

## COP8TAB9/TAC9

# 8-Bit CMOS Flash Microcontroller with 2k Byte or 4k Byte Memory

## 1.0 General Description

The COP8TAB9/TAC9 Flash microcontrollers are highly integrated COP8™ Feature core devices, with 2k or 4k Flash memory and advanced features. These single-chip CMOS devices are suited for applications requiring a full featured, in-system reprogrammable controller with moderate memory and low EMI. The same device is used for development, pre-production and volume production with a range of COP8 software and hardware development tools.

A Masked ROM device (COP8TAB5/TAC5) has been developed and provides identical features except for the Boot ROM and Flash Memory and related features such as In-

System Programming and reprogrammability. The ROM device is supported, in emulation, by this device.

The lack of the Boot ROM and Flash Memory in the ROM device, prompts us to caution the user, utilizing the COP8TAx9 Flash based devices during development for applications to be produced using the COP8TAx5 ROM devices, to ensure that code contains NO calls to Boot ROM functions prior to submission for ROM generation. Instances of the JSRB instruction in ROM based devices will be executed as a JSR instruction to a location in the first 256 bytes of Program Memory.

Device included in this datasheet:

Device	Flash Program Memory (kbytes)	RAM (bytes)	I/O Pins	Packages	Temperature
COP8TAB9	2	128	16, 24 or 40	20 and 28 SOIC WIDE, 44 LLP	-40°C to +85°C
COP8TAC9	4	128			

## 2.0 Features

### KEY FEATURES

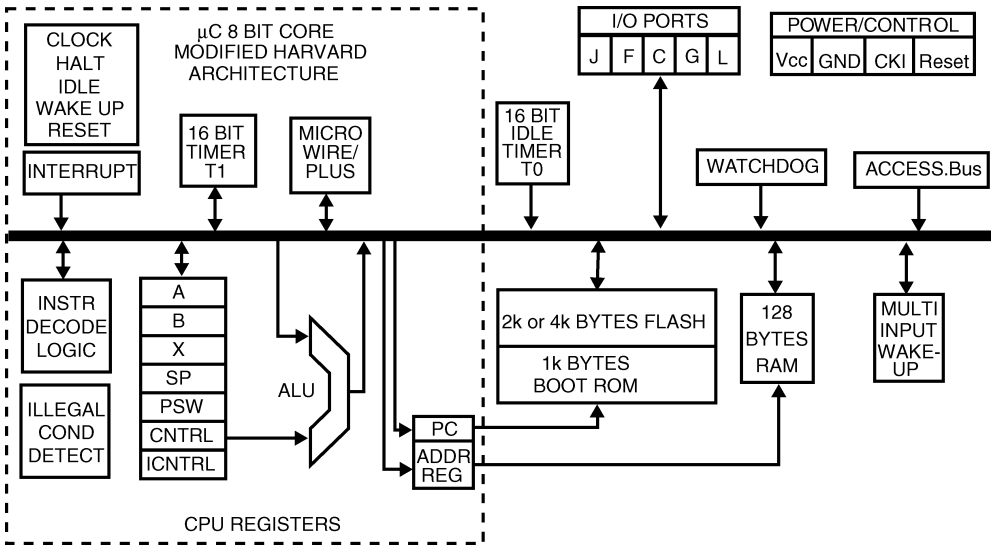
- 2k bytes or 4k bytes Flash Program Memory, with Security Feature, organized in 512 byte pages that can be erased or written individually
- 128 bytes volatile RAM
- 2.25V – 2.75V In-System Programmability of Flash
- High endurance - 20k Erase/Write Cycles
- Superior Data Retention - 100 years
- Crystal Oscillator at 15 MHz or Integrated RC Oscillator at 10MHz
- Clock Prescaler For Adjusting Power Dissipation to Processing Requirements
- Power-On Reset
- HALT/IDLE Power Save Modes
- One 16-bit timer:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- High Current I/Os
  - 10 mA @ 0.4V

### OTHER FEATURES

- Single supply operation:
  - 2.25V–2.75V (-40°C to +85°C)
- Quiet Design (low radiated emissions)
- Multi-Input Wake-Up with optional interrupts
- MICROWIRE/PLUS (Serial Peripheral Interface Compatible)

- ACCESS.Bus Synchronous Serial Interface (compatible with I2C™ and SMBus™)
  - Master Mode and Slave Mode
  - Full Master Mode Capability
  - Bus Speed Up To 400KBits/Sec
  - Low Power Mode With Wake-Up Detection
  - Optional 1.8V ACCESS.Bus Compatibility
- Eight multi-source vectored interrupts servicing:
  - External Interrupt
  - Idle Timer T0
  - One Timers (with 2 interrupts)
  - MICROWIRE/PLUS Serial peripheral interface
  - ACCESS.Bus/I²C/SMBus compatible Synchronous Serial Interface
  - Multi-Input Wake-Up
  - Software Trap
- Idle Timer with programmable interrupt interval
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers
- True bit manipulation
- WATCHDOG and Clock Monitor logic
- Software selectable I/O options
  - TRI-STATE Output/High Impedance Input
  - Push-Pull Output
  - Weak Pull Up Input
- Schmitt trigger inputs on I/O ports
- Temperature range: -40°C to +85°C
- Packaging: 20 and 28 SOIC and 44 LLP

3.0 Block Diagram



4.0 Ordering Information

Part Numbering Scheme

COP8	TA	C	9	H	LQ	8
	Family and Feature Set Indicator	Program Memory Size	Program Memory Type	No. Of Pins	Package Type	Temperature
		B = 2k C = 4k	5 = Masked ROM 9 = Flash	C = 20 Pin E = 28 Pin H = 44 Pin	LQ = LLP MW = SOIC WIDE	8 = -40 to +85°C

**Note:** The user, utilizing the COP8TAx9 Flash based devices during development for applications to be produced using the COP8TAx5 ROM devices, is cautioned to ensure that code contains NO calls to Boot ROM functions prior to submission for ROM generation. Instances of the JSRB instruction in ROM based devices will be executed as a JSR instruction to a location in the first 256 bytes of Program Memory.

Flash and ROM devices are not 100% identical. The execution of the JSRB instruction is an example of the potential differences between the devices. For this reason, the user is strongly advised to obtain a masked ROM prototype devices before committing to production quantities. This will allow the user to ensure there are no unexpected differences between Flash an ROM devices within the application.

## Table of Contents

1.0 General Description .....	1
2.0 Features .....	1
3.0 Block Diagram .....	2
4.0 Ordering Information .....	2
5.0 Connection Diagrams .....	6
6.0 Architectural Overview .....	8
6.1 EMI REDUCTION .....	8
6.2 IN-SYSTEM PROGRAMMING .....	8
6.3 TRUE IN-SYSTEM EMULATION .....	8
6.4 ARCHITECTURE .....	8
6.5 INSTRUCTION SET .....	8
6.5.1 Key Instruction Set Features .....	8
6.5.2 Single Byte/Single Cycle Code Execution .....	8
6.5.3 Many Single-Byte, Multi-Function Instructions .....	8
6.5.4 Bit-Level Control .....	8
6.5.5 Register Set .....	9
6.6 PACKAGING/PIN EFFICIENCY .....	9
7.0 Absolute Maximum Ratings .....	10
8.0 Electrical Characteristics .....	10
9.0 Pin Descriptions .....	13
9.1 EMULATION CONNECTION .....	15
10.0 Functional Description .....	15
10.1 CPU REGISTERS .....	15
10.2 PROGRAM MEMORY .....	16
10.3 DATA MEMORY .....	16
10.4 OPTION REGISTER .....	16
10.5 SECURITY .....	17
10.6 RESET .....	17
10.6.1 External Reset .....	18
10.6.2 On-Chip Power-On Reset .....	18
10.7 OSCILLATOR CIRCUITS .....	19
10.7.1 R/C Oscillator .....	19
10.7.2 Crystal Oscillator .....	20
10.7.3 External Oscillator .....	20
10.7.4 Clock Prescaler .....	20
10.8 CONTROL REGISTERS .....	21
10.8.1 CNTRL Register (Address X'00EE) .....	21
10.8.2 PSW Register (Address X'00EF) .....	21
10.8.3 ICNTRL Register (Address X'00E8) .....	21
10.8.4 ITMR Register (Address X'00CF) .....	21
11.0 In-System Programming .....	21
11.1 INTRODUCTION .....	21
11.2 FUNCTIONAL DESCRIPTION .....	21
11.3 REGISTERS .....	22
11.3.1 ISP Address Registers .....	22
11.3.2 ISP Read Data Register .....	22
11.3.3 ISP Write Data Register .....	22
11.3.4 ISP Write Timing Register .....	22
11.4 MANEUVERING BACK AND FORTH BETWEEN FLASH MEMORY AND BOOT ROM .....	23
11.5 FORCED EXECUTION FROM BOOT ROM .....	23
11.6 RETURN TO FLASH MEMORY WITHOUT HARDWARE RESET .....	24
11.7 MICROWIRE/PLUS ISP .....	24
12.0 Timers .....	25
12.1 TIMER T0 (IDLE TIMER) .....	25
12.1.1 ITMR Register .....	25
12.2 TIMER T1 .....	26
12.3 MODE 1. PROCESSOR INDEPENDENT PWM MODE .....	26
12.4 MODE 2. EXTERNAL EVENT COUNTER MODE .....	26
12.5 MODE 3. INPUT CAPTURE MODE .....	27
12.6 TIMER CONTROL FLAGS .....	28
13.0 Power Save Modes .....	28
13.1 HALT MODE .....	28

