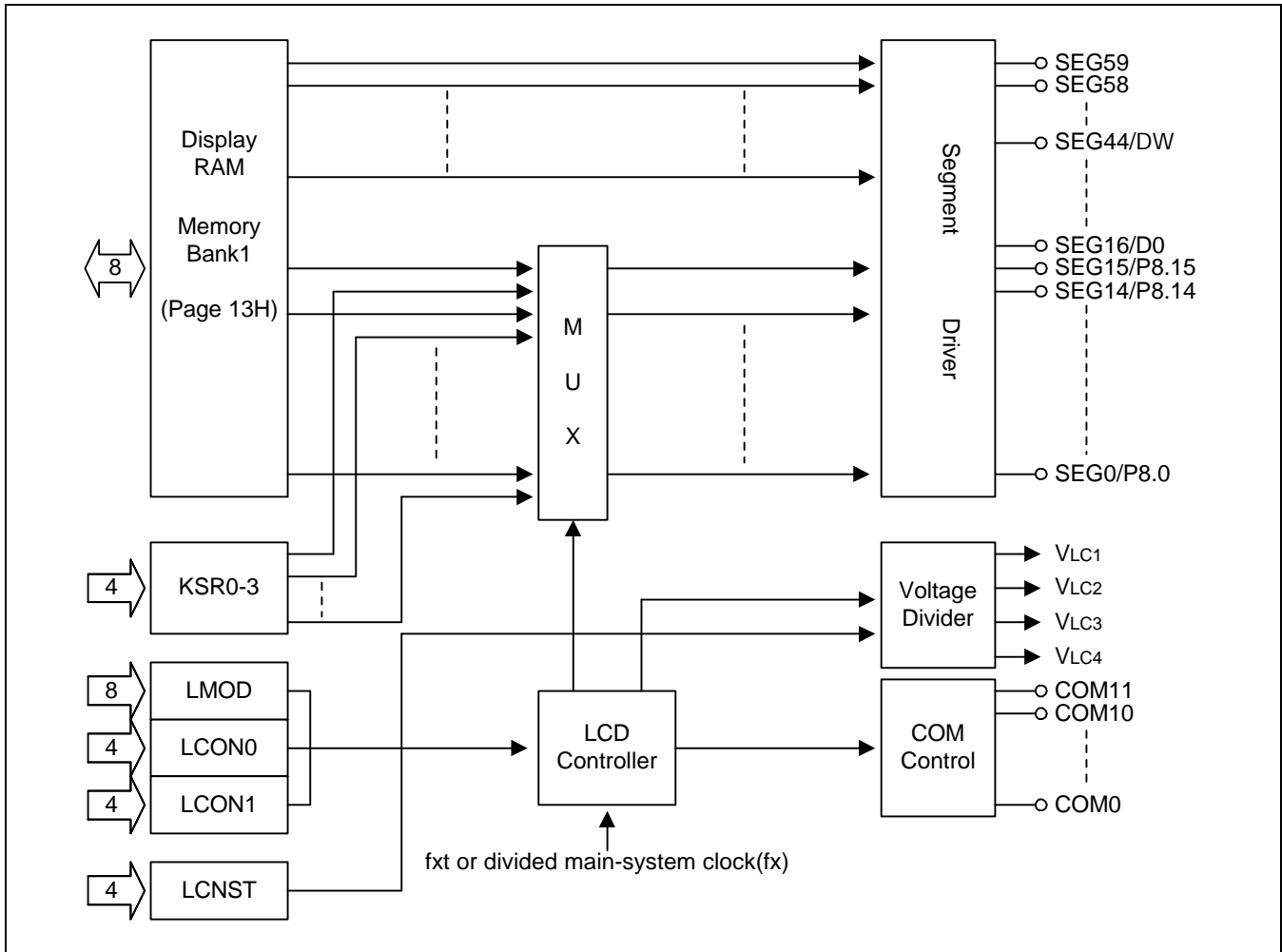


Figure 12-1. LCD Circuit Diagram

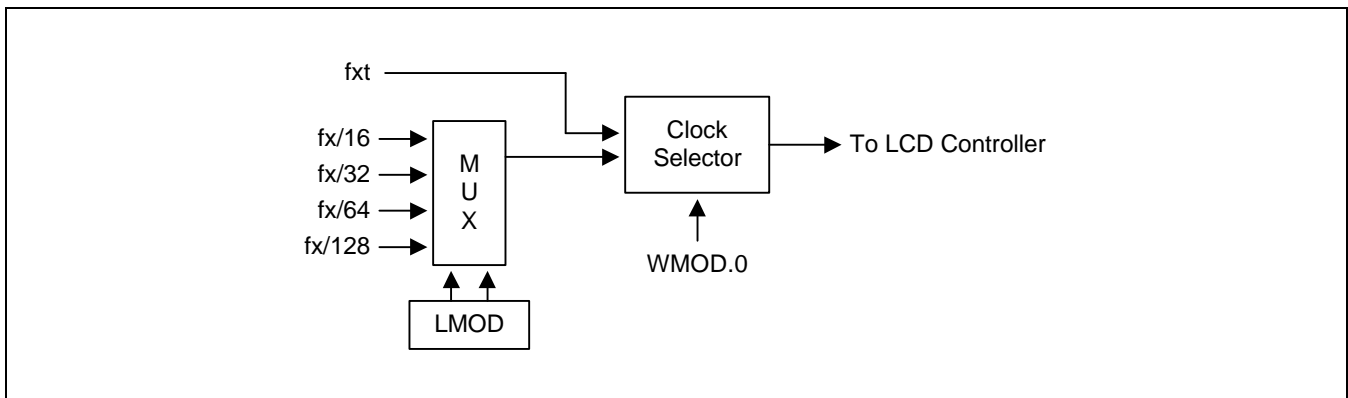


Figure 12-2. LCD Clock Circuit Diagram

LCD RAM ADDRESS AREA

RAM addresses 100H-1BBH of bank1 page 13H are used as LCD data memory. These locations can be addressed by 8-bit instructions only. However, the upper 3 bits of each address must be written to zero. When the bit value of a display segment is "1", the LCD display is turned on; when the bit value is "0", the display is turned off.

Display RAM data are sent out through segment pins SEG0-SEG59 using a direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

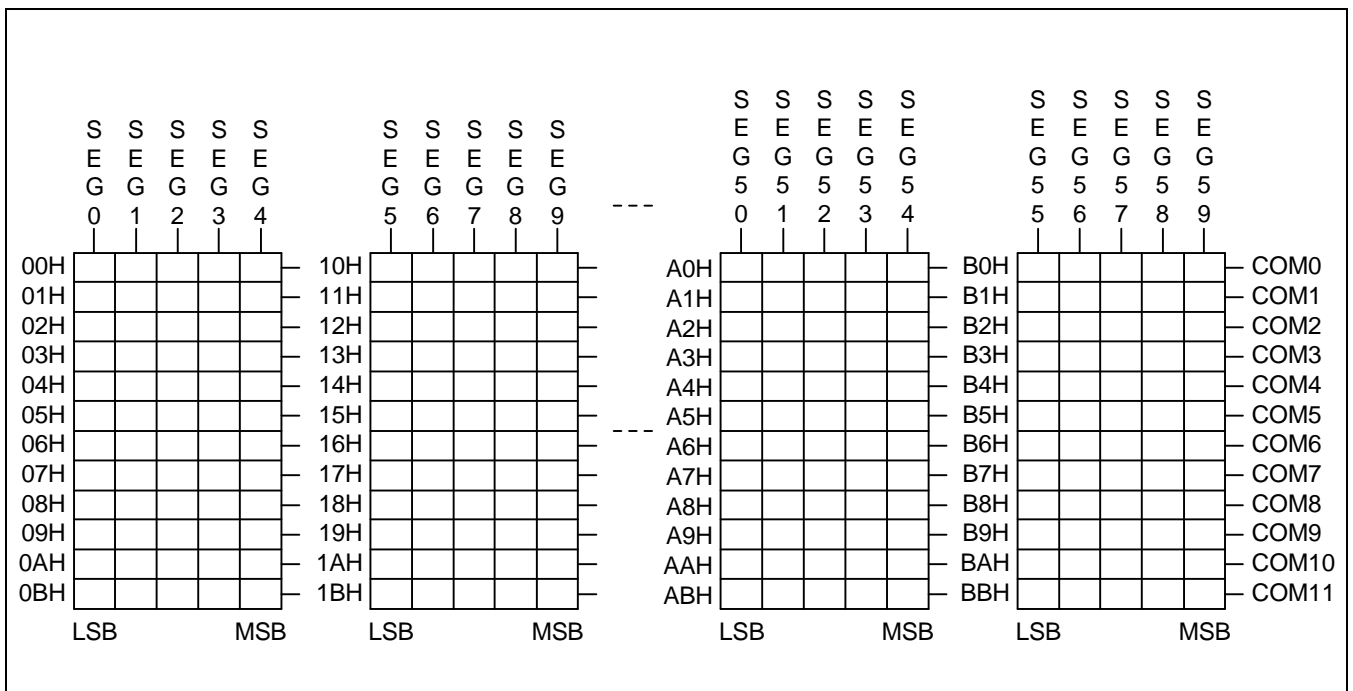


Figure 12-3. Display RAM Organization

LCD CONTRAST CONTROL REGISTER (LCNST)

The LCD contrast control register (LCNST) is used to control the LCD contrast up to 16 step contrast level. Following a RESET, all LCNST values are cleared to "0". This disable the LCD contrast control.

F8AH	LCNST.3	LCNST.2	LCNST.1	LCNST.0
F8BH	LCNST.7	0	0	0

Table 12-1. LCD Contrast Control Register (LCNST) Organization

LCD Contrast Control Enable/Disable Bit				
LCNST.7	Enable/Disable LCD Contrast Control			
0	Disable LCD contrast control			
1	Enable LCD contrast control			
Bits 6-4				
Bits 6-4	Always logic zero			
Segment/Port Output Selection Bits				
LCNST.3	LCNST.2	LCNST.1	LCNST.0	16 Step Contrast Level
0	0	0	0	1/16 step (The dimmest level)
0	0	0	1	2/16 step
0	0	1	0	3/16 step
0	0	1	1	4/16 step
0	1	0	0	4/16 step
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
1	1	1	1	16/16 step (The brightest level)

NOTE: $V_{LCD} = V_{DD} \times (1 - (16-n)/48)$, when $n = 0-15$ (At normal LCD dividing Resistors)

LCD OUTPUT CONTROL REGISTER 0 (LCON0)

The LCD output control register 0, LCON0 can be manipulated using 4-bit write instructions.

F8EH	LCON0.3	LCON0.2	"0"	"0"
------	---------	---------	-----	-----

LCON0 can select LCD duty.

Table 12-2. LCD Output Control Register (LCON0) Organization

LCON0.3	LCON0.2	Duty
0	0	1/9 duty (COM0-COM8 select)
0	1	1/10 duty (COM0-COM9 select)
1	0	1/11 duty (COM0-COM10 select)
1	1	1/12 duty (COM0-COM11 select)

NOTES:

1. COM has priority over normal port in P7.3/COM9-P7.1/COM11. This means these port are assigned to COM pins regardless of the value of PMG2, when duty is selected to 1/10, 1/11, or 1/12 at LCON0 register.
2. The port used COM must be set to output to prevent LCD display distortion.

LCD OUTPUT CONTROL REGISTER 1 (LCON1)

The LCD output control register 1, LCON1 can be manipulated using 4-bit write instructions.

F8FH	LCON1.3	LCON1.2	LCON1.1	LCON1.0
------	---------	---------	---------	---------

LCON1 control the following LCD functions.

- LCD display on/off LCON1.2
- Key check signal output with LCD display off (LCON1.3)
- Diming mode (LCON1.0)

Table 12-3. LCD Output Control Register (LCON1) Organization

LCON1.3	LCON1.2	LCON1.1	LCON1.0	Bias Selection for LCD Display
0	1	0	0	LCD display on
0	1	0	1	Dimming mode
1	0	0	1	Key check signal output with LCD display off

NOTES:

1. To turn off LCD display, you must set LCON1 to 9 not 0.
2. P8 can be used to normal output port, when LCD display off. The value of P8 is determined by KSR0-KSR3 regardless of LMOD.0. (refer to P12-17)

LCD MODE REGISTER (LMOD)

The LCD mode register LMOD can be manipulated using 8-bit write instructions.

F8DH	"0"	LMOD.6	LMOD.5	LMOD.4
F8CH	LMOD.3	LMOD.2	LMOD.1	LMOD.0

LMOD controls the following LCD functions:

- External interrupt INTP0 enable/disable selection bits (LMOD.6, LMOD.5, and LMOD.4)
- External interrupt INTP0 detection pins can be select (LMOD.6, LMOD.5, and LMOD.4)
- When main system clock is selected as watch timer clock by WMOD.0, watch timer clock selection bits (LMOD.3 and LMOD.2)
- LCD dividing resistors selection (LMOD.1)
- Key strobe signal disable/enable selection (LMOD.0)

Table 12-4. LCD Mode Control Register (LMOD) Organization

External Interrupt (INTP0) Pins Selection Bits ⁽¹⁾			
LMOD.6	LMOD.5	LMOD.4	External Interrupt (INTP0) Pins Selection Bits ⁽¹⁾
0	0	0	Interrupt request at K0 triggered by falling edge
0	0	1	Interrupt request at K0-K1 triggered by falling edge
0	1	0	Interrupt request at K0-K2 triggered by falling edge
0	1	1	Interrupt request at K0-K3 triggered by falling edge
1	0	0	Interrupt request at K0-K4 triggered by falling edge
1	0	1	Interrupt request at K0-K5 triggered by falling edge
1	1	0	Interrupt request at K0-K6 triggered by falling edge
1	1	1	Interrupt request flag (IRQP0) cannot be set to logic one
Watch Timer Clock Selection Bits ⁽²⁾			
LMOD.3	LMOD.2	When main system clock is selected as watch timer clock by WMOD.o	
0	0	fx/128	
0	1	fx/64	
1	0	fx/32	
1	1	fx/16	
LCD Dividing Resistor Selection Bits			
LMOD.1	LCD Dividing Resistor		
0	Normal LCD dividing resistors		
1	Diminish LCD dividing resistors to strength LCD drive		
Key Strobe Signal Output Control Bits (SEG0/P8.0-SEG15/P8.15)			
LMOD.0	Key Strobe Signal Output Control (SEG0/P8.0-SEG15/P8.15)		
0	Enable key strobe signal output ⁽³⁾		
1	Disable key strobe signal output ⁽⁴⁾		

NOTES:

1. The pins which are not selected as external interrupt (K0-K6) can be used to normal I/O. To use external interrupts, corresponding pins must be set to input mode.
2. LCD clock can be selected only when main clock(fx) is used as clock source of watch timer. When sub clock(fxt) is used as clock source of watch timer, LCD clock is always fw/48 (1/9 duty), fw/44 (1/10 duty), fw/40 (1/11duty), or fw/36 (1/12 duty).
3. In this case, pull-up resistors of port0,1 are disabled if the value of PUR0 is "0", when the value of PUR0 is "1", the pull-up resistors of selected pins as interrupt input are enabled or disabled by key strobe signal, and that of non-selected pins are disabled.
4. In this case, pull-up resistors of port 0,1 are disabled or enabled by the value of PUR0 flag.