## Table of Contents (Continued)

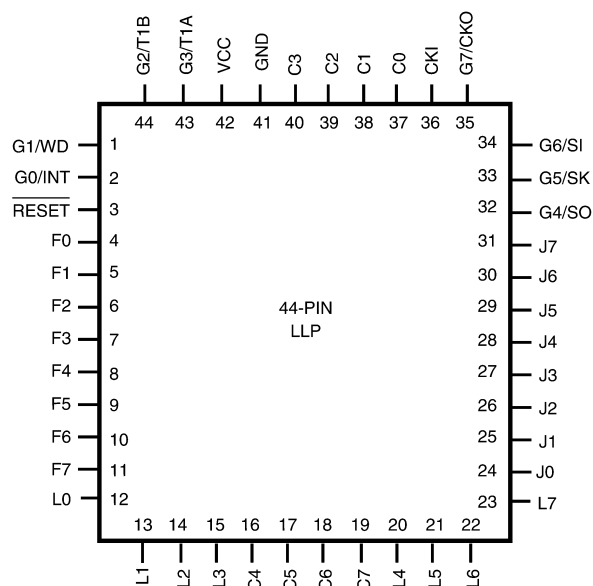
13.2 IDLE MODE .....	29
13.3 MULTI-INPUT WAKE-UP .....	30
14.0 Interrupts .....	31
14.1 INTRODUCTION .....	31
14.2 MASKABLE INTERRUPTS .....	32
14.3 VIS INSTRUCTION .....	33
14.3.1 VIS Execution .....	34
14.4 NON-MASKABLE INTERRUPT .....	35
14.4.1 Pending Flag .....	35
14.4.2 Software Trap .....	35
14.4.2.1 Programming Example: External Interrupt .....	36
14.5 PORT C AND PORT L INTERRUPTS .....	37
14.6 INTERRUPT SUMMARY .....	37
15.0 WATCHDOG/Clock Monitor .....	37
15.1 CLOCK MONITOR .....	38
15.2 WATCHDOG/CLOCK MONITOR OPERATION .....	38
15.3 WATCHDOG AND CLOCK MONITOR SUMMARY .....	38
15.4 DETECTION OF ILLEGAL CONDITIONS .....	39
16.0 MICROWIRE/PLUS .....	39
16.1 MICROWIRE/PLUS OPERATION .....	39
16.2 MICROWIRE/PLUS MASTER MODE OPERATION .....	40
16.3 MICROWIRE/PLUS SLAVE MODE OPERATION .....	40
16.4 ALTERNATE SK PHASE OPERATION AND SK IDLE POLARITY .....	40
17.0 ACCESS.Bus Interface .....	42
17.1 DATA TRANSACTIONS .....	42
17.1.1 Start and Stop .....	43
17.1.2 Acknowledge Cycle .....	43
17.1.3 Addressing Transfer Formats .....	43
17.2 BUS ARBITRATION .....	43
17.3 POWER SAVE MODES .....	43
17.4 SDA AND SCL DRIVER CONFIGURATION .....	43
17.5 ACB SERIAL DATA REGISTER (ACBSDA) .....	44
17.6 ACB STATUS REGISTER (ACBST) .....	44
17.7 ACB CONTROL STATUS REGISTER (ACBCST) .....	44
17.8 ACB CONTROL 1 REGISTER (ACBCTL1) .....	44
17.9 ACB CONTROL REGISTER 2 (ACBCTL2) .....	45
17.10 ACB OWN ADDRESS REGISTER (ACBADDR) .....	45
18.0 Memory Map .....	45
19.0 Instruction Set .....	46
19.1 INTRODUCTION .....	46
19.2 INSTRUCTION FEATURES .....	46
19.3 ADDRESSING MODES .....	46
19.3.1 Operand Addressing Modes .....	47
19.3.2 Transfer-of-Control Addressing Modes .....	48
19.4 INSTRUCTION TYPES .....	48
19.4.1 Arithmetic Instructions .....	48
19.4.2 Transfer-of-Control Instructions .....	48
19.4.3 Load and Exchange Instructions .....	49
19.4.4 Logical Instructions .....	49
19.4.5 Accumulator Bit Manipulation Instructions .....	49
19.4.6 Stack Control Instructions .....	49
19.4.7 Memory Bit Manipulation Instructions .....	49
19.4.8 Conditional Instructions .....	49
19.4.9 No-Operation Instruction .....	49
19.5 REGISTER AND SYMBOL DEFINITION .....	49
19.6 INSTRUCTION SET SUMMARY .....	50
19.7 INSTRUCTION EXECUTION TIME .....	51
20.0 Development Support .....	54
20.1 TOOLS ORDERING NUMBERS FOR THE COP8TA 2.5V FLASH FAMILY DEVICES .....	54
20.2 COP8 TOOLS OVERVIEW .....	55
20.3 WHERE TO GET TOOLS .....	56

**Table of Contents** (Continued)

21.0 Revision History ..... 57

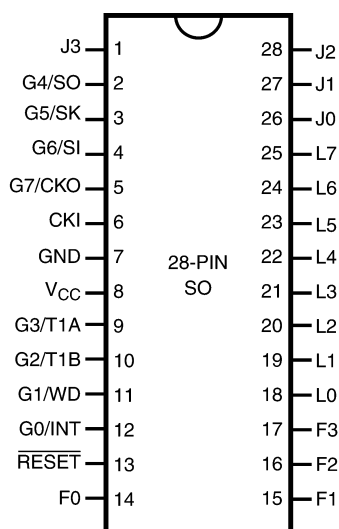
22.0 Physical Dimensions ..... 58

## 5.0 Connection Diagrams



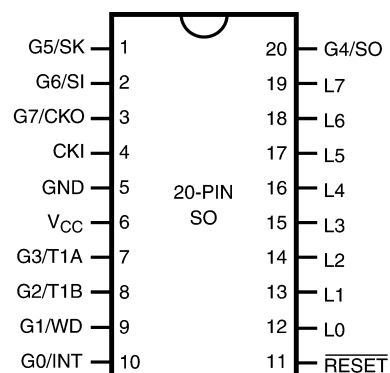
20047502

**Top View**  
**44 Pin LLP Package**  
 See NS Package Number LQA44A



20047504

**Top View**  
**28 Pin Plastic SOIC WIDE Package**  
 See NS Package Number M28B



20047505

**Top View**  
**20 Pin Plastic SOIC WIDE Package**  
 See NS Package Number M20B

Pinouts for 44-, 20- and 28-Pin Packages

Port	Type	Alt. Function	In System Emulation Mode	44-Pin LLP <sup>a</sup>	28-Pin SOIC <sup>a</sup>	20-Pin SOIC <sup>a</sup>
L0	I/O	MIWU/SDA		12	18	12
L1	I/O	MIWU/SCL		13	19	13
L2	I/O	MIWU		14	20	14
L3	I/O	MIWU		15	21	15
L4	I/O	MIWU		20	22	16
L5	I/O	MIWU		21	23	17
L6	I/O	MIWU		22	24	18
L7	I/O	MIWU		23	25	19
G0	I/O	INT	Input	2	12	10
G1	I/O	WDOUT <sup>a</sup>	POUT	1	11	9
G2	I/O	T1B	Output	44	10	8
G3	I/O	T1A	Clock	43	9	7
G4	I/O	SO		32	2	20
G5	I/O	SK		33	3	1
G6	I	SI		34	4	2
G7	I	CKO		35	5	3
C0	I/O	MIWU		37		
C1	I/O	MIWU		38		
C2	I/O	MIWU		39		
C3	I/O	MIWU		40		
C4	I/O	MIWU		16		
C5	I/O	MIWU		17		
C6	I/O	MIWU		18		
C7	I/O	MIWU		19		
F0	I/O			4	14	
F1	I/O			5	15	
F2	I/O			6	16	
F3	I/O			7	17	
F4	I/O			8		
F5	I/O			9		
F6	I/O			10		
F7	I/O			11		
J0	I/O			24	26	
J1	I/O			25	27	
J2	I/O			26	28	
J3	I/O			27	1	
J4	I/O			28		
J5	I/O			29		
J6	I/O			30		
J7	I/O			31		
V <sub>CC</sub>			V <sub>CC</sub>	42	8	6
GND			GND	41	7	5
CKI	I			36	6	4
RESET	I		RESET	3	13	11

a. G1 operation as WDOUT is controlled by Option Register, bit 2.