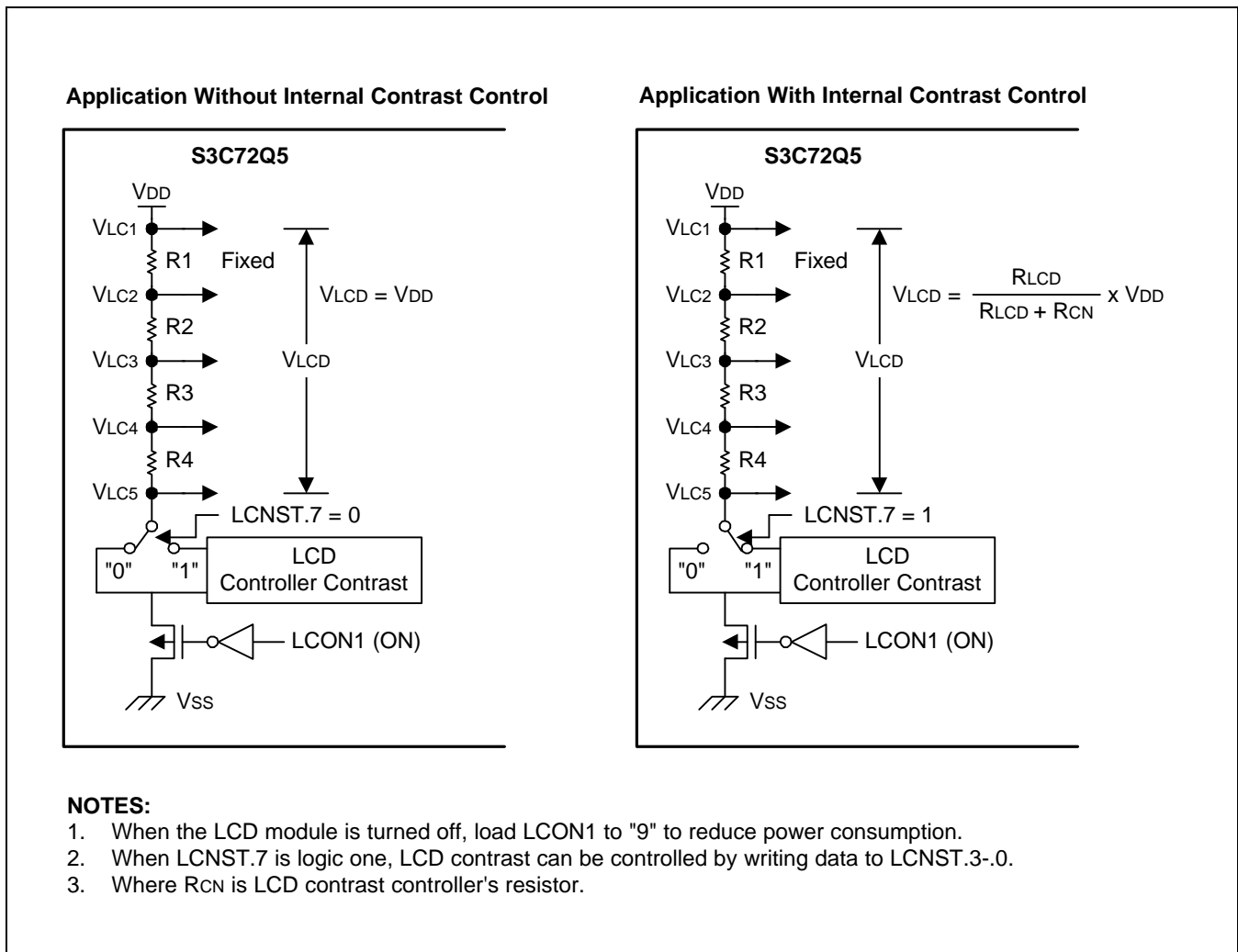


Figure 12-4. LCD Voltage Dividing Resistors Connection

When LCON1 is 4 or 5, LMOD.0 are set to "0" and pull-up enable by PUMOD0, RE and LE signal are generated as below:

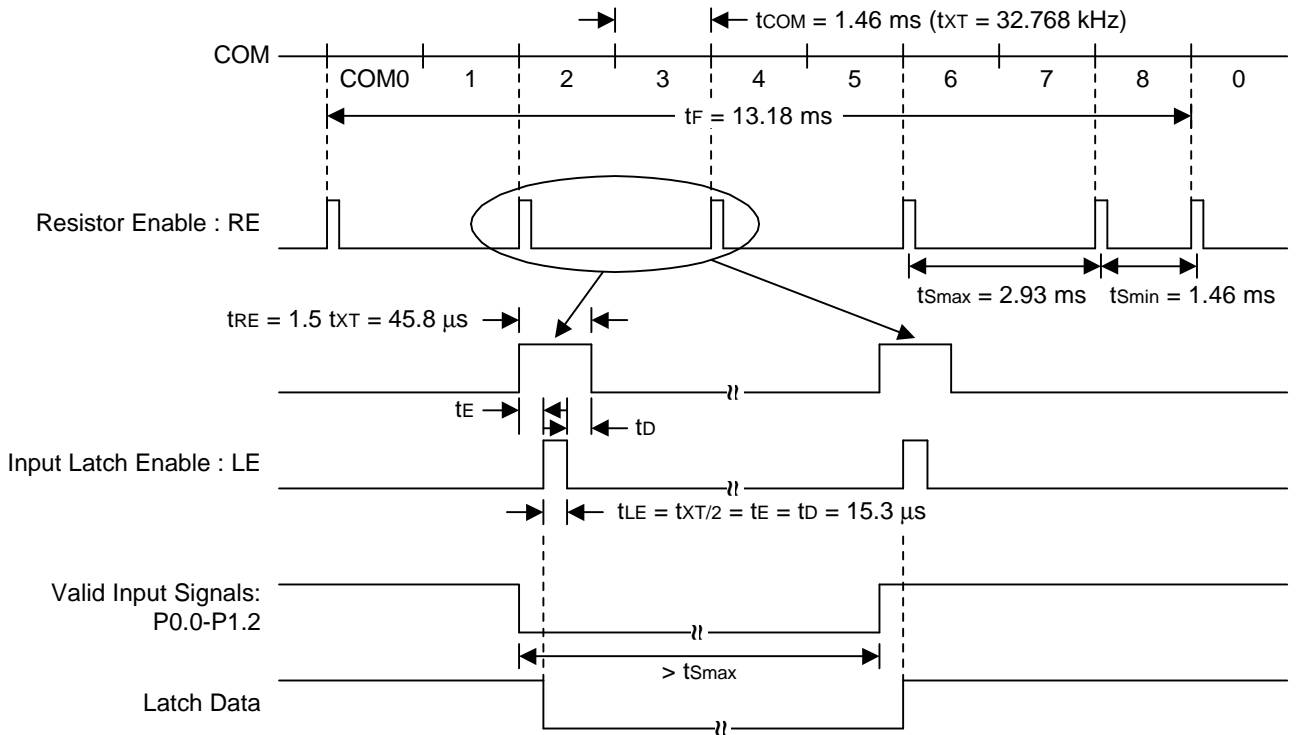


Figure 12-5. RE, LE and Inputs Signal Waveform (1/9 Duty)

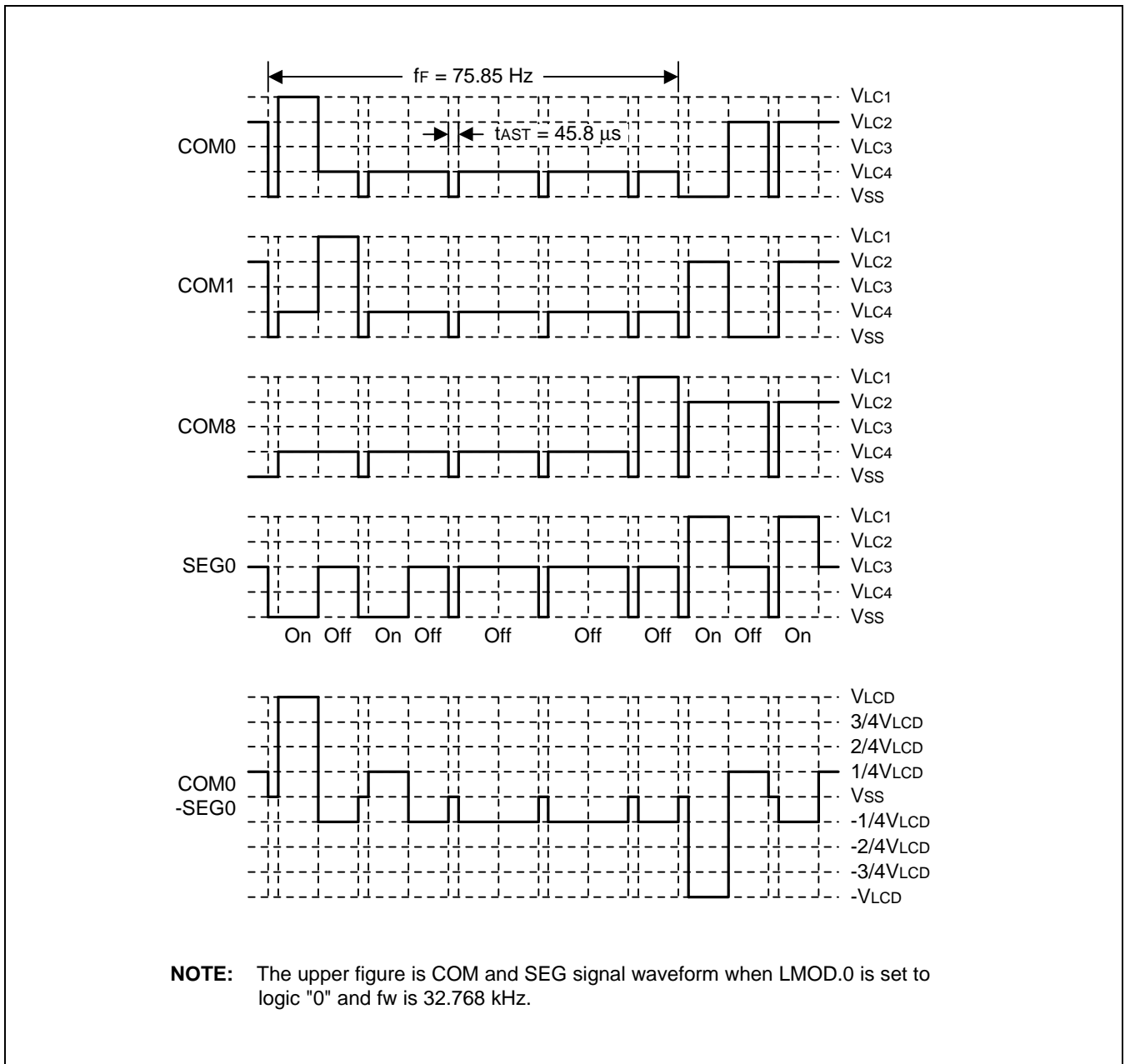


Figure 12-6. LCD Signal Waveform for 1/9 Duty and 1/4 Bias

When LCON1 is 4 or 5, LMOD.0 are set to "0" and pull-up enable by PUMOD0, RE and LE signal are generated as below:

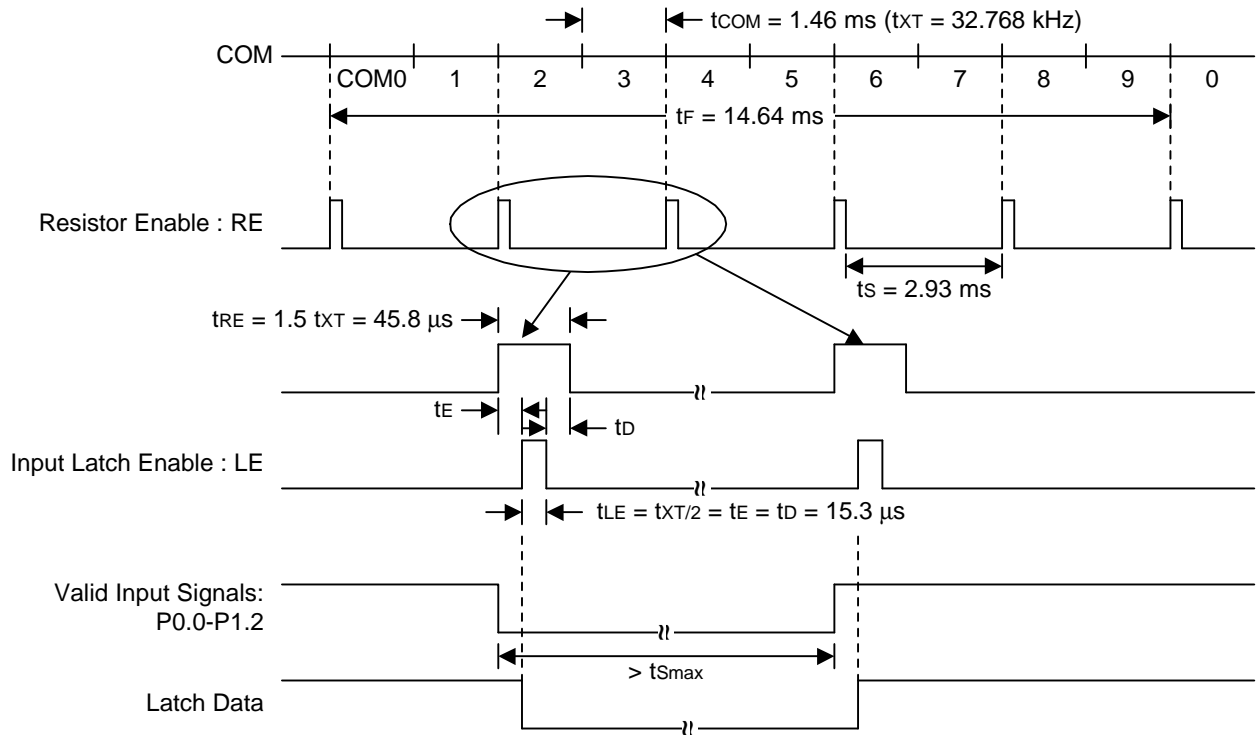


Figure 12-7. RE, LE and Inputs Signal Waveform (1/10 Duty)

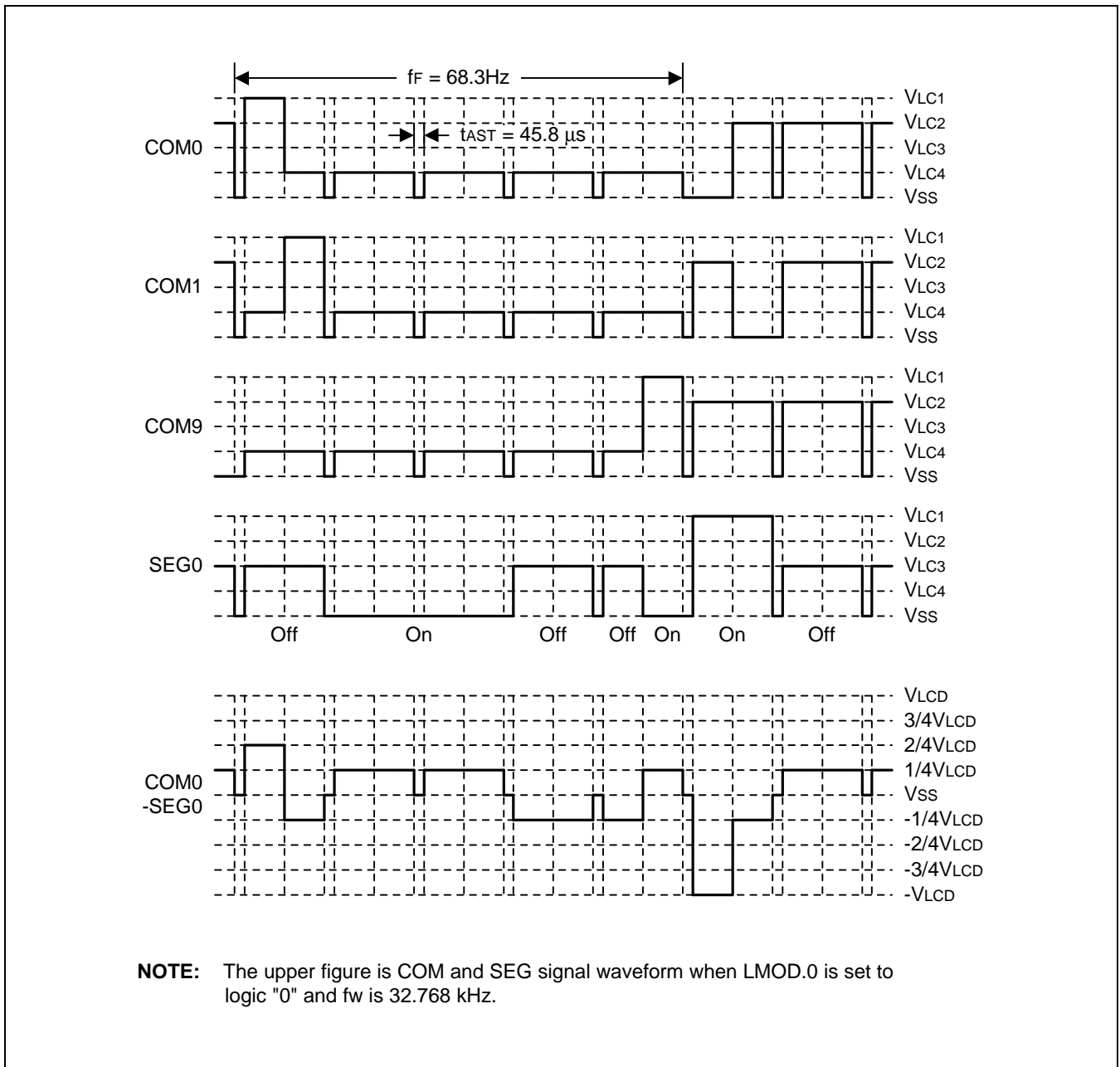


Figure 12-8. LCD Signal Waveform for 1/10 Duty and 1/4 Bias

When LCON1 is 4 or 5, LMOD.0 are set to "0" and pull-up enable by PUMOD0, RE and LE signal are generated as below:

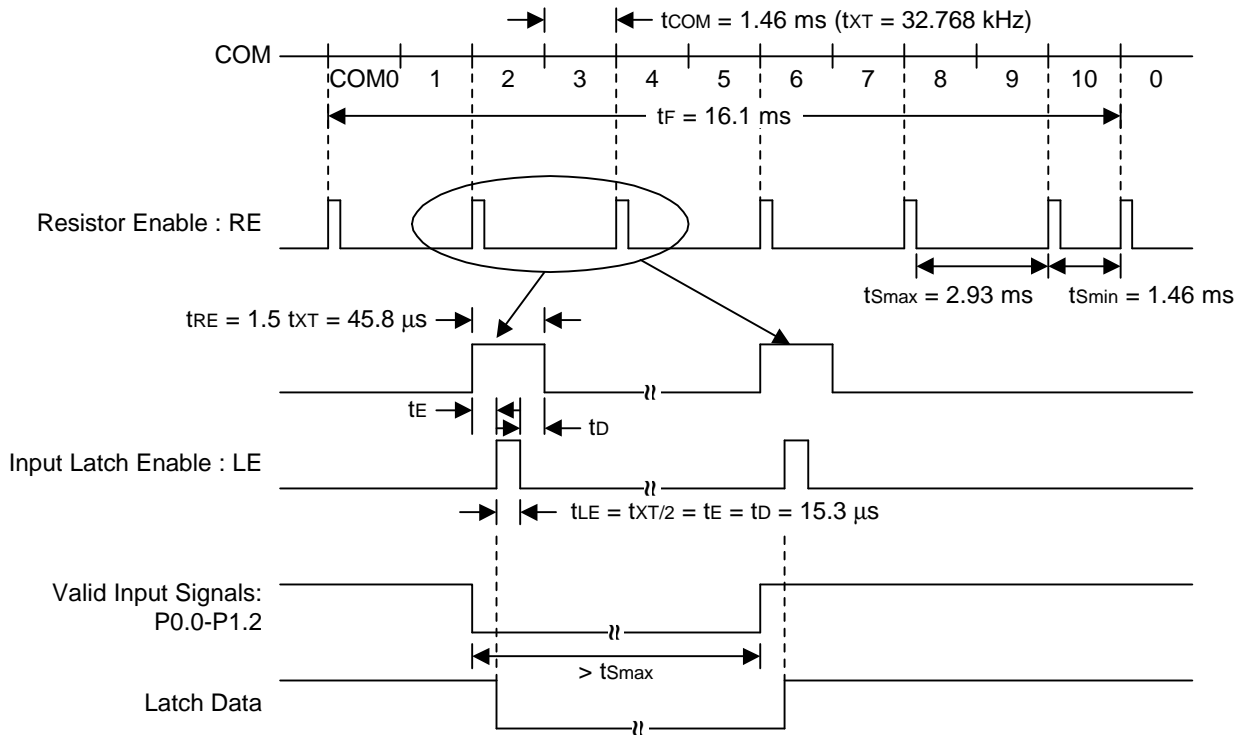
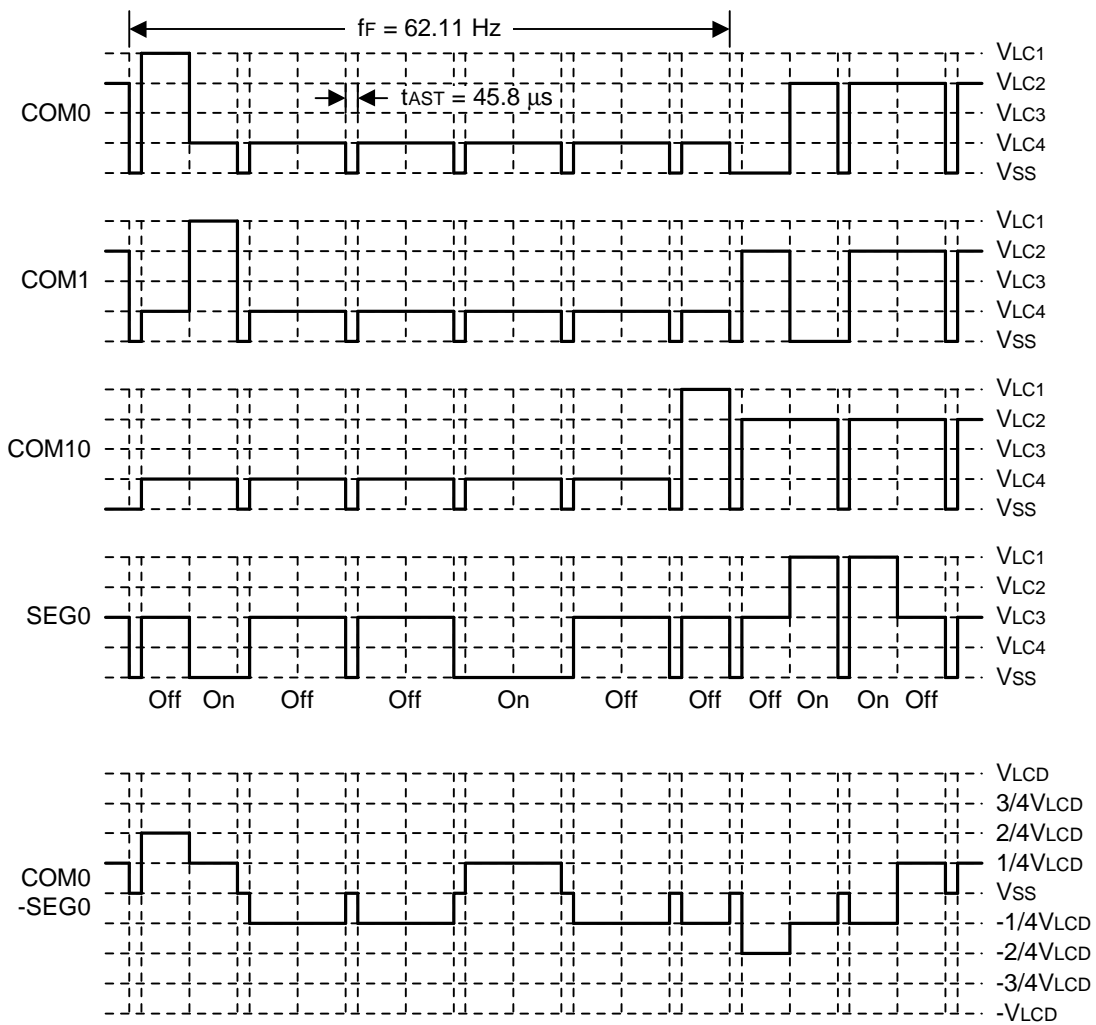


Figure 12-9. RE, LE and Inputs Signal Waveform (1/11 Duty)



NOTE: The upper figure is COM and SEG signal waveform when LMOD.0 is set to logic "0" and fw is 32.768 kHz.

Figure 12-10. LCD Signal Waveform for 1/11 Duty and 1/4 Bias

When LCON1 is 4 or 5, LMOD.0 are set to "0" and pull-up enable by PUMOD0, RE and LE signal are generated as below:

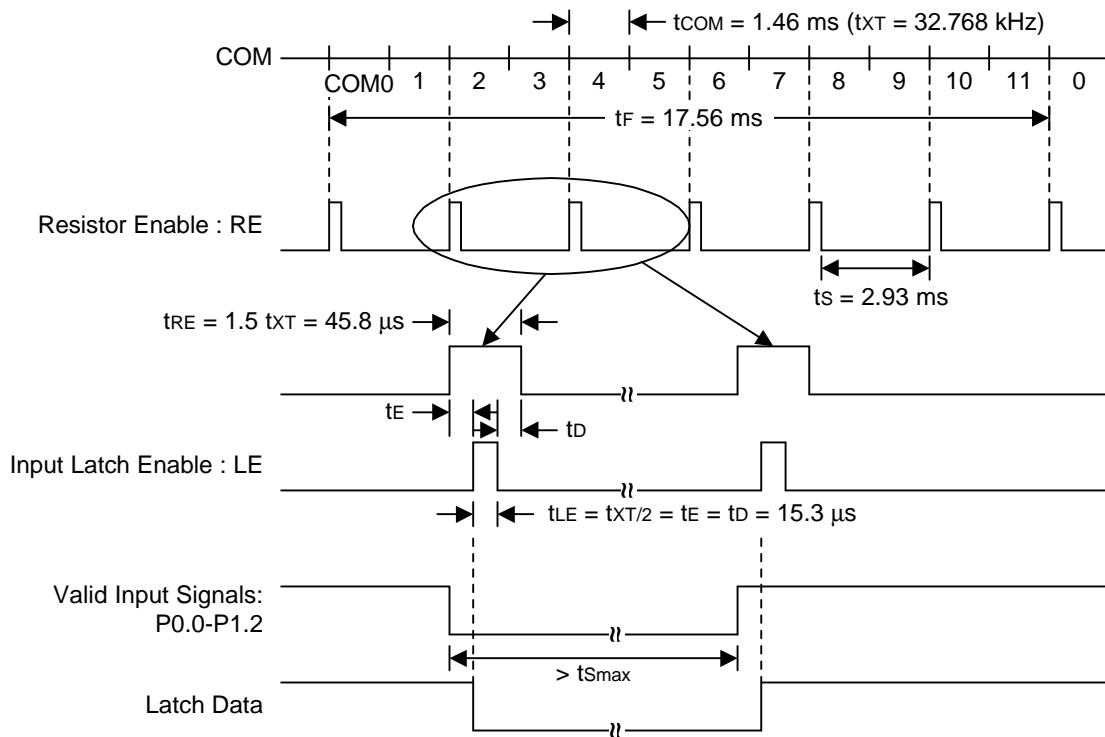


Figure 12-11. RE, LE and Inputs Signal Waveform (1/12 Duty)

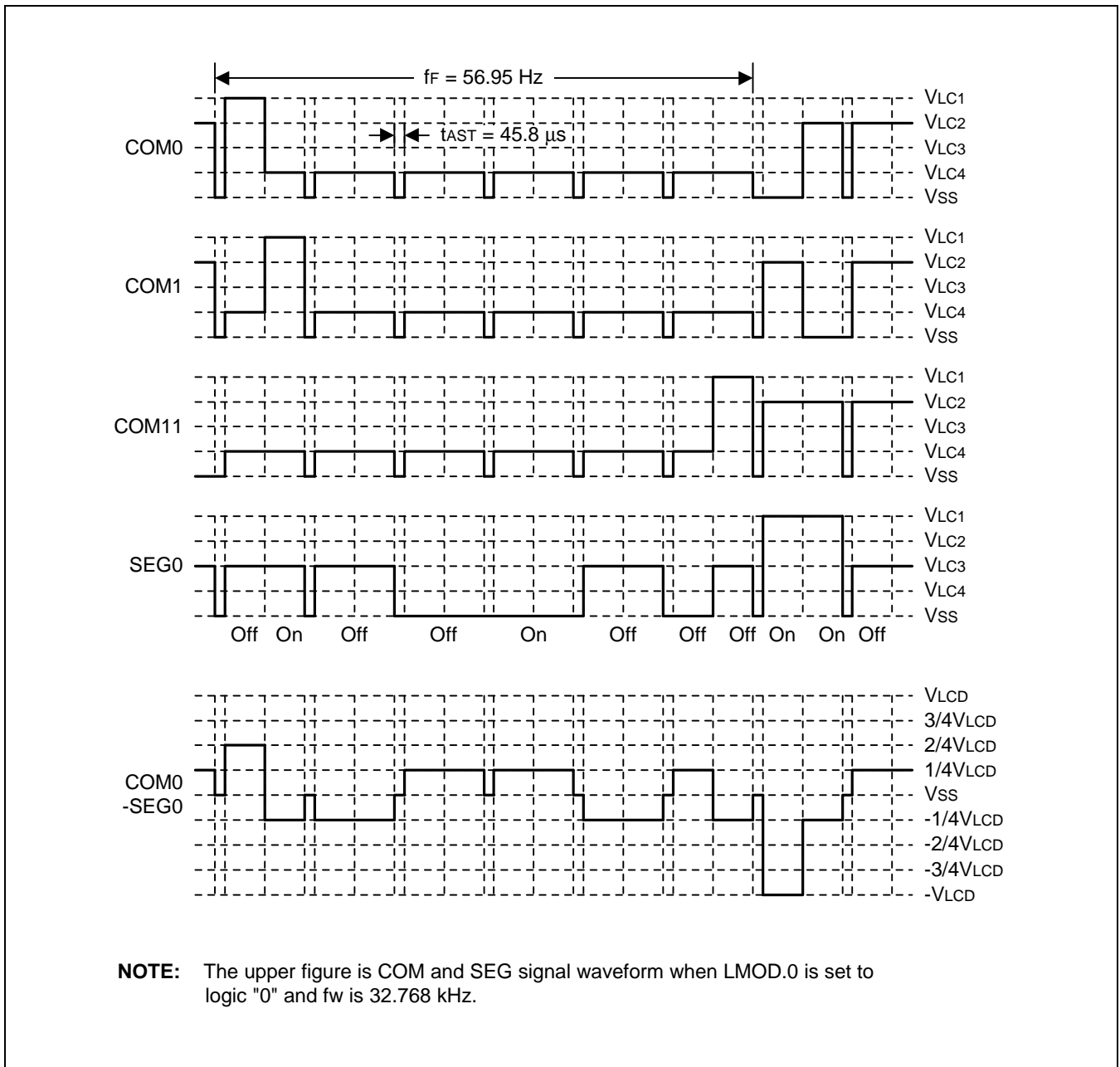


Figure 12-12. LCD Signal Waveform for 1/12 Duty and 1/4 Bias

KEY SCAN REGISTER (KSR)

The 16 output pins (P8.0-P8.15) of 60 segments can be used for key check signal output. KSR0–KSR3 are mapped to the RAM address FA2H-FA5H, and the reset value is "0". KSR is the write-only register that can be manipulated by 4-bits RAM write instruction only.

Table 12-5. KSR Organization

KSR0	KSR0.3	KSR0.2	KSR0.1	KSR0.0	FA2H
KSR1	KSR1.3	KSR1.2	KSR1.1	KSR1.0	FA3H
KSR2	KSR2.3	KSR2.2	KSR2.1	KSR2.0	FA4H
KSR3	KSR3.3	KSR3.2	KSR3.1	KSR3.0	FA5H

When LCON1 is 9, the values of KSR0-KSR3 are output to segment pins for key check regardless of LMOD.0. At this time, only one of 16 bits (KSR0.0-KSR3.3) must be set to logic "1", and the contents of KSR must be changed 16 times one by one for 16 key check by software. When a bit value of KSR is "1", the corresponding segment pin becomes the low level. Figure 12-14 shows its segment pin output.

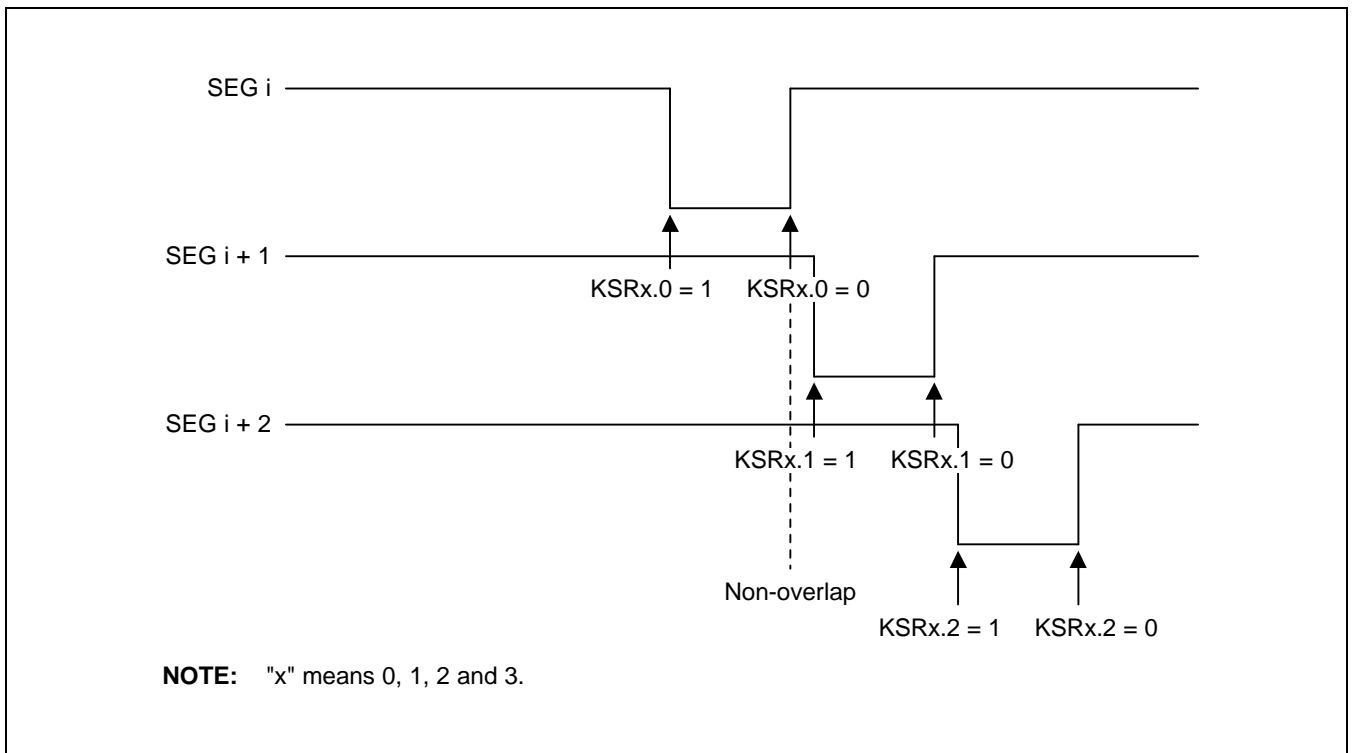


Figure 12-13. Segment Pin Output Signal When LCON1.3 = 1

NOTES

13

EXTERNAL MEMORY INTERFACE

OVERVIEW

The S3C72Q5 microcontroller can directly interface the external memory up to 6 x 4 M-bit. It external memory interface block has the following components.

- 8-bit external memory control register (EMCON)
- External memory address register 0, 1, 2 (EMAR0-EMAR2)
- 8-bit external memory data register 0 (EMDR0)
- External memory interface clock selector
- Six external memory selection pins (DM0-DM5)
- Eight data and nineteen address pins (D0-D7, A0-A18)

It should be taken care to the LCD contrast due to an external memory interface, since all external memory interface lines except DM0-DM5 are shared with segment driver pins of LCD driver/controller.