## 6.0 Architectural Overview

### 6.1 EMI REDUCTION

The COP8TAB9/TAC9 devices incorporate circuitry that guards against electromagnetic interference - an increasing problem in today's microcontroller board designs. National's patented EMI reduction technology offers low EMI clock circuitry, gradual turn-on output drivers (GTOs) and internal Icc smoothing filters, to help circumvent many of the EMI issues influencing embedded control designs. National has achieved 15 dB–20 dB reduction in EMI transmissions when designs have incorporated its patented EMI reducing circuitry.

### 6.2 IN-SYSTEM PROGRAMMING

The devices include a program in a boot ROM that provides the capability, through the MICROWIRE/PLUS serial interface, to erase, program and read the contents of the Flash memory.

Additional routines are included in the boot ROM, which can be called by the user program, to enable the user to customize in-system software update capability if MICROWIRE/PLUS is not desired.

The contents of the boot ROM have been defined by National. Execution of code from the boot ROM is dependent on the state of the FLEX bit in the Option Register on exit from RESET. If the FLEX bit is a zero, the Flash Memory is assumed to be empty and execution from the boot ROM begins.

### 6.3 TRUE IN-SYSTEM EMULATION

On-chip emulation capability has been added which allows the user to perform true in-system emulation using final production boards and devices. This simplifies testing and evaluation of software in real environmental conditions. The user, merely by providing for a standard connector which can be bypassed by jumpers on the final application board, can provide for software and hardware debugging using actual production units.

### 6.4 ARCHITECTURE

The COP8 family is based on a modified Harvard architecture, which allows data tables to be accessed directly from program memory. This is very important with modern microcontroller-based applications, since program memory is usually ROM or EPROM, while data memory is usually RAM. Consequently constant data tables need to be contained in non-volatile memory, so they are not lost when the microcontroller is powered down. In a modified Harvard architecture, instruction fetch and memory data transfers can be overlapped with a two stage pipeline, which allows the next instruction to be fetched from program memory while the current instruction is being executed using data memory. This is not possible with a Von Neumann single-address bus architecture.

The COP8 family supports a software stack scheme that allows the user to incorporate many subroutine calls. This capability is important when using High Level Languages. With a hardware stack, the user is limited to a small fixed number of stack levels.

### 6.5 INSTRUCTION SET

In today's 8-bit microcontroller application arena cost/performance, flexibility and time to market are several of the key issues that system designers face in attempting to build

well-engineered products that compete in the marketplace. Many of these issues can be addressed through the manner in which a microcontroller's instruction set handles processing tasks. And that's why the COP8 family offers a unique and code-efficient instruction set - one that provides the flexibility, functionality, reduced costs and faster time to market that today's microcontroller based products require.

Code efficiency is important because it enables designers to pack more on-chip functionality into less program memory space (ROM, OTP or Flash). Selecting a microcontroller with less program memory size translates into lower system costs, and the added security of knowing that more code can be packed into the available program memory space.

#### 6.5.1 Key Instruction Set Features

The COP8 family incorporates a unique combination of instruction set features, which provide designers with optimum code efficiency and program memory utilization.

#### 6.5.2 Single Byte/Single Cycle Code Execution

The efficiency is due to the fact that the majority of instructions are of the single byte variety, resulting in minimum program space. Because compact code does not occupy a substantial amount of program memory space, designers can integrate additional features and functionality into the microcontroller program memory space. Also, the majority instructions executed by the device are single cycle, resulting in minimum program execution time. In fact, 77% of the instructions are single byte single cycle, providing greater code and I/O efficiency, and faster code execution.

#### 6.5.3 Many Single-Byte, Multi-Function Instructions

The COP8 instruction set utilizes many single-byte, multi-function instructions. This enables a single instruction to accomplish multiple functions, such as DRSZ, DCOR, JID, LD (Load) and X (Exchange) instructions with post-incrementing and post-decrementing, to name just a few examples. In many cases, the instruction set can simultaneously execute as many as three functions with the same single-byte instruction.

JID: (Jump Indirect); Single byte instruction decodes external events and jumps to corresponding service routines (analogous to "DO CASE" statements in higher level languages).

LAID: (Load Accumulator-Indirect); Single byte look up table instruction provides efficient data path from the program memory to the CPU. This instruction can be used for table lookup and to read the entire program memory for checksum calculations.

RETSK: (Return Skip); Single byte instruction allows return from subroutine and skips next instruction. Decision to branch can be made in the subroutine itself, saving code.

AUTOINC/DEC: (Auto-Increment/Auto-Decrement); These instructions use the two memory pointers B and X to efficiently process a block of data (simplifying "FOR NEXT" or other loop structures in higher level languages).

#### 6.5.4 Bit-Level Control

Bit-level control over many of the microcontroller's I/O ports provides a flexible means to ease layout concerns and save board space. All members of the COP8 family provide the ability to set, reset and test any individual bit in the data memory address space, including memory-mapped I/O ports and associated registers.



## 6.0 Architectural Overview (Continued)

### 6.5.5 Register Set

Three memory-mapped pointers handle register indirect addressing and software stack pointer functions. The memory data pointers allow the option of post-incrementing or post-decrementing with the data movement instructions (LOAD/EXCHANGE). Fifteen (15) memory-mapped registers allow designers to optimize the precise implementation of certain specific instructions.

### 6.6 PACKAGING/PIN EFFICIENCY

Real estate and board configuration considerations demand maximum space and pin efficiency, particularly given today's high integration and small product form factors. Microcon-

troller users try to avoid using large packages to get the I/O needed. Large packages take valuable board space and increase device cost, two trade-offs that microcontroller designs can ill afford.

The COP8 family offers a wide range of packages to minimize the need for unused pins.

## 7.0 Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage ( $V_{CC}$ )	3.5V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Protection Level (Human Body Model)	2 kV
(Machine Model)	200V

Total Current into $V_{CC}$ Pin (Source)	80 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	$-65^{\circ}\text{C}$ to $+140^{\circ}\text{C}$

**Note 1:** Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

## 8.0 Electrical Characteristics

**DC Electrical Characteristics  $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$  unless otherwise specified.**

Datasheet min/max specification limits are guaranteed by design, test, or statistical analysis.