EXTERNAL MEMORY CONTROL REGISTER (EMCON)

The external memory control register (EMCON) is used to read data from or write data in a external memory, to select external memory interface clock frequency, to increase automatically the address (a value of EMAR2-EMAR0) or not, and to select data memory pin (one of DM0-DM5). The EMCON can be manipulated using 8-bit write.

FD2H	EMCON.3	EMCON.2	EMCON.1	EMCON.0
FD3H	EMCON.7	EMCON.6	EMCON.5	EMCON.4

The memory access clock frequency, f_M , determines the read and write time for external memory. The f_M should be selected appropriately because a memory has read and write time specified and the external memory interface lines except DM0-DM5 are shared with segment driver pins of LCD driver/controller.

Table 13-1. External Memory Control Register (EMCON) Organization

Memory Read/Write Control Bit			
EMCON.7	0	Memory read signal output	
	1	Memory write signal output	
Memory Access Clock Selection Bits			
EMCON.6	EMCON.5	Memory Access Clock Frequency (f_M)	
0	0	$f_{xx}/8$	
0	1	$f_{xx}/4$	
1	0	$f_{xx}/2$	
1	1	$f_{xx}/1$	
Address Increment Control Bit			
EMCON.4	0	The address(a value of EMAR2 - EMAR0) is not increased automatically after memory access.	
	1	The address(a value of EMAR2 - EMAR0) is increased automatically after memory access.	
Memory Selection Bits			
EMCON.3	EMCON.2	EMCON.1	External data memory selection
0	0	0	Data memory 0(DM0 active)
0	0	1	Data memory 1(DM1 active)
0	1	0	Data memory 2(DM2 active)
0	1	1	Data memory 3(DM3 active)
1	0	0	Data memory 4(DM4 active)
1	0	1	Data memory 5(DM5 active)
Memory Access Start Bit (This bit is cleared automatically when memory access is finished)			
EMCON.0	0	Not busy (read)	
	1	Start a memory access (write) busy (read)	

NOTES:

1. When it reads data from a external memory, the data are written to the register EMDR0.
2. When it writes data to a external memory, the data to the register EMDR0 are written to a external memory.
3. The external memory selection pins of P6.0/DM0 - P7.1/DM5 should be set to push-pull output and the latches should be set to logic "1".

HOW TO ACCESS THE EXTERNAL MEMORY

The pin which are selected for external memory interface of DM0-DM5 should be set to push-pull output and the latches should be set to logic "1". The procedure for external memory interface may be summarized as follows.

1. To read data form external memory

- Load the address of external memory to EMAR2-EMAR0 in bank 15.
- Clear EMCON.7 to logic 0, load appropriate values to EMCON.6-.1
- Wait for memory access set-up time
- Set EMCON.0 to logic 1.
- Check EMCON.0 until for not busy ("0")
- Read a value of EMDR0 in bank 15 if EMCON.0 is "0".

2. To write data to external memory

- Load the address of external memory to EMAR2-EMAR0 in bank 15.
- Load data to EMDR0 in bank 15
- Set EMCON.7 to logic 1, load appropriate values to EMCON.6-.1
- Wait for memory access set-up time
- Set EMCON.0 to logic 1.
- Check EMCON.0 until not busy (0) before writing other data to external memory.

PROGRAMMING TIP — External Memory Interface

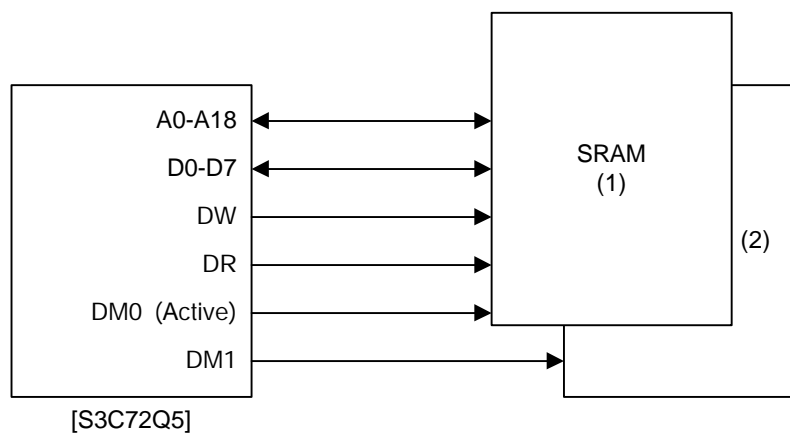
The external memory selection pins of P6.0/DM0-P7.1/DM5 should be set to push-pull output and the latches should be set to logic "1".

- To read data form external memory when DM0 and DM1 are used to control CS pins external memories respectively.

```

        BITS      EMB          ; Initial part
        SMB       15
        LD        EA,#03H
        LD        PMG2,EA      ; DM0 and DM1 ← output
        LD        EA,#03H
        LD        P6,EA        ; DM0(P6.0) and DM1 (P6.1) latches ← "1"
        LD        EA,#00010000B ; Read mode, increase address automatically,
                                ; select DM0, fm = fxx/8
        LD        EMCON,EA
        :           ; need set-up time
        :
        :
        BITS      EMB
        SMB       15
        LD        EA,#00H
        LD        EMAR0,EA
        LD        EMAR1,EA
        LD        A,#4H
        LD        EMAR2,A      ; External memory address ← 40000H
        LD        EA,#00010001B
        LD        EMCON,EA     ; Start a memory reading
;
; DELAY          LD        EA,EMCON
;                AND      A,#0001B
;                DECS    A
;                JPS     DELAY
NEXT           LD        EA,EMDR0 ; Data
;                SMB     0
;                LD      ADATR_B,EA

```



PROGRAMMING TIP — External Memory Interface (Continued)

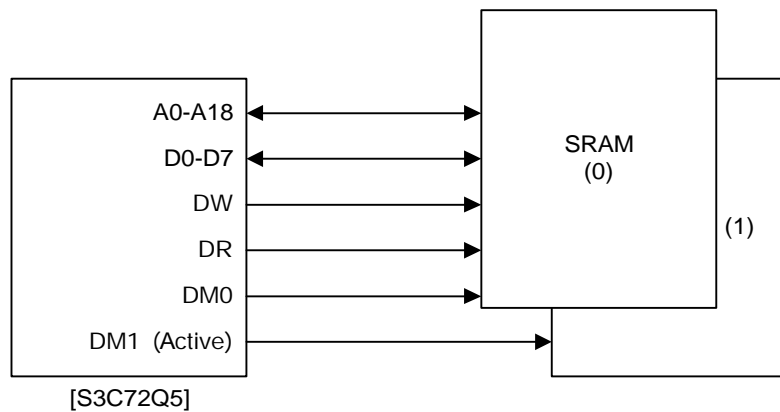
- 2. To write data to external memory when DM0 and DM1 are used to control CS pins of external memory, respectively.

```

BITS      EMB          ; Initial part
SMB       15
LD        EA,#03H
LD        PMG2,EA      ; DM0 and DM1 ← output
LD        EA,#03H
LD        P6,EA        ; DM0(P6.0) and DM1 (P6.1) latches ← "1"
LD        EA,#10010010B ; Write mode, increase address automatically,
                        ; select DM1, fM = fxx/8

LD        EMCON,EA
:
:
:
BITS      EMB
SMB       15
LD        EA,#00H
LD        EMAR0,EA
LD        EMAR1,EA
LD        A,#4H
LD        EMAR2,A      ; External memory address 40000H
LD        EA,#38H
LD        EMDR0,EA     ; Data
LD        EA,#10010011B ; Write mode, increase address automatically,
                        ; select DM1, fM = fxx/8

;
DELAY    LD        EA,EMCON
        AND       A,#0001B
        DECS     A
        JPS      DELAY
NEXT     SMB       0
        LD        EA,ADATR_B
        SMB       15
        LD        EMDR0,EA      ; Data
    
```



EXTERNAL MEMORY WRITE CYCLE TIMING DIAGRAM

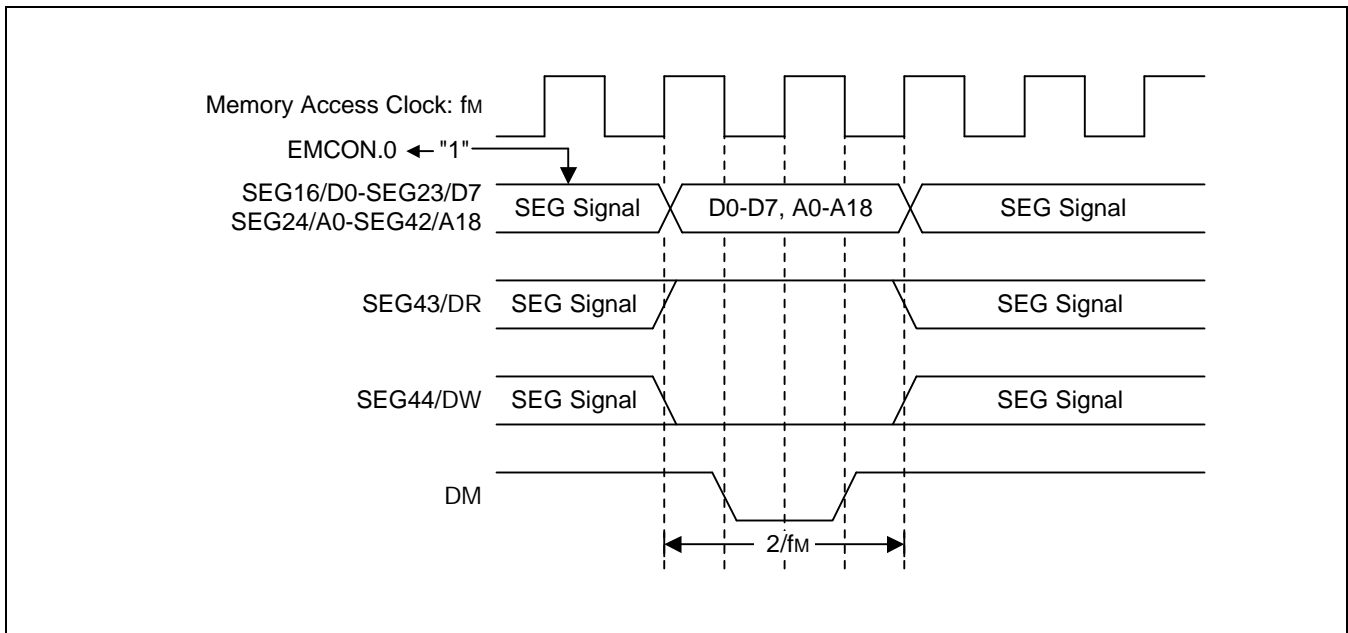


Figure 13-1. External Memory Write Cycle Timing Diagram

EXTERNAL MEMORY READ CYCLE TIMING DIAGRAM

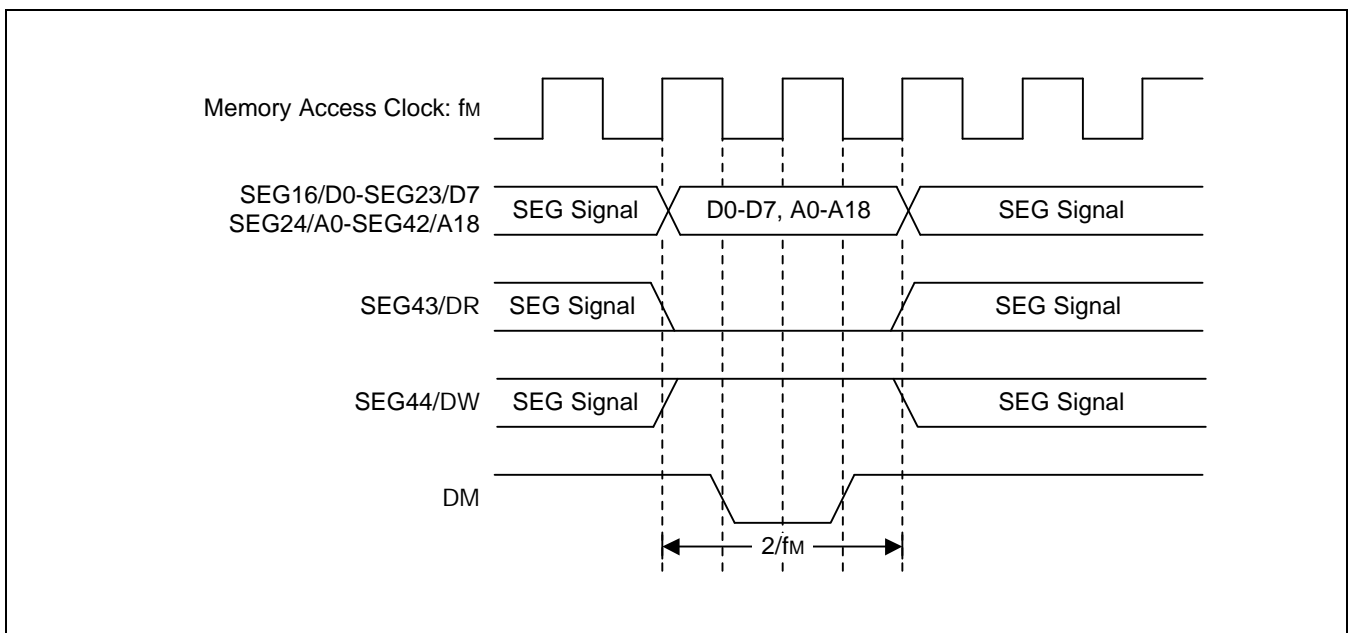


Figure 13-2. External Memory Read Cycle Timing Diagram

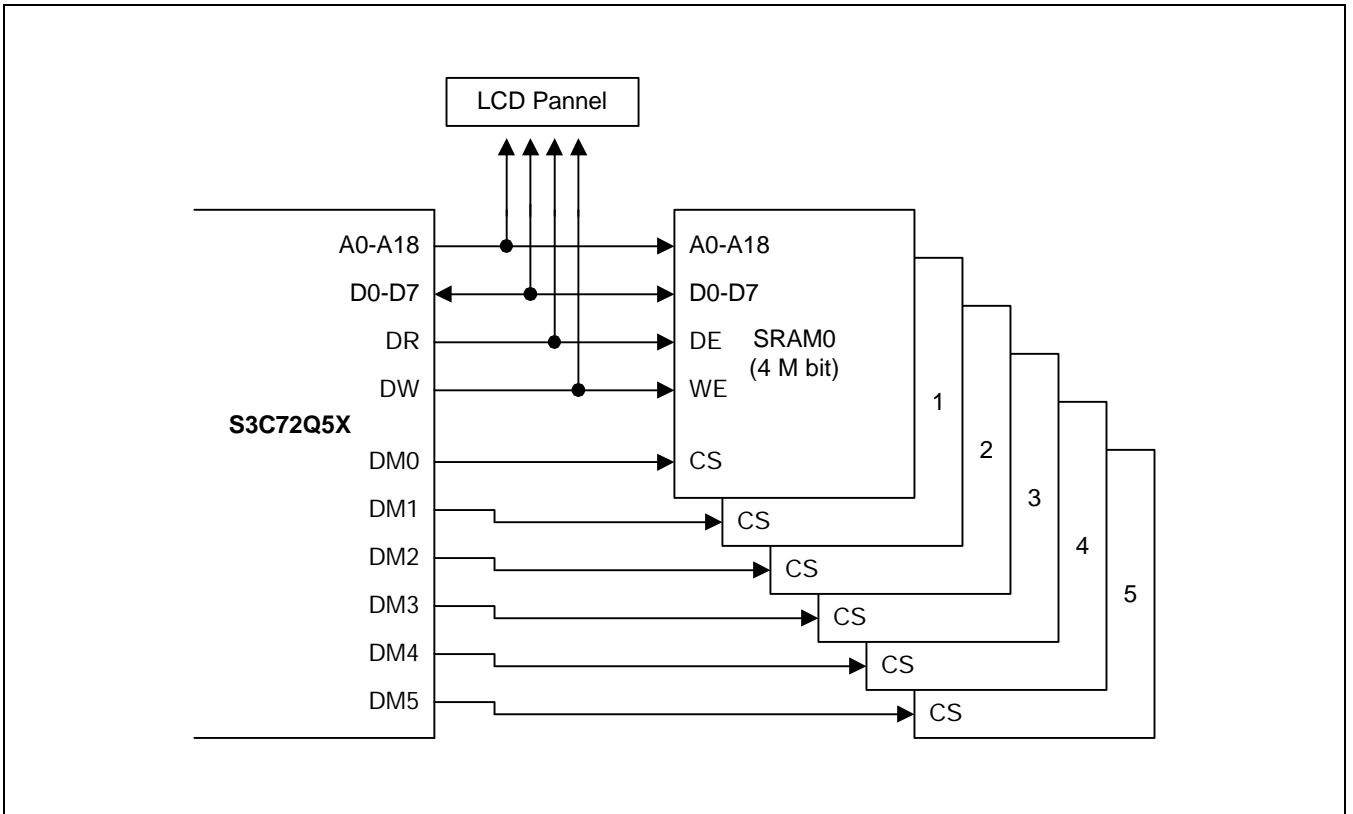


Figure 13-3. External Interface Function Diagram (S3C72Q5, SRAM, EPROM, EEPROM)

NOTES

14 ELECTRICAL DATA

OVERVIEW

In this section, information on S3C72Q5 electrical characteristics is presented as tables and graphics. The information is arranged in the following order:

Standard Electrical Characteristics

- Absolute maximum ratings
- D.C electrical characteristics
- Main-system clock oscillator characteristics
- Sub-system clock oscillator characteristics
- I/O capacitance
- A.C electrical characteristics
- Operating voltage range

Miscellaneous Timing Waveforms

- A.C timing measurement point
- Clock timing measurement at X_{IN}
- Clock timing measurement at XT_{IN}
- TCL0 timing
- Input timing for RESET
- Input timing for external interrupts

Stop Mode Characteristics and Timing Waveforms

- RAM data retention supply voltage in stop mode
- Stop mode release timing when initiated by RESET
- Stop mode release timing when initiated by an interrupt request

Table 14-1. Absolute Maximum Ratings

 $(T_A = 25\text{ }^\circ\text{C})$

Parameter	Symbol	Conditions	Rating	Units
Supply Voltage	V_{DD}	–	- 0.3 to + 6.5	V
Input Voltage	V_I	Ports 0, 1, 4 - 7	- 0.3 to $V_{DD} + 0.3$	V
Output Voltage	V_O	–	- 0.3 to $V_{DD} + 0.3$	V
Output Current High	I_{OH}	One I/O pin active	- 15	mA
		All I/O pins active	- 30	
Output Current Low	I_{OL}	One I/O pin active	+ 30 (Peak value)	mA
			+ 15(note)	
		Total for ports 0, 1, 4 - 7, 8	+ 100 (Peak value)	
			+ 60(note)	
Operating Temperature	T_A	–	- 40 to + 85	$^\circ\text{C}$
Storage Temperature	T_{stg}	–	- 65 to + 150	$^\circ\text{C}$

NOTE: The values for Output Current Low (I_{OL}) are calculated as Peak Value $\times \sqrt{\text{Duty}}$.

Table 14-2. D.C Characteristics

 $(T_A = -40\text{ }^\circ\text{C}$ to $+85\text{ }^\circ\text{C}$, $V_{DD} = 1.8\text{ V}$ to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Voltage	V_{IH1}	Ports 0, 1, 4 - 7 and D0 - D7	$0.8V_{DD}$	–	V_{DD}	V
	V_{IH2}	RESET	$0.7V_{DD}$		V_{DD}	
	V_{IH3}	X_{IN} , X_{OUT} , and XT_{IN}	$V_{DD} - 0.1$		V_{DD}	
Input Low Voltage	V_{IL1}	Ports 0, 1, 4 - 7 and D0 - D7	–		$0.2V_{DD}$	
	V_{IL2}	RESET			$0.3V_{DD}$	
	V_{IL3}	X_{IN} , X_{OUT} , and XT_{IN}			0.1	
Output High Voltage	V_{OH}	$V_{DD} = 4.5\text{ V}$ to 5.5 V $I_{OH} = -1\text{ mA}$ Ports 0,1,4-7,Memory access pins	$V_{DD} - 1.0$		–	
Output Low Voltage	V_{OL}	$V_{DD} = 4.5\text{ V}$ to 5.5 V $I_{OL} = 15\text{ mA}$ Ports 0,1,4-7,Memory access pins	–		2.0	
		$V_{DD} = 1.8\text{ V}$ to 5.5 V $I_{OL} = 1.6\text{ mA}$			0.4	

Table 14-2. D.C Characteristics (continued)

(T_A = - 40 °C to + 85C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Leakage Current	I _{LIH1}	V _I = V _{DD} All input pins except those specified below for I _{LIH2}	-	-	3	μA
	I _{LIH2}	V _I = V _{DD} X _{IN} , X _{OUT} , XT _{IN}			20	
Input Low Leakage Current	I _{LIL1}	V _I = 0 V All input pins except RESET , X _{IN} , X _{OUT} , and XT _{IN}			- 3	
	I _{LIL2}	V _I = 0 V X _{IN} , X _{OUT} , XT _{IN}			- 20	
Output High Leakage Current	I _{LOH}	V _O = V _{DD} All output pins			3	
Output Low Leakage Current	I _{LOL}	V _O = 0 V All output pins			- 3	
Pull-Up Resistor	R _{L1}	V _I = 0 V; V _{DD} = 5V Ports 0, 1, 4 - 7	25	50	75	kΩ
		V _{DD} = 3V	50	100	150	
	R _{L2}	V _I = 0 V; V _{DD} = 5V; RESET	100	200	300	
		V _{DD} = 3V	250	500	750	
LCD Voltage Dividing Resistor	R _{LCD1}	T _A = + 25 °C When LMOD.1 = "0"	46	66	86	
	R _{LCD2}	T _A = + 25 °C When LMOD.1 = "1"	23	33	43	
V _{LC1-COMi} Voltage Drop (i = 0-11)	V _{DC}	- 15 μA per common pin	-	-	120	mV
V _{LC1-SEGx} Voltage Drop (x = 0-59)	V _{DS}	- 15 μA per common pin	-	-	120	
Middle Output Voltage (1)	V _{LC2}	V _{DD} = 2.4V to 5.5V, 1/4 bias LCD clock = 0Hz	0.75V _{DD} - 0.2	0.75V _{DD}	0.75V _{DD} + 0.2	V
	V _{LC3}		0.5V _{DD} - 0.2	0.5V _{DD}	0.5V _{DD} + 0.2	
	V _{LC4}		0.25V _{DD} - 0.2	0.25V _{DD}	0.25V _{DD} + 0.2	

NOTES:

1. It is middle output voltage when LCD controller/driver is 1/12 duty and 1/4 bias.
2. Low leakage current is absolute value.

Table 14-2. D.C Characteristics (continued)

(T_A = -40 °C to +85°C, V_{DD} = 1.8 V to 5.5 V)

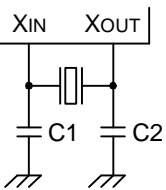
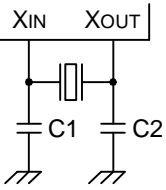
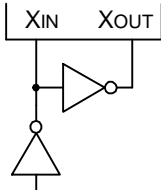
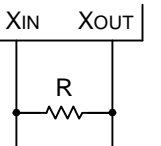
Parameter	Symbol	Conditions		Min	Typ	Max	Units		
Supply Current (1)	I _{DD1} (2)	V _{DD} = 5 V ± 10%	6 MHz	-	4.5	8.0	mA		
		Crystal oscillator C1 = C2 = 22pF	4.19 MHz		3.2	5.5			
	I _{DD2} (2)	V _{DD} = 3 V ± 10%	6 MHz		2.0	4.0			
			4.19 MHz		1.5	3.0			
	I _{DD2} (2)	Idle mode V _{DD} = 5 V ± 10%	6 MHz		1.3	2.5			
		Crystal oscillator C1 = C2 = 22pF	4.19 MHz		1.0	1.8			
	I _{DD2} (2)	V _{DD} = 3 V ± 10%	6 MHz		0.5	1.5			
			4.19 MHz		0.4	1.0			
	I _{DD3} (3)	V _{DD} = 3 V ± 10%	32 kHz crystal oscillator		20	35		μA	
	I _{DD4} (3)	Idle mode; V _{DD} = 3 V ± 10%	32 kHz crystal oscillator		5.0	15			
I _{DD5}	Stop mode; V _{DD} = 5 V ± 10%	SCMOD = 0000B XT _{IN} = 0V	2.5	5					
	Stop mode; V _{DD} = 3 V ± 10%		0.5	3					
	V _{DD} = 5 V ± 10%	SCMOD = 0100B	0.2	3					
	V _{DD} = 3 V ± 10%		0.1	2					

NOTES:

1. Currents in the following circuits are not included; on-chip pull-up resistors, internal LCD voltage dividing resistors, output port drive currents.
2. Data includes power consumption for subsystem clock oscillation.
3. When the system clock control register, SCMOD, is set to 1001B, main system clock oscillation stops and the subsystem clock is used.
4. Every values in this table is measured when the power control register (PCON) is set to "0011B".

Table 14-3. Main System Clock Oscillator Characteristics

(T_A = -40 °C + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency(f _x) ⁽¹⁾	–	0.4	–	6	MHz
		Stabilization time ⁽²⁾	After V _{DD} reaches the minimum level of its variable range; V _{DD} = 2.0 V to 5.5 V	–	–	4	ms
Crystal Oscillator		Oscillation frequency(f _x) ⁽¹⁾	–	0.4	–	6	MHz
		Stabilization time ⁽²⁾	V _{DD} = 4.5 V to 5.5 V	–	–	10	ms
		Stabilization time ⁽²⁾	V _{DD} = 2.0 V to 5.5 V	–	–	30	
External Clock		X _{IN} input frequency(f _x) ⁽¹⁾	–	0.4	–	6	MHz
		X _{IN} input high and low level width (t _{XH} , t _{XL})	–	83.3	–	1250	ns
RC Oscillator		Frequency	V _{DD} = 5 V	0.4	–	2	MHz
			V _{DD} = 3 V	0.4	–	1	

NOTES:

- Oscillation frequency and input frequency data are for oscillator characteristics only.
- Stabilization time is the interval required for oscillator stabilization after a power-on or release of STOP mode.

Table 14-4. Recommended Oscillator Constants

($T_A = -40\text{ }^\circ\text{C} + 85\text{ }^\circ\text{C}$, $V_{DD} = 1.8\text{ V to } 5.5\text{ V}$)

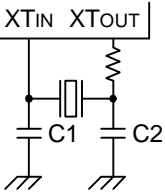
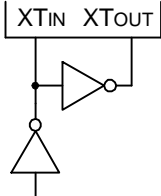
Manufacturer	Series Number ⁽¹⁾	Frequency Range	Load Cap (pF)		Oscillator Voltage Range (V)		Remarks
			C1	C2	MIN	MAX	
TDK	FCR ǒM5	3.58 MHz-6.0 MHz	33	33	2.0	5.5	Leaded Type
	FCR ǒMC5	3.58 MHz-6.0 MHz	(2)	(2)	2.0	5.5	On-chip C Leaded Type
	CCR ǒMC3	3.58 MHz-6.0 MHz	(3)	(3)	2.0	5.5	On-chip C SMD Type

NOTES:

1. Please specify normal oscillator frequency.
2. On-chip C: 30pF built in.
3. On-chip C: 38pF built in.

Table 14-5. Subsystem Clock Oscillator Characteristics

(T_A = -40 °C +85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal Oscillator		Oscillation frequency ⁽¹⁾	–	32	32.768	35	kHz
		Stabilization time ⁽²⁾	V _{DD} = 4.5 V to 5.5 V	–	1.0	2	s
			V _{DD} = 2.0 V to 5.5 V	–	–	10	
External Clock		XT _{IN} input frequency ⁽¹⁾	–	32	–	100	kHz
		XT _{IN} input high and low level width (t _{XTL} , t _{XTH})	–	5	–	15	uS

NOTES:

- Oscillation frequency and XT_{IN} input frequency data are for oscillator characteristics only.
- Stabilization time is the interval required for oscillating stabilization after a power-on occurs or release of sub clock stop

Table 14-6. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Input Capacitance	C _{IN}	f = 1 MHz; Unmeasured pins are returned to V _{SS}	–	–	15	pF
Output Capacitance	C _{OUT}					
I/O Capacitance	C _{IO}					

Table 14-7. A.C. Electrical Characteristics

(T_A = -40 °C to +85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction Cycle Time ^(note)	t _{CY}	V _{DD} = 2.7 V to 5.5 V	0.67	-	64	uS
		V _{DD} = 1.8 V to 5.5 V	1.33		64	
		With subsystem clock (fxt)	114	122	125	
TCL0 Input Frequency	f _{TI}	V _{DD} = 2.7 V to 5.5 V	0	-	1.5	MHz
		V _{DD} = 1.8 V to 5.5 V			1	kHz
TCL0 Input High, Low Width	t _{TIH}	V _{DD} = 2.7 V to 5.5 V	0.48	-	-	uS
	t _{TIL}	V _{DD} = 1.8 V to 5.5 V	1.8			
Interrupt Input High, Low Width	f _{INTH} , f _{INTL}	INT0, INT1, KS0 - KS7	10			
		K0 - K6				
RESET Input Low Width	t _{RSL}	Input	10			

NOTE: Unless otherwise specified, Instruction Cycle Time condition values assume a main system clock (fx) source.

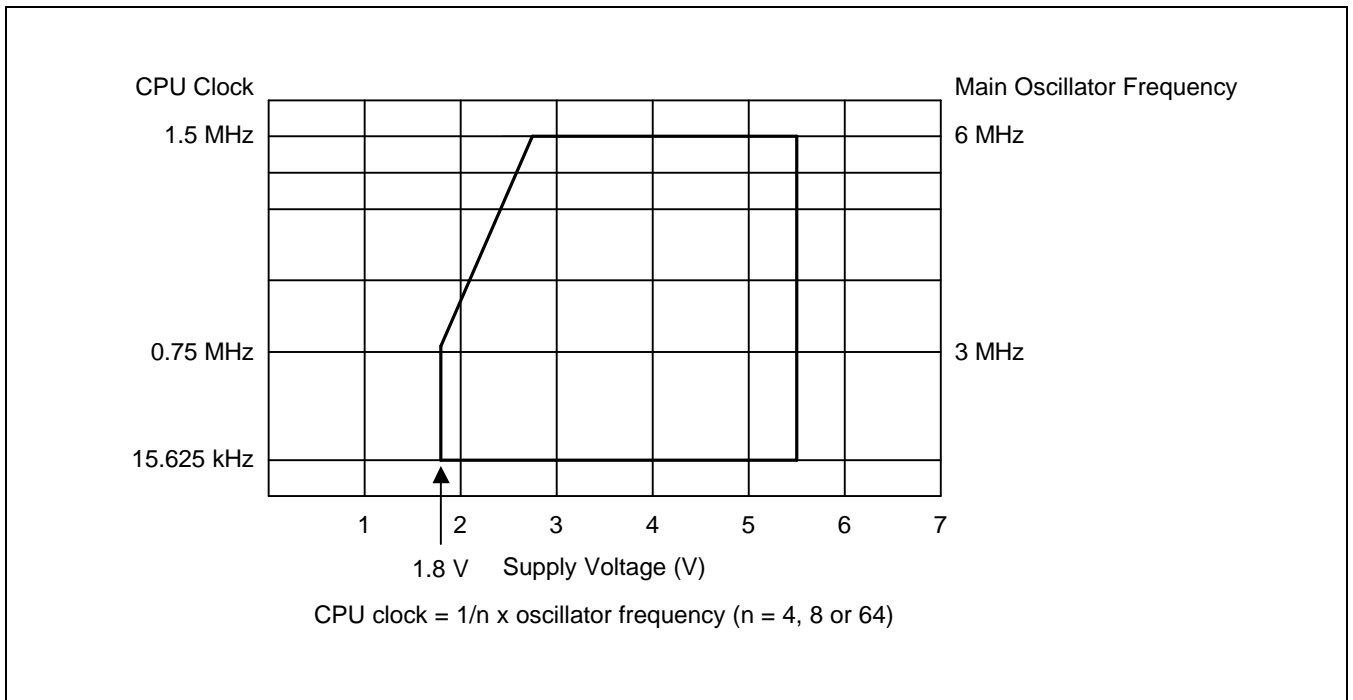


Figure 14-1. Standard Operating Voltage Range

Table 14-8. RAM Data Retention Supply Voltage in Stop Mode

(T_A = -40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V _{DDDR}	-	1.8	-	5.5	V
Data retention supply current	I _{DDDR}	V _{DDDR} = 1.8 V	-	0.1	10	uA
Release signal set time	t _{SREL}	-	0	-	-	uS
Oscillator stabilization wait time ⁽¹⁾	t _{WAIT}	Released by RESET	-	2 ¹⁷ /fx	-	ms
		Released by interrupt	-	(2)	-	

NOTES:

1. During oscillator stabilization wait time, all CPU operations must be stopped to avoid instability during oscillator start-up.
2. Use the basic timer mode register (BMOD) interval timer to delay execution of CPU instructions during the wait time.