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.25		2.75	V
Power Supply Rise Time from 0.0V (On-Chip Power-On Reset Selected)		20 $\mu\text{s}$		10 ms	
Power Supply Ripple (Note 3)	Peak-to-Peak			0.1 $V_{CC}$	V
Supply Current (Note 4)					
CKI = 15 MHz	$V_{CC} = 2.75V$ , $t_C = 0.65\mu\text{s}$			6	mA
CKI = 5MHz	$V_{CC} = 2.75V$ , $t_C = 2.0 \mu\text{s}$			3	mA
HALT Current (Note 5) — WATCHDOG Disabled	$V_{CC} = 2.75V$ , CKI = 0 MHz $T_A = 25^{\circ}\text{C}$ $T_A = 85^{\circ}\text{C}$		2	15 100	$\mu\text{A}$ $\mu\text{A}$
IDLE Current (Note 4)					
CKI = 15 MHz	$V_{CC} = 2.75V$ , $t_C = 0.65\mu\text{s}$			1	mA
CKI = 5MHz	$V_{CC} = 2.75V$ , $t_C = 2.0 \mu\text{s}$			0.8	mA
Input Levels ( $V_{IH}$ , $V_{IL}$ )					
Logic High L0 (SDA), L1 (SCL) and L2	1.8V compatibility option selected and ACCESS.Bus is enabled	1.4			V
All Other Inputs		0.8 $V_{CC}$			V
Logic Low				0.25 $V_{CC}$	V
Value of the Internal Bias Resistor for the Crystal/Resonator Oscillator		0.3	1.0	2.5	M $\Omega$
Hi-Z Input Leakage (same as TRI-STATE output)	$V_{CC} = 2.75V$	-0.1		+0.1	$\mu\text{A}$
Input Pullup Current	$V_{CC} = 2.75V$ , $V_{IN} = 0V$	-15		-120	$\mu\text{A}$
Port Input Hysteresis		0.1			V
Output Current Levels					
Source (Weak Pull-Up)	$V_{CC} = 2.25V$ , $V_{OH} = 1.7V$	-10		-80	$\mu\text{A}$
Source (Push-Pull Mode)	$V_{CC} = 2.25V$ , $V_{OH} = 1.7V$	-10			mA
Sink (Push-Pull Mode)	$V_{CC} = 2.25V$ , $V_{OL} = 0.4V$	10			mA
Allowable Sink & Source Current per Pin				16	mA
Maximum Input Current without Latchup				$\pm 200$	mA
RAM Retention Voltage, $V_r$				TBD	V
Input Capacitance				8.5	pF
Flash Endurance		20K			Erase/Write Cycles
Flash Data Retention	$25^{\circ}\text{C}$			100	years

## 8.0 Electrical Characteristics (Continued)

**AC Electrical Characteristics**  $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$  unless otherwise specified.

Datasheet min/max specification limits are guaranteed by design, test, or statistical analysis.

Parameter	Conditions	Min	Typ	Max	Units
Oscillator Frequency					
Crystal/Resonator, External	$2.25\text{V} \leq V_{CC} \leq 2.75\text{V}$		9.0	15	MHz
Internal R/C Oscillator	$2.25\text{V} \leq V_{CC} \leq 2.75\text{V}$				MHz
R/C Oscillator Frequency Variation	$2.25\text{V} \leq V_{CC} \leq 2.75\text{V}$			$\pm 30$	%
Instruction Cycle Time ( $t_C$ )					
Crystal/Resonator, External	$2.25\text{V} \leq V_{CC} \leq 2.75\text{V}$	0.65	1.1	DC	$\mu\text{s}$
Internal R/C Oscillator	$2.25\text{V} \leq V_{CC} \leq 2.75\text{V}$				$\mu\text{s}$
External CKI Clock Duty Cycle	fr = Max	45		55	%
Rise Time	fr = 10 MHz Ext Clock			12	ns
Fall Time	fr = 10 MHz Ext Clock			8	ns
MICROWIRE Setup Time ( $t_{UWS}$ ) <i>Figure 1</i>		20			ns
MICROWIRE Hold Time ( $t_{UWH}$ ) <i>Figure 1</i>		20			ns
MICROWIRE Output Propagation Delay ( $t_{UPD}$ ) <i>Figure 1</i>				150	ns
MICROWIRE Maximum Shift Clock					
Master Mode				750	kHz
Slave Mode				1.5	MHz
Input Pulse Width					
Interrupt Input High Time (Note 2)		1			$t_C$
Interrupt Input Low Time		1			$t_C$
Timer Input High Time		1			$t_C$
Timer Input Low Time		1			$t_C$
Reset Pulse Width		0.5			$\mu\text{s}$
Mass Erase Time		200			ms
Page Erase Time		20			ms
ACCESS.Bus Input signals (Note 6)					
Bus Free Time Between Stop and Start Condition ( $t_{BUF_i}$ ) <i>Figure 2</i>		$t_{SCLhigho}$			
SCL Setup Time ( $t_{CSTOS_i}$ ) <i>Figure 2</i>	Before Stop Condition	8			mclk
SCL Hold Time ( $t_{CSTRh_i}$ ) <i>Figure 2</i>	After Start Condition	8			mclk
SCL Setup Time ( $t_{CSTRs_i}$ ) <i>Figure 3</i>	Before Start Condition	8			mclk
Data High Setup Time ( $t_{DHCs_i}$ ) <i>Figure 3</i>	Before SCL Rising Edge (RE)	2			mclk
Data Low Setup Time ( $t_{DLCs_i}$ ) <i>Figure 2</i>	Before SCL RE	2			mclk
SCL Low Time ( $t_{SCLlow_i}$ ) <i>Figure 4</i>	After SCL Falling Edge (FE)	12			mclk
SCL High Time ( $t_{SCLhigh_i}$ ) <i>Figure 4</i>	After SCL RE	12			mclk
SDA Hold Time ( $t_{SDAh_i}$ ) <i>Figure 4</i>	After SCL FE	0			ns
SDA Setup Time ( $t_{SDAs_i}$ ) <i>Figure 4</i>	Before SCL RE	2			mclk
ACCESS.Bus Output Signals (Note 6)					
Bus Free Time Between Stop and Start Condition ( $t_{BUF_o}$ ) <i>Figure 2</i>		$t_{SCLhigho}$			
SCL Setup Time ( $t_{CSTOs_o}$ ) <i>Figure 2</i>	Before Stop Condition	$t_{SCLhigho}$			
SCL Hold Time ( $t_{CSTRh_o}$ ) <i>Figure 3</i>	After Start Condition	$t_{SCLhigho}$			
SCL Setup Time ( $t_{CSTRs_o}$ ) <i>Figure 3</i>	Before Start Condition	$t_{SCLhigho}$			
Data High Setup Time ( $t_{DHCs_o}$ ) <i>Figure 3</i>	Before SCL RE	$t_{SCLhigho}$			
Data Low Setup Time ( $t_{DLCs_o}$ ) <i>Figure 2</i>	Before SCL RE	$t_{SCLhigho}$			
SCL Low Time ( $t_{SCLlow_o}$ ) <i>Figure 4</i>	After SCL FE	16			mclk
SCL High Time ( $t_{SCLhigh_o}$ ) <i>Figure 4</i>	After SCL RE	16			mclk

## 8.0 Electrical Characteristics (Continued)

### AC Electrical Characteristics $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ unless otherwise specified. (Continued)

Datasheet min/max specification limits are guaranteed by design, test, or statistical analysis.

Parameter	Conditions	Min	Typ	Max	Units
SDA Hold Time ( $t_{SDAho}$ ) <i>Figure 4</i>	After SCL FE	7			mclk
SDA Valid Time ( $t_{SDAso}$ ) <i>Figure 4</i>	Before SCL RE	7			mclk

**Note 2:**  $t_C$  = Instruction cycle time (Clock input frequency divided by 10).

**Note 3:** Maximum rate of voltage change must be  $< 0.5 \text{ V/ms}$ .

**Note 4:** Supply and IDLE currents are measured with CKI driven with a square wave Oscillator, inputs connected to  $V_{CC}$  and outputs driven low but not connected to a load.

**Note 5:** The HALT mode will stop CKI from oscillating in the R/C and the Crystal configurations. CKI is TRI-STATE. Measurement of  $I_{DD}$  HALT is done with device neither sourcing nor sinking current; with L, F, C, G0, and G2–G5 programmed as low outputs and not driving a load; all outputs programmed low and not driving a load; all inputs tied to  $V_{CC}$ ; WATCHDOG and clock monitor disabled. Parameter refers to HALT mode entered via setting bit 7 of the G Port data register.

**Note 6:** The ACCESS.Bus interface of the COP8TAB9/TAC9 device implements and meets the timings necessary for interface to the I<sup>2</sup>C and SMBus protocols at logic levels. The bus drivers are designed with open-drain outputs, as required for proper bidirectional operation. The device will not meet the AC timing and current/voltage drive requirements of the full bus specifications.

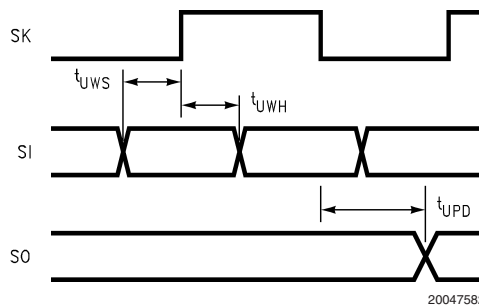
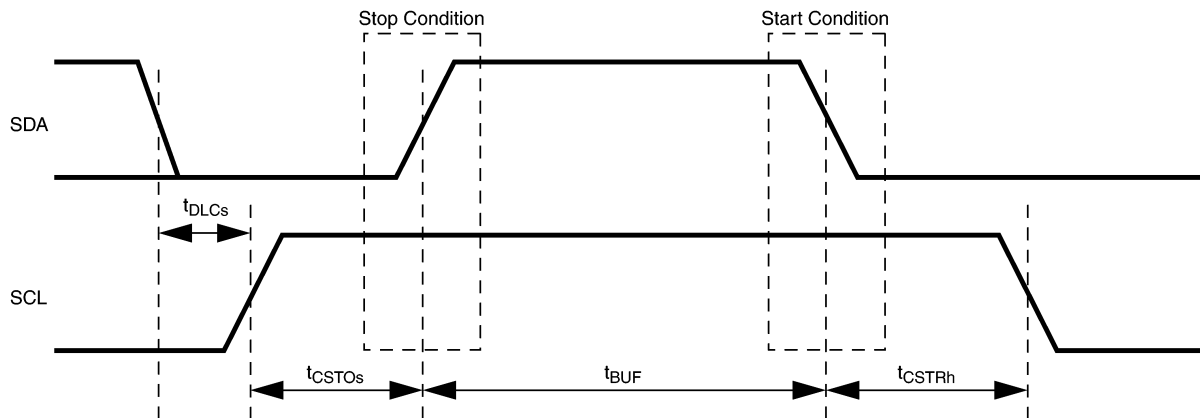


FIGURE 1. MICROWIRE/PLUS Timing

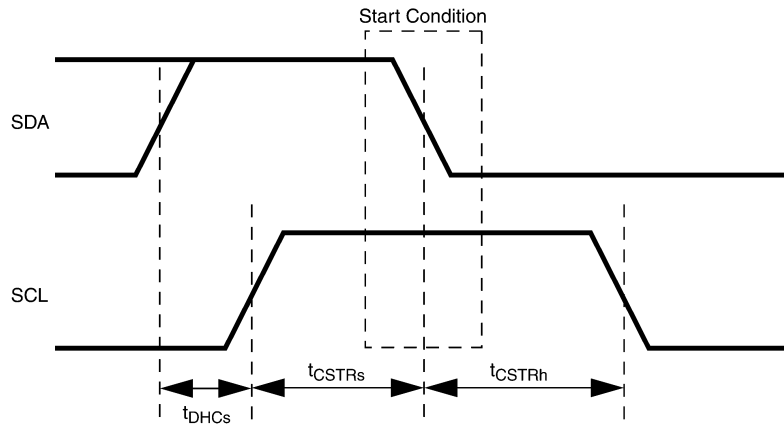


Note: In the timing tables the parameter name is appended with an "o" for output signal timing and "i" for input signal timing.

20047583

FIGURE 2. ACB Start and Stop Condition Timing

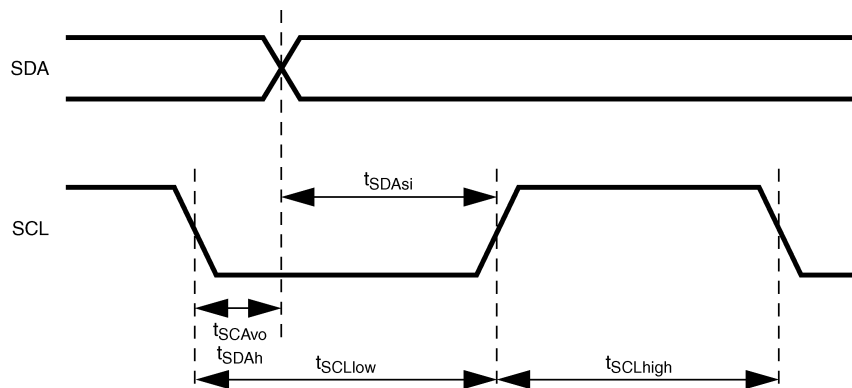
## 8.0 Electrical Characteristics (Continued)



Note: In the timing tables the parameter name is appended with an "o" for output signal timing and "i" for input signal timing.

20047584

FIGURE 3. ACB Start Condition Timing



Note: In the timing tables the parameter name is appended with an "o" for output signal timing and "i" for input signal timing, unless the parameter already includes the suffix.

20047585

FIGURE 4. ACB Data Timing

## 9.0 Pin Descriptions

The COP8TAB9/TAC9 I/O structure enables designers to reconfigure the microcontroller's I/O functions with a single instruction. Each individual I/O pin can be independently configured as output pin low, output high, input with high impedance or input with weak pull-up device. A typical example is the use of I/O pins as the keyboard matrix input lines. The input lines can be programmed with internal weak pull-ups so that the input lines read logic high when the keys are all open. With a key closure, the corresponding input line will read a logic zero since the weak pull-up can easily be overdriven. When the key is released, the internal weak pull-up will pull the input line back to logic high. This eliminates the need for external pull-up resistors. The high current options are available for driving LEDs, motors and

speakers. This flexibility helps to ensure a cleaner design, with fewer external components and lower costs. Below is the general description of all available pins.

$V_{CC}$  and GND are the power supply pins.

Users of the LLP package are cautioned to be aware that the central metal area and the pin 1 index mark on the bottom of the package may be internally connected to GND. See figure below:

## 9.0 Pin Descriptions (Continued)

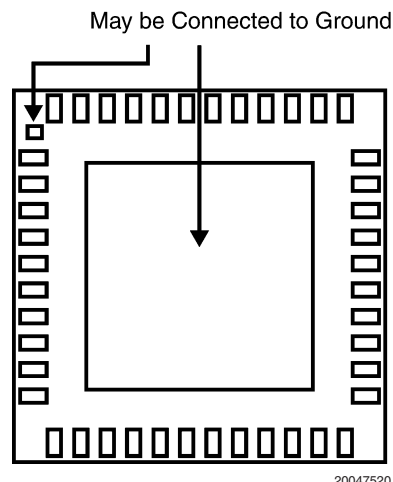


FIGURE 5. LLP Package Bottom View

CKI is the clock input. This pin can be connected (in conjunction with CKO) to an external crystal circuit to form a crystal oscillator, to an external resistor for RC oscillator operation or to an external clock. See Oscillator Description section.

**RESET** is the master reset input. See Reset description section.

The device contains up to five bidirectional 8-bit I/O ports (C, F, G, J and L), where each individual bit may be independently configured as an input (Schmitt trigger inputs on all ports), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has three associated 8-bit memory mapped registers, the CONFIGURATION register, the output DATA register and the Pin input register. (See the memory map for the various addresses associated with the I/O ports.) Figure 6 shows the I/O port configurations. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Port C supports the Multi-Input Wake-Up feature on all eight pins. Port C is not available on 20 and 28 pin packages. When using these packages, the user should ensure that Port C Multi-Input Wake-Up is disabled by clearing the CWKEN Register to prevent spurious interrupts and wake-up events. Port C has the following alternate pin functions:

- C7 Multi-Input Wake-Up
- C6 Multi-Input Wake-Up
- C5 Multi-Input Wake-Up
- C4 Multi-Input Wake-Up
- C3 Multi-Input Wake-Up

- C2 Multi-Input Wake-Up

- C1 Multi-Input Wake-Up

- C0 Multi-Input Wake-Up

Port F is an 8-bit I/O port. All F pins have Schmitt triggers on the inputs.

Port G is an 8-bit port. Pin G0, G2–G5 are bi-directional I/O ports. Pin G6 is always a general purpose Hi-Z input. All G pins have Schmitt Triggers on their inputs. **Pin G1 serves as the dedicated WATCHDOG output with weak pull-up if the WATCHDOG feature is selected by the Option register. The pin is a general purpose I/O, if WATCHDOG feature is not selected.** If WATCHDOG feature is selected, bit 1 of the Port G configuration and data register does not have any effect on Pin G1 setup. Pin G7 is either input or output depending on the oscillator option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the internal R/C or the external oscillator option selected, G7 serves as a general purpose Hi-Z input pin and is also used to bring the device out of HALT mode with a low to high transition on G7. Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C or external clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

The device will be placed in the HALT mode by writing a “1” to bit 7 of the Port G Data Register. Similarly the device will be placed in the IDLE mode by writing a “1” to bit 6 of the Port G Data Register.

Writing a “1” to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config. Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G7 CKO Oscillator dedicated output or general purpose input.