TIMING WAVEFORMS

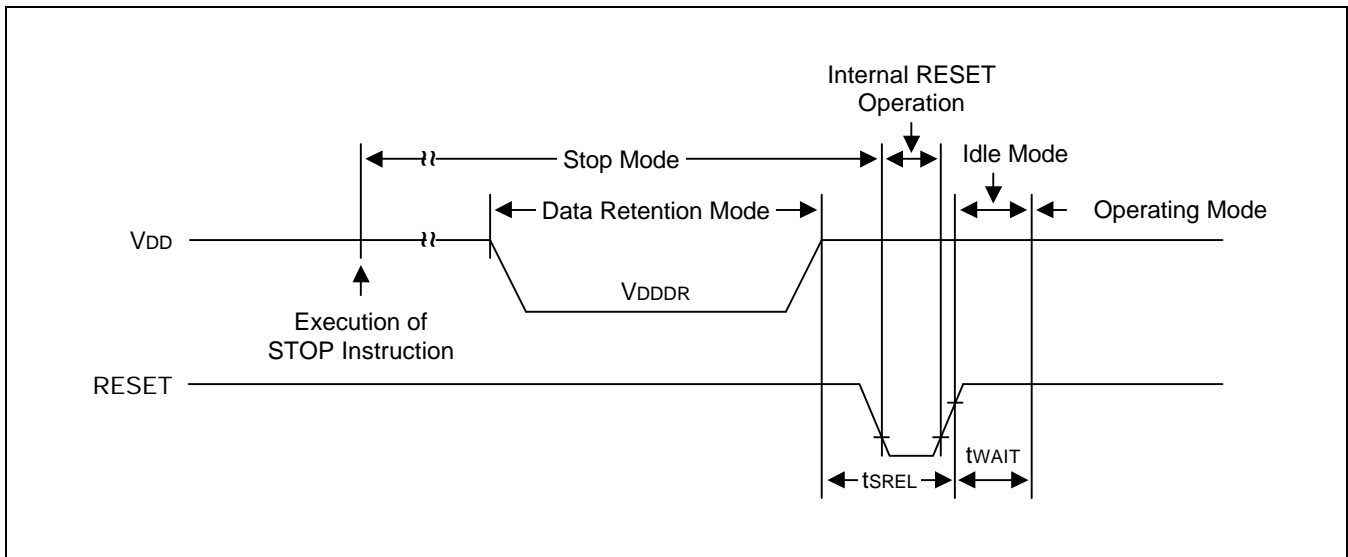


Figure 14-2. Stop Mode Release Timing When Initiated By RESET

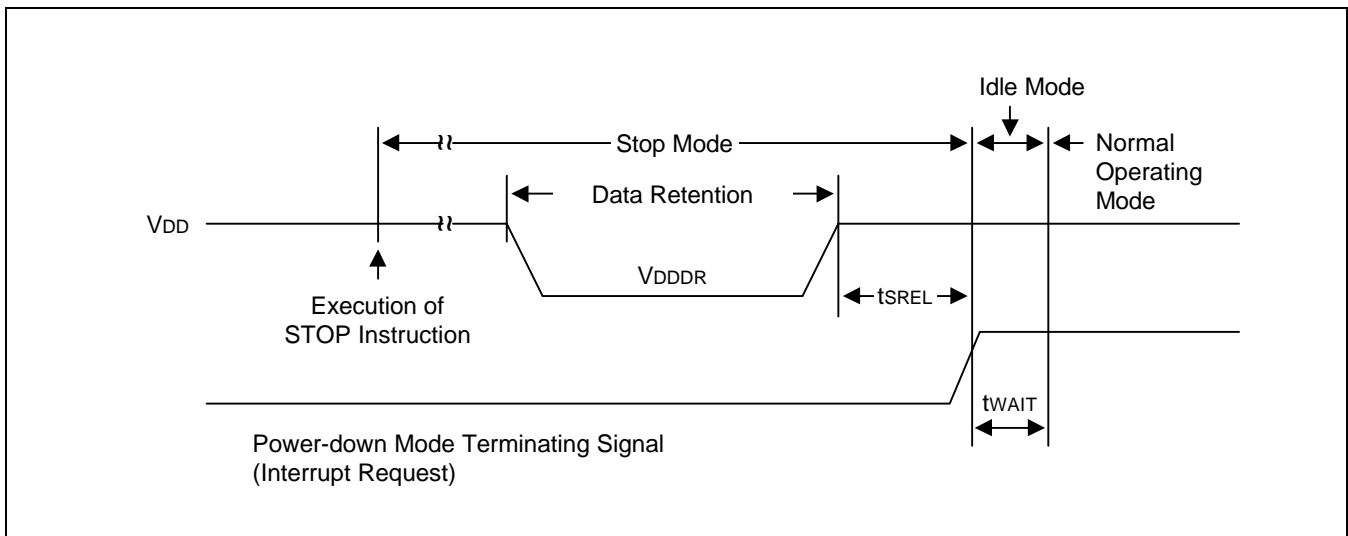


Figure 14-3. Stop Mode Release Timing When Initiated By Interrupt Request

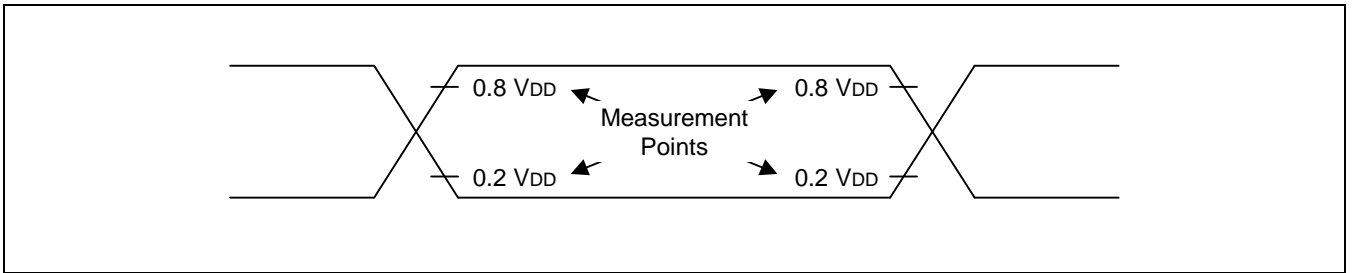


Figure 14-4. A.C. Timing Measurement Points (Except for X_{IN} and XT_{IN})

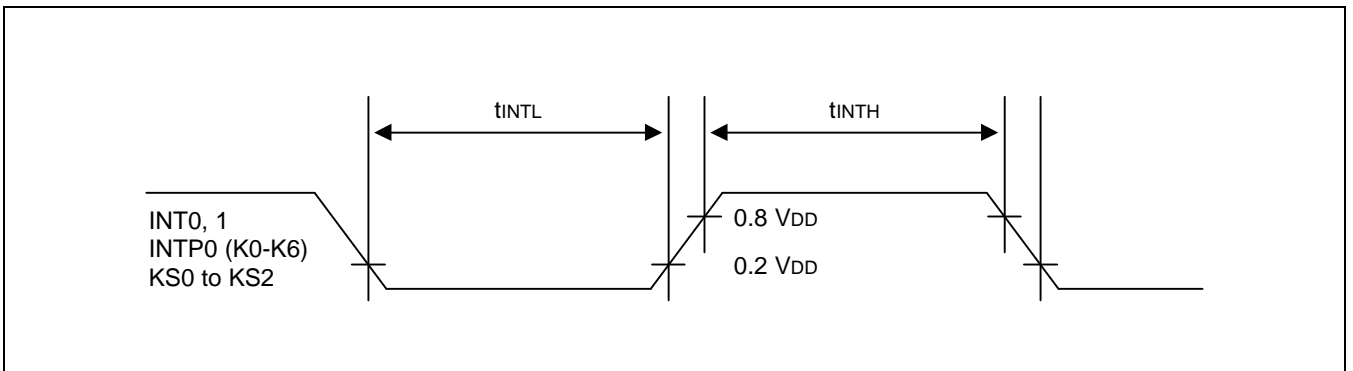
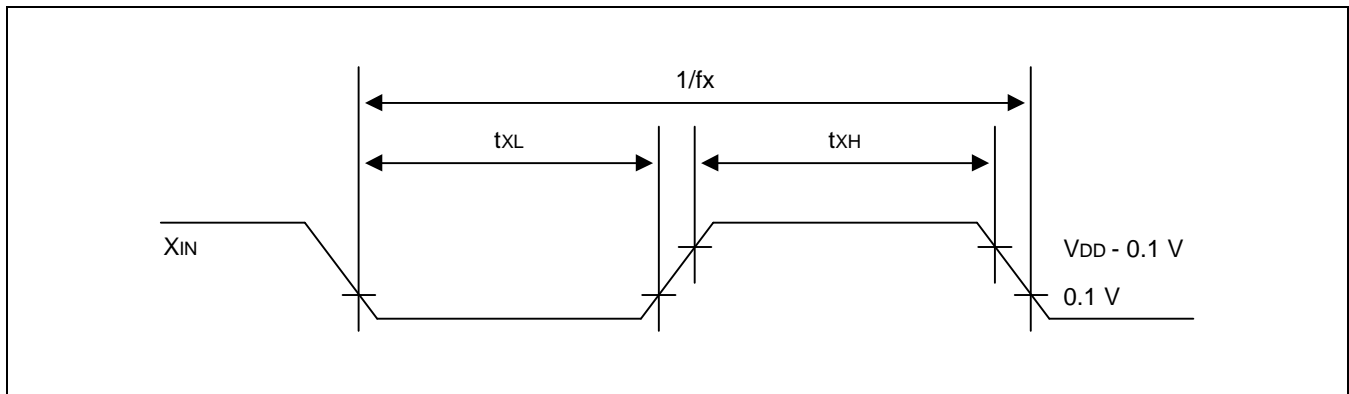
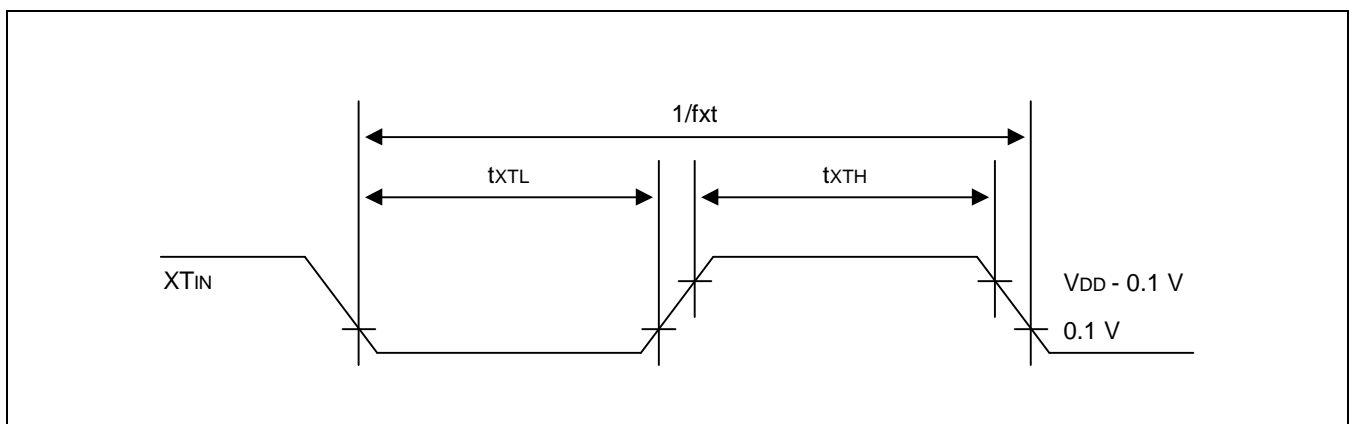


Figure 14-5. Input Timing for External Interrupts and Quasi-Interrupts

Figure 14-6. Clock Timing Measurement at X_{IN} Figure 14-7. Clock Timing Measurement at XT_{IN}

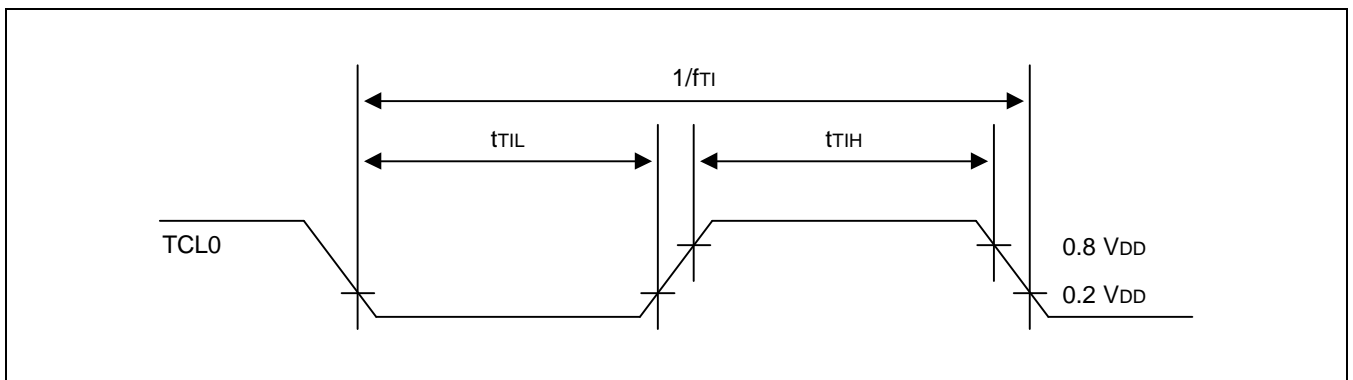


Figure 14-6. TCL0 Timing

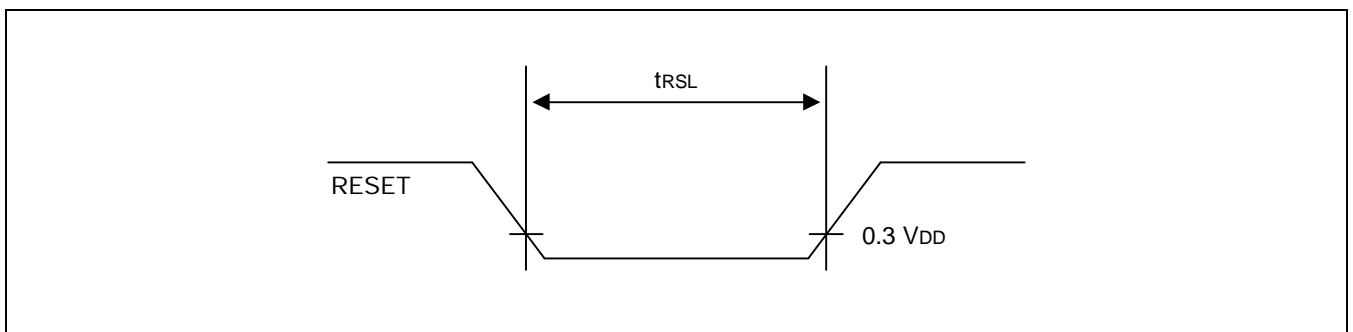


Figure 14-7. Input Timing for RESET Signal

NOTES

15 MECHANICAL DATA

OVERVIEW

The S3C72Q5 microcontroller is currently available in a 100-pin QFP package.

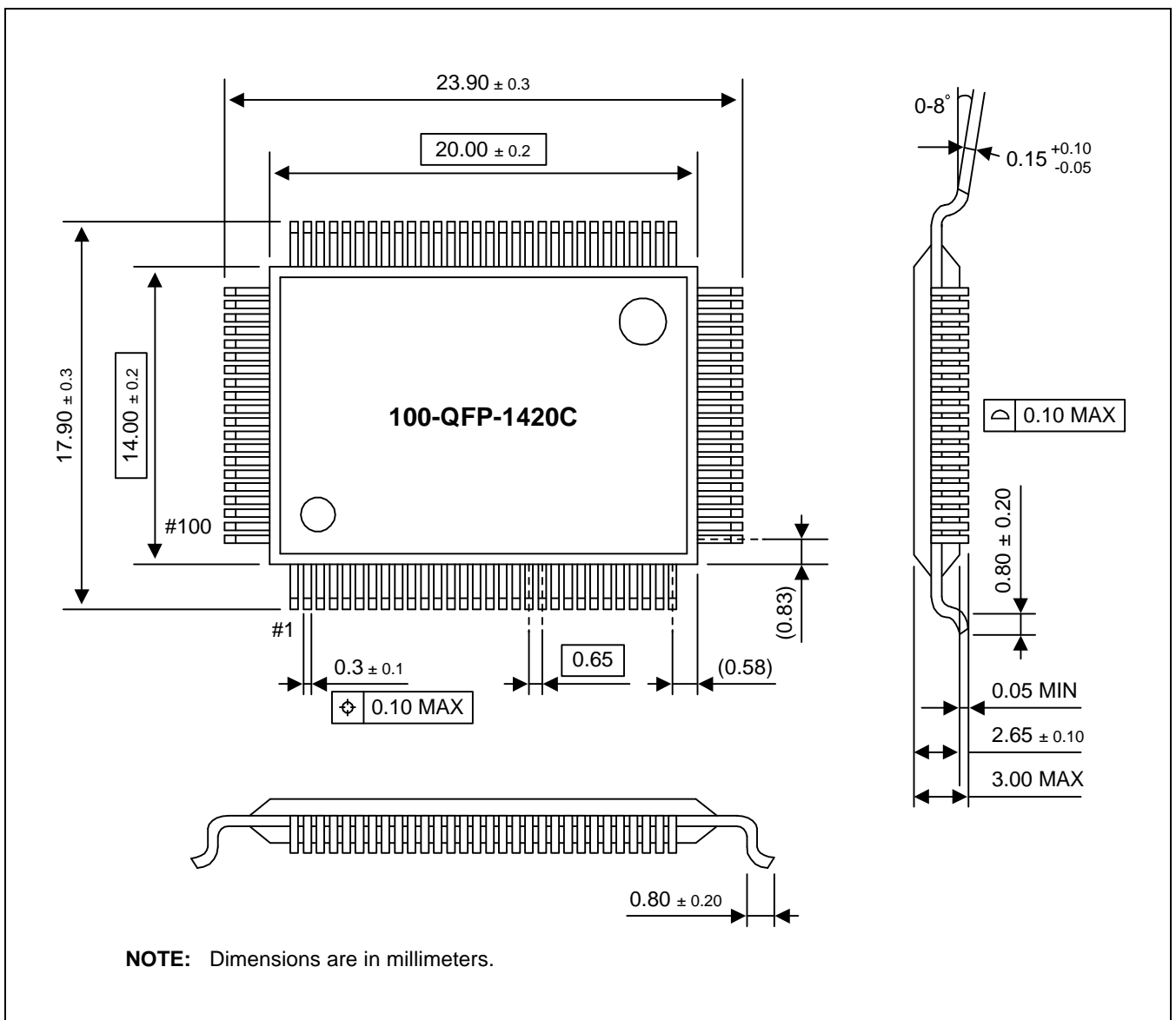


Figure 15-1. 100-QFP-1420 Package Dimensions

NOTES

16

S3P72Q5 OTP

OVERVIEW

The S3P72Q5 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C72Q5 microcontroller. It has an on-chip OTP ROM instead of masked ROM. The EPROM is accessed by serial data format.