G6 SI (MICROWIRE/PLUS Serial Data Input)

G5 SK (MICROWIRE/PLUS Serial Clock)

G4 SO (MICROWIRE/PLUS Serial Data Output)

G3 T1A (Timer T1 I/O)

G2 T1B (Timer T1 Capture Input)

G1 WDOUT WATCHDOG and/or Clock Monitor if WATCHDOG enabled, otherwise it is a general purpose I/O

G0 INTR (External Interrupt Input)

G0 through G3 are also used for In-System Emulation.

Port J is an 8-bit I/O port. All J pins have Schmitt triggers on the inputs. At RESET, Port J outputs are enabled and are forced to the High state.

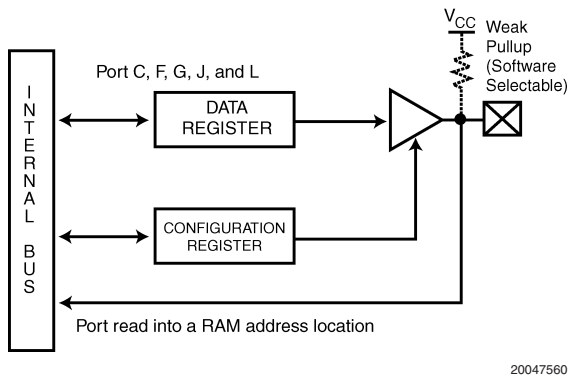
Port L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Pins L0 (SDA), L1 (SCL) and L2 inputs provide compatibility with 1.8V logic levels when LVCMP (Option Register bit 7) is set and the ACCESS.Bus is enabled.

Port L supports the Multi-Input Wake-Up feature on all eight pins. Port L has the following alternate pin functions:

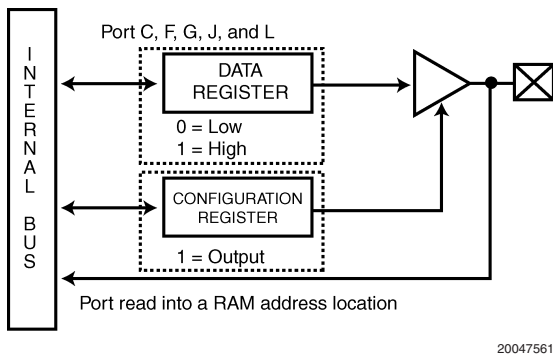
## 9.0 Pin Descriptions (Continued)

- L7 Multi-Input Wake-Up
- L6 Multi-Input Wake-Up
- L5 Multi-Input Wake-Up
- L4 Multi-Input Wake-Up
- L3 Multi-Input Wake-Up
- L2 Multi-Input Wake-Up (optional 1.8V compatible input)
- L1 Multi-Input Wake-Up or ACCESS.Bus Serial Clock (optional 1.8V compatible input)
- L0 Multi-Input Wake-Up or ACCESS.Bus Serial Data (optional 1.8V compatible input)



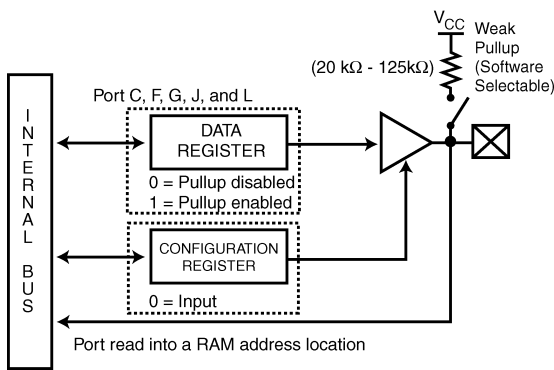
20047560

FIGURE 6. I/O Port Configurations



20047561

FIGURE 7. I/O Port Configurations—Output Mode

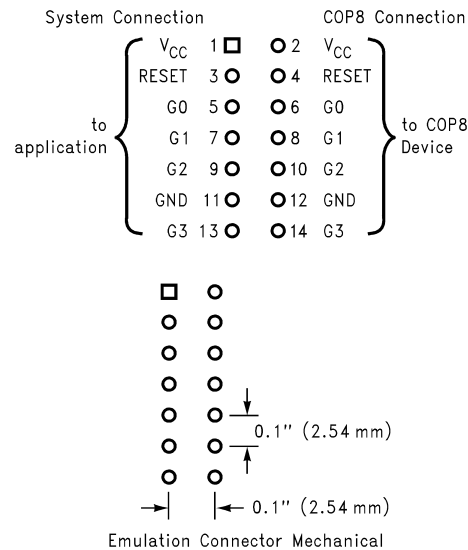


20047562

FIGURE 8. I/O Port Configurations—Input Mode

## 9.1 EMULATION CONNECTION

Connection to the emulation system is made via a 2 x 7 connector which interrupts the continuity of the RESET, G0, G1, G2 and G3 signals between the COP8 device and the rest of the target system (as shown in Figure 9). This connector can be designed into the production pc board and can be replaced by jumpers or signal traces when emulation is no longer necessary. The emulator will replicate all functions of G0 - G3 and RESET. For proper operation, no connection should be made on the device side of the emulator connector.



Emulation Connector Mechanical

20047509

FIGURE 9. Emulation Connection

## 10.0 Functional Description

The architecture of the device is a modified Harvard architecture. With the Harvard architecture, the program memory (Flash) is separate from the data store memory (RAM). Both Program Memory and Data Memory have their own separate addressing space with separate address buses. The architecture, though based on the Harvard architecture, permits transfer of data from Flash Memory to RAM.

### 10.1 CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction ( $t_c$ ) cycle time.

## 10.0 Functional Description

(Continued)

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). With reset, the SP is initialized to RAM address 06F Hex. The SP is decremented as items are pushed onto the stack. SP points to the next available location on the stack.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

**TABLE 1. Available Memory Address Ranges**

Device	Program Memory Size (Flash)(Bytes)	Flash Memory Page Size (Bytes)	Option Register Address (Hex)	Data Memory Size (RAM) (Bytes)	RAM Segments Available	Maximum RAM Address (HEX)
COP8TAB9	2048	512	0x07FF (hex)	128	Segment 0	06F
COP8TAC9	4096		0x0FFF (hex)			

### 10.3 DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, ACCESS.Bus Interface and the various registers and counters associated with the timer, T1. Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The data memory consists of 128 bytes of RAM. Sixteen bytes of RAM are mapped as “registers” at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers being available for general usage.

The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Note: RAM contents are undefined upon power-up.

### 10.4 OPTION REGISTER

The Option register, located at address 0x0FFF (hex) or 0x07FF (hex) in the Flash Program Memory, is used to configure the user selectable security, WATCHDOG, HALT and Oscillator selection options. The register can be programmed only in external Flash Memory programming or ISP Programming modes. Therefore, the register must be programmed at the same time as the program memory. The contents of the Option register shipped from the factory read 00 Hex.

### 10.2 PROGRAM MEMORY

The program memory consists of 4096 bytes of Flash Memory. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the device vector to program memory location 00FF Hex. The program memory reads 00 Hex in the erased state. Program execution starts at location 0 after RESET.

If a Return instruction is executed when the SP contains 6F (hex), instruction execution will continue from Program Memory location 7FFF (hex). If location 7FFF is accessed by an instruction fetch, the Flash Memory will return a value of 00. This is the opcode for the INTR instruction and will cause a Software Trap.

For the purpose of erasing and rewriting the Flash Memory, it is organized in pages of 512 bytes as shown in Table 1.

The format of the Option register is as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LVCMP	CLKSEL2	SEC	CLKSEL1	CLKSEL0	WATCH DOG	HALT	FLEX

Bit 7 When this bit is set and the ACCESS.Bus is enabled, inputs L0, L1 and L2, are compatible with 1.8V logic levels.

Bit 6 This bit defines the most significant bit of the oscillator selection. (See Section 10.7 OSCILLATOR CIRCUITS) for more information on Oscillator selection.)

Bit 5  
= 1 Security enabled. Flash Memory read and write are not allowed except in User ISP/Virtual E<sup>2</sup> commands. Mass Erase is allowed.

= 0 Security disabled. Flash Memory read and write are allowed.

Bits 4, 3 These bits define the two least significant bits of the oscillator selection.

Bit 2  
= 1 WATCHDOG feature disabled. G1 is a general purpose I/O.

= 0 WATCHDOG feature enabled. G1 pin is WATCHDOG output with weak pullup.

Bit 1  
= 1 HALT mode disabled.

= 0 HALT mode enabled.



## 10.0 Functional Description

(Continued)

Bit 0

- = 1 Execution following RESET will be from Flash Memory.
- = 0 Flash Memory is erased. Execution following RESET will be from Boot ROM with the MICROWIRE/PLUS ISP routines.

The COP8 assembler defines a special ROM section type, CONF, into which the Option Register data may be coded. The Option Register is programmed automatically by programmers that are certified by National.

The user needs to ensure that the FLEX bit will be set when the device is programmed.

The following examples illustrate the declaration of the Option Register.

Syntax:

```
[label:].sect    config, conf
               .db    value        ;1 byte,
                                   ;configures
                                   ;options

               .endsect
```

Example: The following sets a value in the Option Register and User Identification for a COP8TAC9HLQ7. The Option Register bit values shown select options: Security disabled, WATCHDOG enabled HALT mode enabled and execution will commence from Flash Memory.

```
.chip          8TAC
.sect          option, conf
.db            0x01          ;wd, halt, flex
.endsect
...
.end           start
```

Note: All programmers certified for programming this family of devices will support programming of the Option Register. Please contact National or your device programmer supplier for more information.

### 10.5 SECURITY

The device has a security feature which, when enabled, prevents external reading of the Flash program memory. The security bit in the Option Register determines, whether security is enabled or disabled. If the security feature is disabled, the contents of the internal Flash Memory may be read by external programmers or by the built in MICROWIRE/PLUS serial interface ISP. **Security must be enforced by the user when the contents of the Flash Memory are accessed via the user ISP.**

If the security feature is enabled, then any attempt to externally read the contents of the Flash Memory will result in the value FF (hex) being read from all program locations (except the Option Register). In addition, with the security feature enabled, the write operation to the Flash program memory and Option Register is inhibited. Page Erases are also inhibited when the security feature is enabled. The Option Register is readable regardless of the state of the security bit by accessing location FFFF (hex). Mass Erase Operations are possible regardless of the state of the security bit.

The security bit can be erased only by a Mass Erase of the entire contents of the Flash unless Flash operation is under the control of User ISP functions.

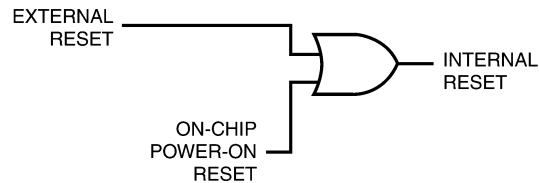
Note: The actual memory address of the Option Register is dependent on the Flash Memory size and is defined in *Table*

1, however the MICROWIRE/PLUS ISP routines require the address FFFF (hex) to be used to read the Option Register when the Flash Memory is secured.

The entire Option Register must be programmed at one time and cannot be rewritten without first erasing the entire last page of Flash Memory.

### 10.6 RESET

The device is initialized when the  $\overline{\text{RESET}}$  pin is pulled low or the On-chip Power-On Reset is activated.



20047521

FIGURE 10. Reset Logic

The on-chip Power-On Reset circuit is provided for compatibility to the ROM based version of this device, but is insufficient to ensure against inadvertent Flash memory corruption. See *Section 10.6.1 External Reset* for the requirements.

Applications which are designed to use either the COP8TAB9/TAC9 or the COP8TAB5/TAC5 must meet the Reset requirements of the Flash device.

The following occurs upon initialization:

**Port C:** TRI-STATE (High Impedance Input)

**Port F:** TRI-STATE (High Impedance Input)

**Port G:** TRI-STATE (High Impedance Input). Exceptions: If Watchdog is enabled, then G1 is Watchdog output. G0 and G2 have their weak pull-up enabled during RESET.

**Port J:** Output High

**Port L:** TRI-STATE (High Impedance Input)

**PC:** CLEARED to 0000

**PSW, CNTRL and ICNTRL registers:** CLEARED

**SIOR:**

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

**ITMR, CLKPS:** Cleared

**Accumulator and Timer 1:**

RANDOM after RESET

**CWKEN, CWKEDG, LWKEN, LWKEDG:** CLEARED

**CWKPNL, LWKPNL:** RANDOM

**SP (Stack Pointer):**

Initialized to RAM address 06F Hex

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

**RAM:**

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

**ISP CONTROL:**

**ISPADLO:** CLEARED

**ISPADHI:** CLEARED

**PGMTIM:** PRESET TO VALUE FOR 10 MHz CKI

## 10.0 Functional Description

(Continued)

### WATCHDOG (if enabled):

The device comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of 64k T<sub>0</sub> clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low output on pin G1. This error output will continue until 16–32 T<sub>0</sub> clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will go high.

### 10.6.1 External Reset

The  $\overline{\text{RESET}}$  input, when pulled low, initializes the device. The  $\overline{\text{RESET}}$  pin must be held low for a minimum of one instruction cycle to guarantee a valid reset. Reset should be long enough to ensure crystal start-up upon Power-Up, if the Crystal Oscillator option has been selected.

$\overline{\text{RESET}}$  may also be used to cause an exit from the HALT mode.

**Any controller which features the capability for the software to modify the contents of the Flash memory is susceptible to inadvertent writes or erases any time  $V_{CC}$  is below the minimum guaranteed operating conditions, the clock is running and the device is not held in Reset. Flash memory is subject to corruption if the Reset sequence shown in Figure 11 is not applied to the  $\overline{\text{RESET}}$  input. For this reason, the use of external brownout detection is strongly recommended. Any Reset circuit must ensure that RESET is set high only after  $V_{CC}$  reaches  $V_{CC(\text{min})}$  and RESET is set low immediately when  $V_{CC}$  falls to  $V_{CC(\text{min})}$  as specified in the DC Electrical Characteristics.**

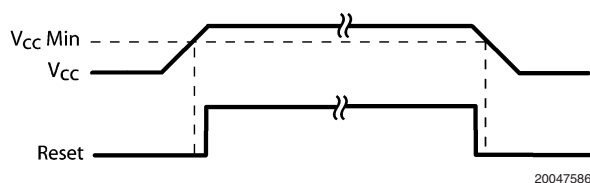
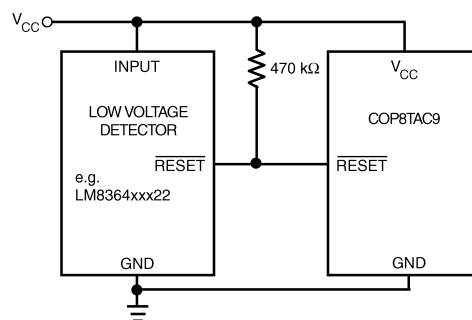


FIGURE 11. Reset vs Vcc

A recommended reset circuit for this device is shown in Figure 12.



20047522

FIGURE 12. Reset Circuit Using External Reset

### 10.6.2 On-Chip Power-On Reset

The device generates an internal reset as  $V_{CC}$  rises to a voltage level above 2.0V. The on-chip reset circuitry is able to detect both fast and slow rise times on  $V_{CC}$  ( $V_{CC}$  rise time between 20  $\mu$ s and 10 ms).

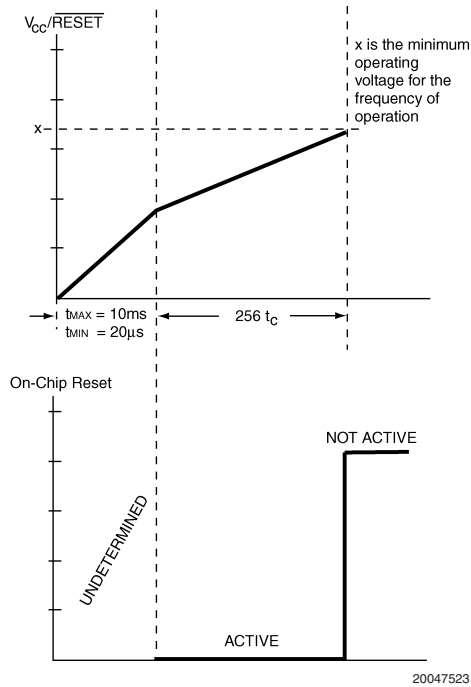
Under no circumstances should the  $\overline{\text{RESET}}$  pin be allowed to float. If the on-chip Power-On Reset feature is being used, the  $\overline{\text{RESET}}$  pin should be connected to  $V_{CC}$ , either directly or through a pull-up resistor. If forced operation from the Boot ROM is anticipated, a pull-up resistor should be used so that the ISP circuit can override the RESET circuit and force the  $\overline{\text{RESET}}$  pin low. The output of the power-on reset detector will **always** preset the Idle timer to 00FF(256 t<sub>C</sub>). At this time, the internal reset will be generated.