The S3P72Q5 is fully compatible with the S3C72Q5, both in function and in pin configuration. Because of its simple programming requirements, the S3P72Q5 is ideal for use as an evaluation chip for the S3C72Q5.

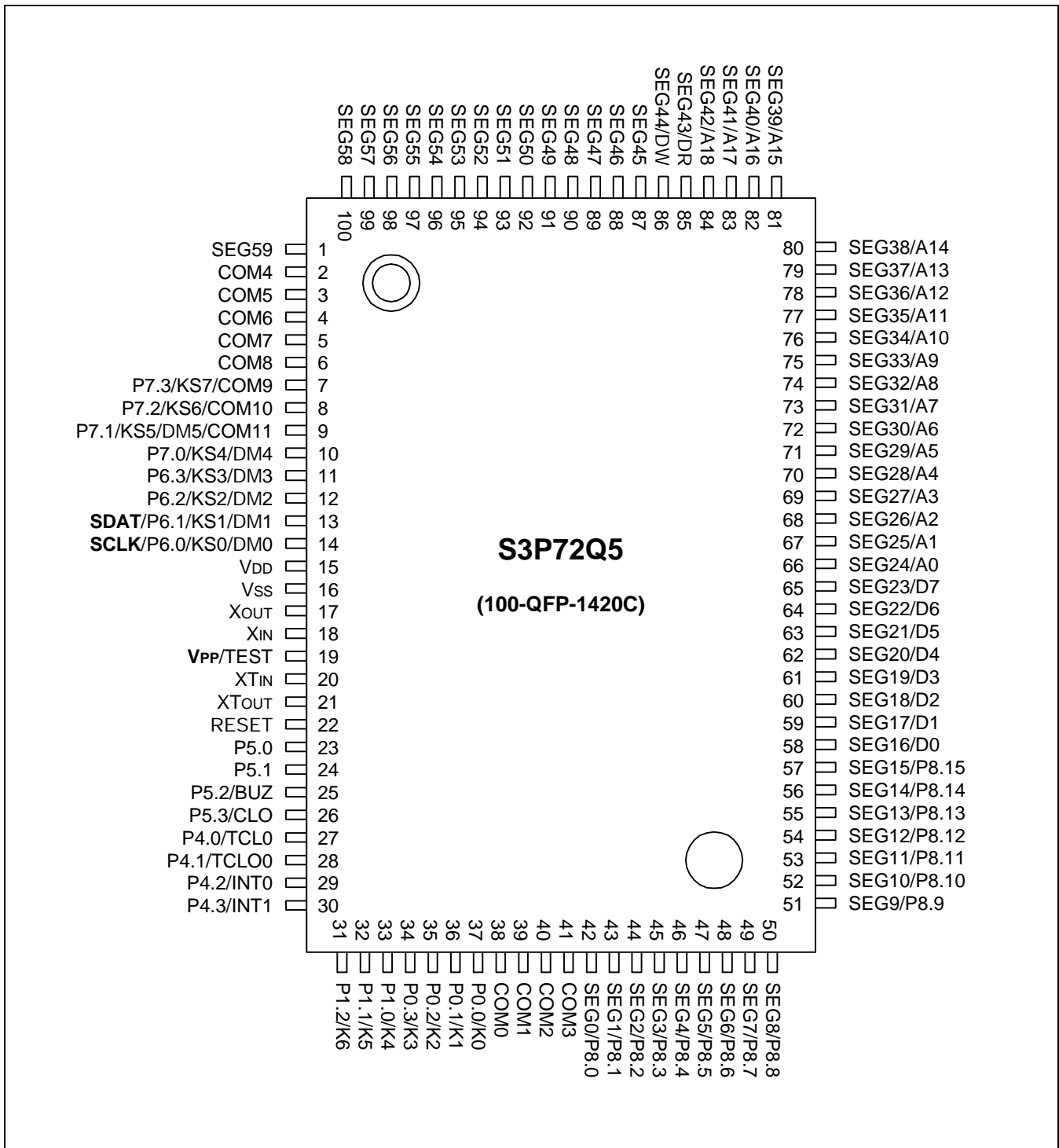


Figure 16-1. S3P72Q5 Pin Assignments (100-QFP Package)

Table 16-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P3.1	SDAT	13	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input / push-pull output port.
P3.0	SCLK	14	I/O	Serial clock pin. Input only pin.
TEST	V _{PP} (TEST)	19	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	22	I	Chip initialization
V _{DD} / V _{SS}	V _{DD} / V _{SS}	15/16	I	Logic power supply pin. V _{DD} should be tied to +5 V during programming.

Table 16-2. Comparison of S3P72Q5 and S3C72Q5 Features

Characteristic	S3P72Q5	S3C72Q5
Program Memory	16 Kbyte EPROM	16 Kbyte mask ROM
Operating Voltage (V _{DD})	1.8 V to 5.5 V	1.8 V to 5.5V
OTP Programming Mode	V _{DD} = 5 V, V _{PP} (TEST)=12.5V	
Pin Configuration	100 QFP	100 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP}(TEST) pin of the S3P72Q5, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 16-3 below.

Table 16-3. Operating Mode Selection Criteria

V _{DD}	V _{PP} (TEST)	REG/MEM	Address (A15-A0)	R/W	Mode
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

Table 16-4. D.C Characteristics

(T_A = -40 °C to +85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Units		
Supply Current (1)	I _{DD1} (2)	V _{DD} = 5 V ± 10%	6 MHz	-	4.5	8.0	mA		
		Crystal oscillator C1 = C2 = 22pF	4.19 MHz		3.2	5.5			
	I _{DD2} (2)	V _{DD} = 3 V ± 10%	6 MHz		2.0	4.0			
			4.19 MHz		1.5	3.0			
	I _{DD2} (2)	Idle mode V _{DD} = 5 V ± 10%	6 MHz		1.3	2.5			
		Crystal oscillator C1 = C2 = 22pF	4.19 MHz		1.0	1.8			
	I _{DD2} (2)	V _{DD} = 3 V ± 10%	6 MHz		0.5	1.5			
			4.19 MHz		0.4	1.0			
	I _{DD3} (3)	V _{DD} = 3 V ± 10%	32 kHz crystal oscillator		20	35		μA	
	I _{DD4} (3)	Idle mode; V _{DD} = 3 V ± 10%	32 kHz crystal oscillator		5.0	15			
I _{DD5}	Stop mode; V _{DD} = 5 V ± 10%	SCMOD = 0000B XT _{IN} = 0V	2.5	5					
	Stop mode; V _{DD} = 3 V ± 10%		0.5	3					
	V _{DD} = 5 V ± 10%	SCMOD = 0100B	0.2	3					
	V _{DD} = 3 V ± 10%		0.1	2					

NOTES:

1. Currents in the following circuits are not included; on-chip pull-up resistors, internal LCD voltage dividing resistors, output port drive currents.
2. Data includes power consumption for subsystem clock oscillation.
3. When the system clock control register, SCMOD, is set to 1001B, main system clock oscillation stops and the subsystem clock is used.
4. Every values in this table is measured when the power control register (PCON) is set to "0011B".

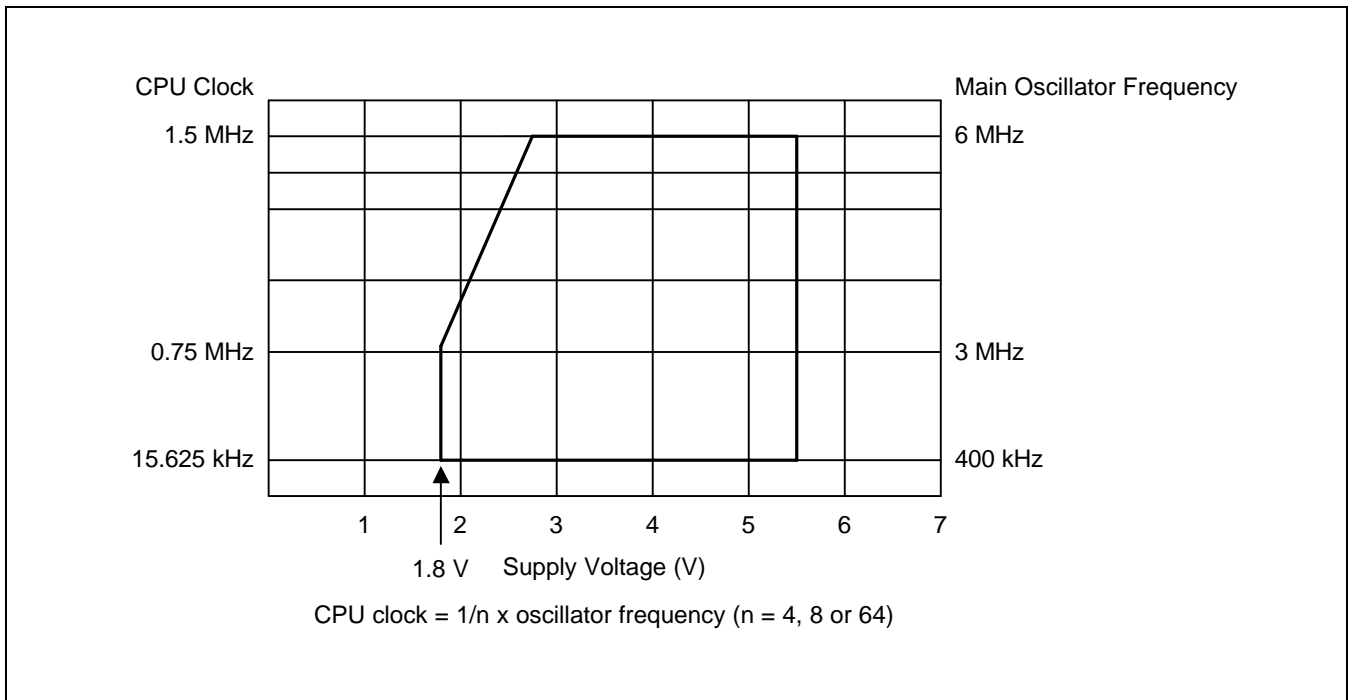


Figure 16-2. Standard Operating Voltage Range

NOTES

17

DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for S3C7, S3C9, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM57

The SASM57 is an relocatable assembler for Samsung's S3C7-series microcontrollers. The SASM57 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM57 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code (.OBJ file) by HEX2ROM, the value 'FF' is filled into the unused ROM area upto the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all S3C7-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the S3C72Q5 microcontroller and OTP programmer (Gang) are now available.

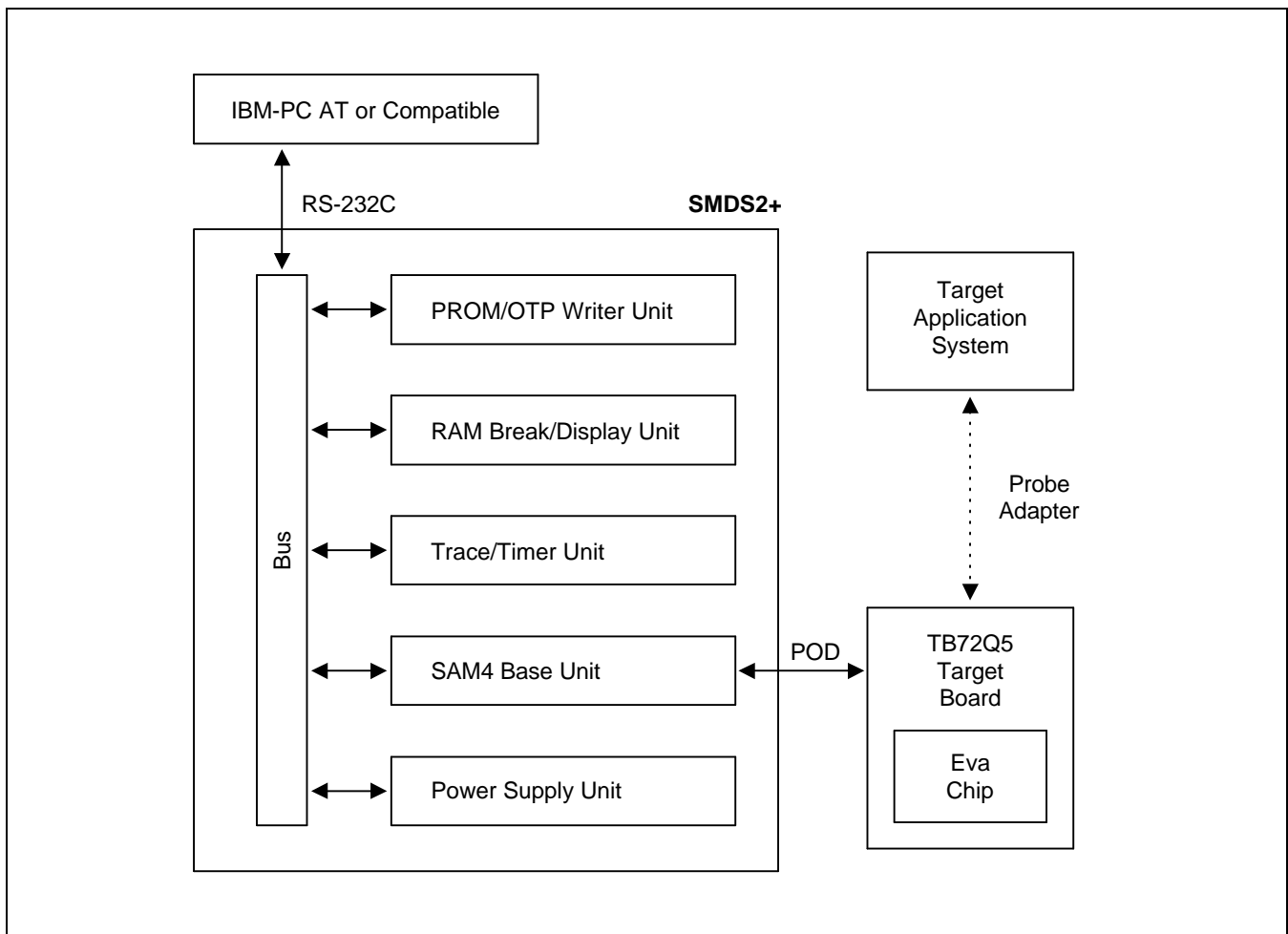


Figure 17-1. SMDS Product Configuration (SMDS2+)

TB72Q5 TARGET BOARD

The TB72Q5 target board is used for the S3C72Q5 microcontroller. It is supported by the SMDS2+ development system.

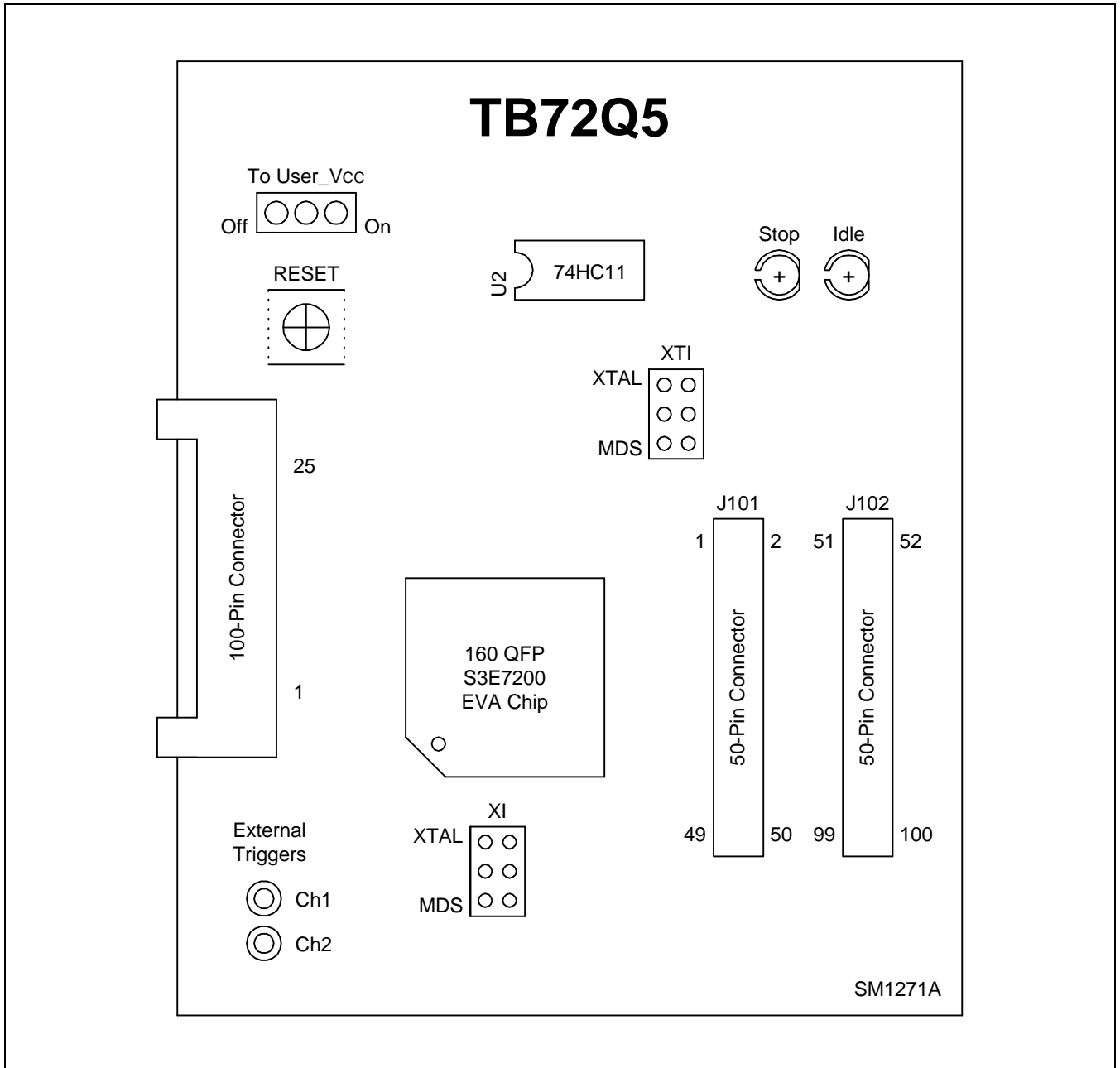


Figure 17-2. TB72Q5 Target Board Configuration

Table 17-1. Power Selection Settings for TB72Q5

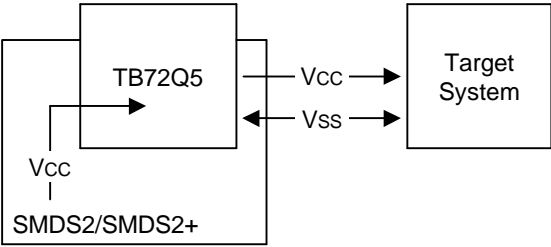
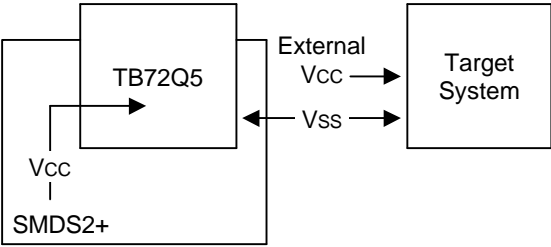
'To User_Vcc' Settings	Operating Mode	Comments
<p>To User_VCC</p> <p>Off <input type="radio"/> <input checked="" type="radio"/> <input checked="" type="radio"/> On</p>		<p>The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.</p>
<p>To User_VCC</p> <p>Off <input type="radio"/> <input checked="" type="radio"/> <input checked="" type="radio"/> On</p>		<p>The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.</p>

Table 17-2. Main-clock Selection Settings for TB72Q5

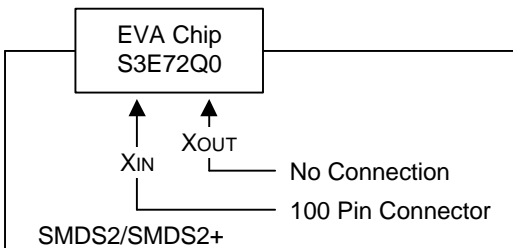
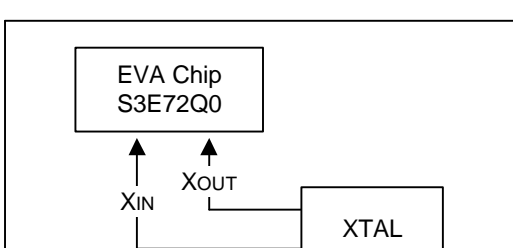
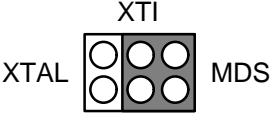
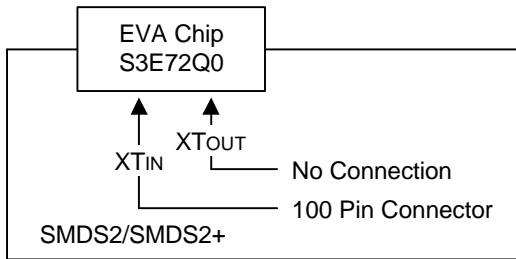
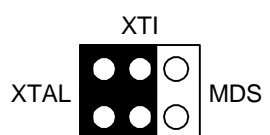
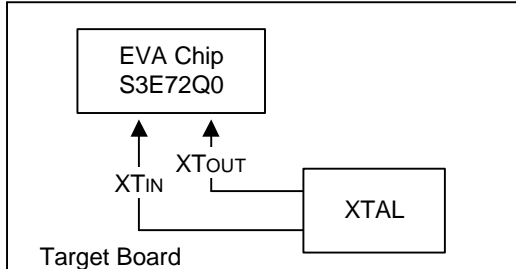
Sub Clock Setting	Operating Mode	Comments
<p>XI</p> <p>XTAL <input type="radio"/> <input checked="" type="radio"/> <input checked="" type="radio"/> <input checked="" type="radio"/> MDS</p>		<p>Set the XI switch to "MDS" when the target board is connected to the SMDS2/SMDS2+.</p>
<p>XI</p> <p>XTAL <input checked="" type="radio"/> <input checked="" type="radio"/> <input checked="" type="radio"/> <input type="radio"/> MDS</p>		<p>Set the XI switch to "XTAL" when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.</p>

Table 17-3. Sub-clock Selection Settings for TB72Q5

Sub Clock Setting	Operating Mode	Comments
		<p>Set the XTI switch to “MDS” when the target board is connected to the SMDS2/SMDS2+.</p>
		<p>Set the XTI switch to “XTAL” when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.</p>

IDLE LED

This LED is ON when the evaluation chip (S3E72Q0) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (S3E72Q0) is in stop mode.

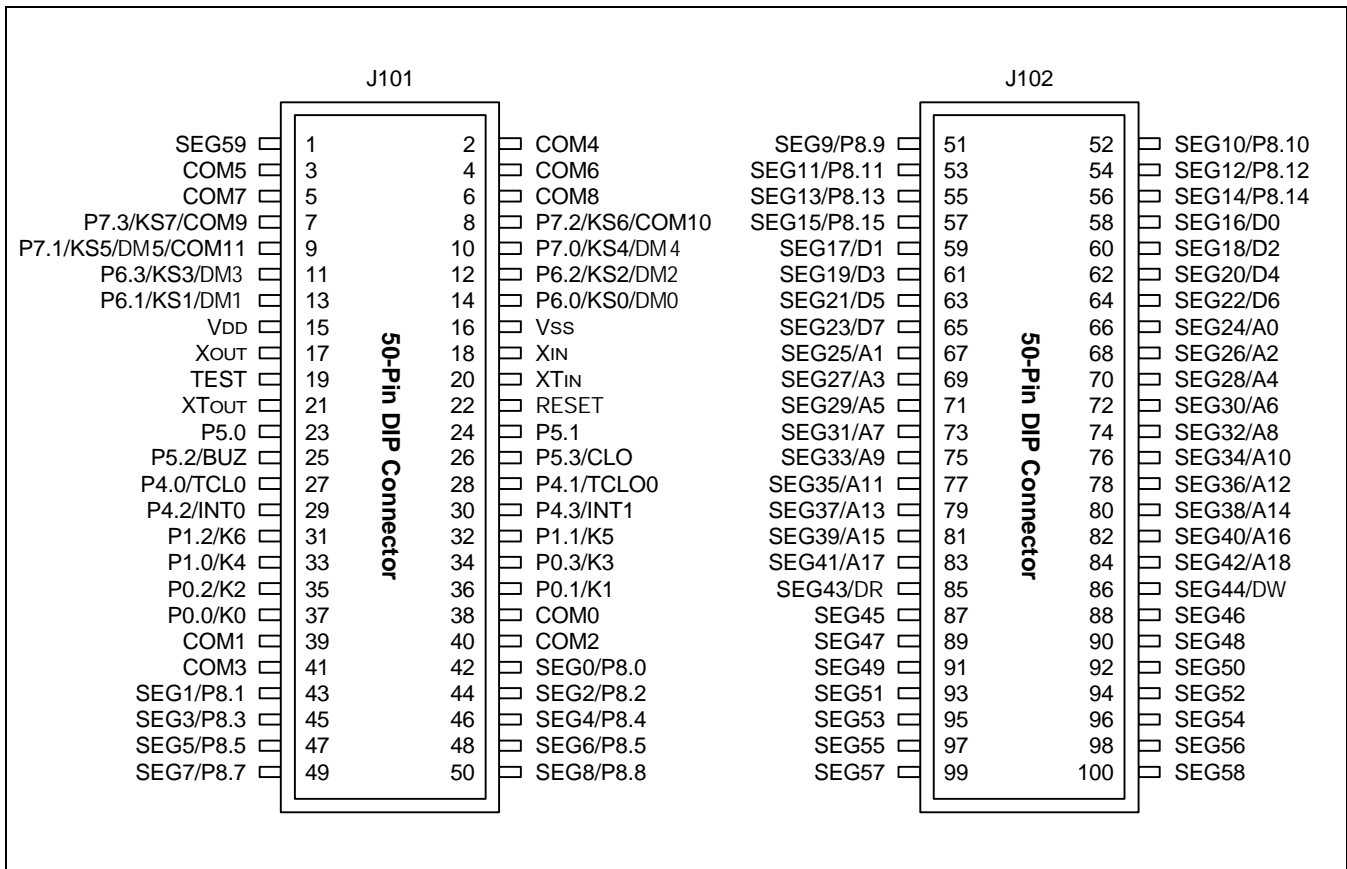


Figure 17-3. 50-Pin Connectors for TB72Q5

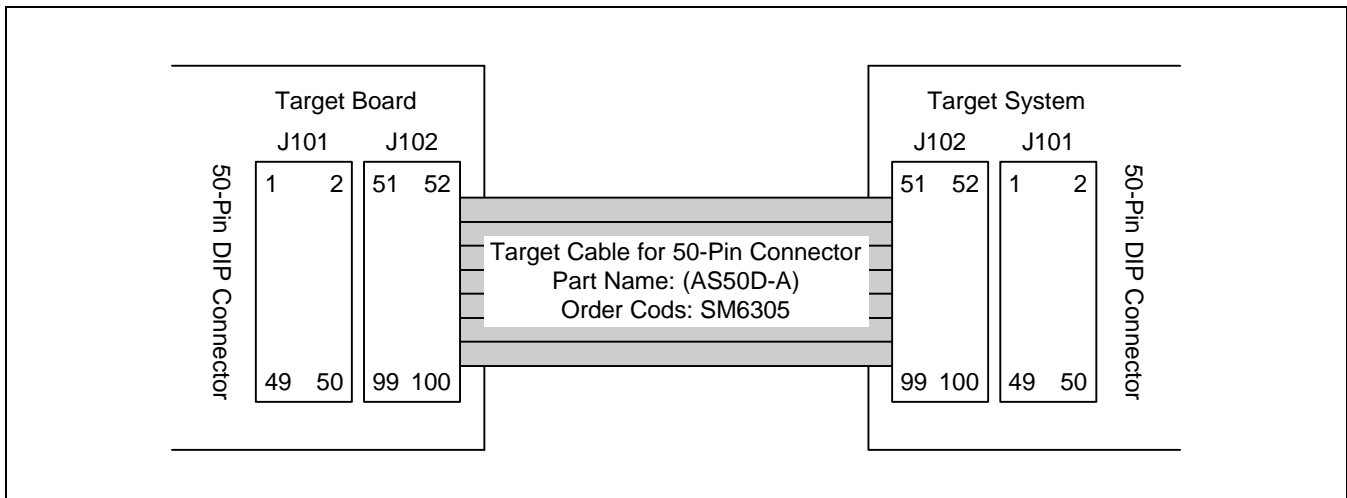


Figure 17-4. TB72Q5 Adapter Cable for 100-QFP Package (S3C72Q5/P72Q5)

S3C7 SERIES MASK ROM ORDER FORM

Product description:

Device Number: S3C7 _____ - _____ (write down the ROM code number)

Product Order Form: Package Pellet Wafer Package Type: _____

Package Marking (Check One):

- Standard
 Custom A (Max 10 chars)
 Custom B (Max 10 chars each line)

SEC @ YWW Device Name

@ YWW Device Name _____

@ YWW _____ _____

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

Delivery Dates and Quantities:

Deliverable	Required Delivery Date	Quantity	Comments
ROM code	-	Not applicable	See ROM Selection Form
Customer sample			
Risk order			See Risk Order Sheet

Please answer the following questions:

For what kind of product will you be using this order?

- New product
 Upgrade of an existing product
 Replacement of an existing product
 Other

If you are replacing an existing product, please indicate the former product name
(_____)

What are the main reasons you decided to use a Samsung microcontroller in your product?

Please check all that apply.

- Price
 Product quality
 Features and functions
 Development system
 Technical support
 Delivery on time
 Used same micom before
 Quality of documentation
 Samsung reputation

Mask Charge (US\$ / Won): _____

Customer Information:

Company Name: _____ Telephone number: _____

Signatures: _____
 (Person placing the order) (Technical Manager)

S3C7 SERIES

REQUEST FOR PRODUCTION AT CUSTOMER RISK

Customer Information:

Company Name: _____

Department: _____

Telephone Number: _____ Fax: _____

Date: _____

Risk Order Information:

Device Number: S3C7 _____ - _____ (write down the ROM code number)

Package: _____ Number of Pins: _____ Package Type: _____

Intended Application: _____

Product Model Number: _____

Customer Risk Order Agreement:

We hereby request SEC to produce the above named product in the quantity stated below. We believe our risk order product to be in full compliance with all SEC production specifications and, to this extent, agree to assume responsibility for any and all production risks involved.

Order Quantity and Delivery Schedule:

Risk Order Quantity: _____ PCS

Delivery Schedule:

Delivery Date (s)	Quantity	Comments

Signatures: _____

(Person Placing the Risk Order)

(SEC Sales Representative)

S3C72Q5 MASK OPTION SELECTION FORM

Device Number: S3C72Q5-_____ (write down the ROM code number)


Attachment (Check one): Diskette PROM

Customer Checksum: _____

Company Name: _____

Signature (Engineer): _____

Please answer the following questions:

 **Application** (Product Model ID: _____)

- | | | |
|---------------------------------------|---|--|
| <input type="checkbox"/> Audio | <input type="checkbox"/> Video | <input type="checkbox"/> Telecom |
| <input type="checkbox"/> LCD Databank | <input type="checkbox"/> Caller ID | <input type="checkbox"/> LCD Game |
| <input type="checkbox"/> Industrials | <input type="checkbox"/> Home Appliance | <input type="checkbox"/> Office Automation |
| <input type="checkbox"/> Remocon | <input type="checkbox"/> Other | |

Please describe in detail its application

S3C7 SERIES OTP FACTORY WRITING ORDER FORM (1/2)

Product Description:

Device Number: S3P _____ - _____ (write down the ROM code number)

Product Order Form: Package Pellet Wafer

If the product order form is package: Package Type: _____

Package Marking (Check One):

Standard Custom A (Max 10 chars) Custom B (Max 10 chars each line)

SEC @ YWW Device Name

@ YWW Device Name _____

@ YWW _____ _____

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

Delivery Dates and Quantity:

ROM Code Release Date	Required Delivery Date of Device	Quantity

Please answer the following questions:

What is the purpose of this order?

- New product development Upgrade of an existing product
 Replacement of an existing microcontroller Other

If you are replacing an existing microcontroller, please indicate the former microcontroller name ()

What are the main reasons you decided to use a Samsung microcontroller in your product?

Please check all that apply.

- Price Product quality Features and functions
 Development system Technical support Delivery on time
 Used same micom before Quality of documentation Samsung reputation

Customer Information:

Company Name: _____ Telephone number _____

Signatures: _____ (Person placing the order) _____ (Technical Manager)

S3P72Q5 OTP FACTORY WRITING ORDER FORM (2/2)

Device Number: S3P ____ - _____ (write down the ROM code number)

Customer Checksums: _____

Company Name: _____

Signature (Engineer): _____


Read Protection ⁽¹⁾: **Yes** **No**

Please answer the following questions:

 **Are you going to continue ordering this device?**

Yes **No**

If so, how much will you be ordering? _____ pcs

 **Application** (Product Model ID: _____)

- | | | |
|---------------------------------------|---|--|
| <input type="checkbox"/> Audio | <input type="checkbox"/> Video | <input type="checkbox"/> Telecom |
| <input type="checkbox"/> LCD Databank | <input type="checkbox"/> Caller ID | <input type="checkbox"/> LCD Game |
| <input type="checkbox"/> Industrials | <input type="checkbox"/> Home Appliance | <input type="checkbox"/> Office Automation |
| <input type="checkbox"/> Remocon | <input type="checkbox"/> Other | |

Please describe in detail its application

NOTES

1. Once you choose a read protection, you cannot read again the programming code from the EPROM.
2. FLASH MCU Writing will be executed in our manufacturing site.
3. The writing program is completely verified by a customer. Samsung does not take on any responsibility for errors occurred from the writing program.