The internal reset will not be turned off until the Idle timer underflows. The internal reset will perform the same functions as external reset. The user is responsible for ensuring that  $V_{CC}$  is at the minimum level for operating within the 256 t<sub>C</sub>. After the underflow, the logic is designed such that the Power On Reset circuit will generate no additional internal resets as long as  $V_{CC}$  remains above 2.0V.

The contents of data registers and RAM are unknown following the on-chip reset.

## 10.0 Functional Description

(Continued)



**FIGURE 13. Reset Timing (Power-On Reset Enabled) with  $V_{CC}$  Tied to  $\overline{RESET}$**

### 10.7 OSCILLATOR CIRCUITS

There are five clock oscillator options available: fully internal R/C Oscillator, R/C Oscillator with external frequency determination resistor, Crystal Oscillator with or without on-chip bias resistor and External Oscillator. The oscillator feature is selected by programming the Option Byte, which is summarized in *Table 2*.

**TABLE 2. Oscillator Option**

Option Bit 6	Option Bit 4	Option Bit 3	Oscillator Option
0	0	0	On-Chip R/C Oscillator
0	0	1	R/C Oscillator with External Resistor
0	1	0	Crystal Oscillator with Internal Bias Resistor
0	1	1	Crystal Oscillator with External Bias Resistor
1	x	x	External Oscillator

#### 10.7.1 R/C Oscillator

The device features an R/C oscillator with two modes of operation:

1. R/C with internal R operating at a fixed frequency (R/C mode)
2. R/C with external frequency control resistor (R/C+R mode).

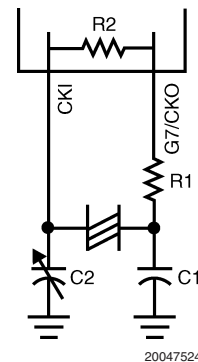
If the Oscillator Selection bits of the Option Byte remain unprogrammed (equal to zero), the internal R/C Oscillator mode will be selected. In internal R/C oscillation mode, CKI may be left unconnected, while G7/CKO is available as a general purpose input G7 and/or HALT control. The internal R/C oscillator has on-chip resistor and capacitor for predetermined R/C oscillator frequency operation. The predetermined frequency is 10 MHz  $\pm$  20% for temperature range of  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ .

The R/C Oscillator with external frequency control resistor mode (R/C+R) can be selected by programming Option Bit 3 to 1 and Option Bits 6 and 4 to 0. In R/C+R mode, the frequency of oscillation is controlled by the current through a resistor connected from the CKI pin to GND. G7/CKO is available as a general purpose input G7 and/or HALT control. The maximum frequency is 10 MHz  $\pm$  20% for temperature range of  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ . For lower frequencies, an external resistor should be connected between CKI and GND. PC board trace length on the CKI pin should be kept as short as possible.

*Table 3* shows the oscillator frequency as a function of approximate external resistance on the CKI pin. *Figure 17* shows the R/C oscillator configuration.

**TABLE 3. R/C+R Oscillator Configuration,  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ,  
OSC Freq. Variation of  $\pm$  20%**

External Resistor (k $\Omega$ )	OSC Freq (MHz)	Instr. Cycle ( $\mu\text{s}$ )
60	15	0.65
100	10	1.0
200	5	2.0
500	2	5.0



**FIGURE 14. Crystal Oscillator With On-Chip Bias Resistor**

## 10.0 Functional Description

(Continued)

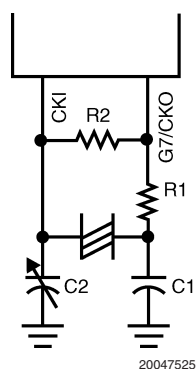
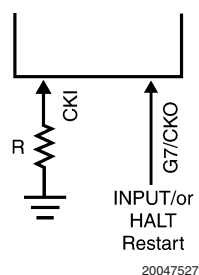


FIGURE 15. Crystal Oscillator With External Bias Resistor

With External Frequency Control Resistor (R/C+R)



With Fully On-Chip R/C Oscillator.

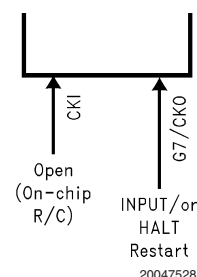


FIGURE 17. R/C Oscillator

### 10.7.2 Crystal Oscillator

The Crystal Oscillator mode can be selected by programming Option Bit 4 to 1. CKI is the clock input while G7/CKO is the clock generator output to the crystal. An on-chip bias resistor connected between CKI and CKO can be enabled by programming Option Bit 3 to 0. The value of the resistor is in the range of 0.3M to 2.5M (typically 1.0M).

Table 4 shows the component values required for various standard crystal values. Resistor R2 is only used when the on-chip bias resistor is disabled. Figure 14 and Figure 15 show the crystal oscillator connection diagrams.

TABLE 4. Crystal Oscillator Configuration,  
 $T_A = 25^\circ\text{C}$ ,  $V_{CC} = 2.5\text{V}$

R1 (k $\Omega$ )	R2 (M $\Omega$ )	C1 (pF)	C2 (pF)	CKI Freq. (MHz)
0	1	18	18	15
0	1	18	18	10
0	1	45	30–36	4
5.6	1	100	100–156	0.455

### 10.7.3 External Oscillator

The External Oscillator mode can be selected by programming Option Bit 3 to 0 and Option Bit 4 to 0. CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels.

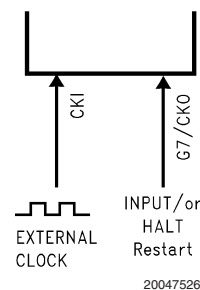


FIGURE 16. External Oscillator

G7/CKO is available as a general purpose input G7 and/or Halt control. Figure 13 shows the external oscillator connection diagram.

### 10.7.4 Clock Prescaler

The device is equipped with a programmable clock prescaler which allows the user to dynamically adjust the clock speed, and thus the power dissipation, to the processing needs of the application. By merely writing an eight-bit value to the CLKPS register, the user can divide the input oscillator clock by an integer multiple (1—256) and reduce the CPU clock frequency. The format of the CLKPS Register is shown in Table 5. The value written to the CLKPS register is one less than the desired divider. A value of 0 (zero) written to the CLKPS register yields a CPU clock equal to the input clock frequency. A value of 255 written to the CLKPS register yields a CPU clock with a period equal to 256 input clock periods.

TABLE 5. Clock Prescale Register (CLKPS)

CLKPS	
Bit 7	Bit 0

## 10.0 Functional Description

(Continued)

### 10.8 CONTROL REGISTERS

#### 10.8.1 CNTRL Register (Address X'00EE)

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7				Bit 0			

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

T1C3	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C1	Timer T1 mode control bit
T1C0	Timer T1 Start/Stop control in timer modes 1 and 2. T1 Underflow Interrupt Pending Flag in timer mode 3
MSEL	Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
SL1 & SL0	Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)

#### 10.8.2 PSW Register (Address X'00EF)

HC	C	T1PND	T1ENA	EXPND	BUSY	EXEN	GIE
Bit 7				Bit 0			

The PSW register contains the following select bits:

HC	Half Carry Flag
C	Carry Flag
T1PND	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
EXPND	External interrupt pending
BUSY	MICROWIRE/PLUS busy shifting flag
EXEN	Enable external interrupt
GIE	Global interrupt enable (enables interrupts)

The Half-Carry flag is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the Carry and Half Carry flags.

#### 10.8.3 ICNTRL Register (Address X'00E8)

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PND	T1ENB
Bit 7				Bit 0			

The ICNTRL register contains the following bits:

LPEN	L/C Port Interrupt Enable (Multi-Input Wake-Up/Interrupt)
T0PND	Timer T0 Interrupt pending
T0EN	Timer T0 Interrupt Enable (Bit 12 toggle)
μWPND	MICROWIRE/PLUS interrupt pending
μWEN	Enable MICROWIRE/PLUS interrupt
T1PND	Timer T1 Interrupt Pending Flag for T1B capture edge

T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

#### 10.8.4 ITMR Register (Address X'00CF)

RSVD	ITSEL2	ITSEL1	ITSEL0
Bit 7		Bit 0	

The ITMR register contains the following bits:

RSVD	These bits are reserved and must be 0.
ITSEL2	Idle Timer period select bit.
ITSEL1	Idle Timer period select bit.
ITSEL0	Idle Timer period select bit.

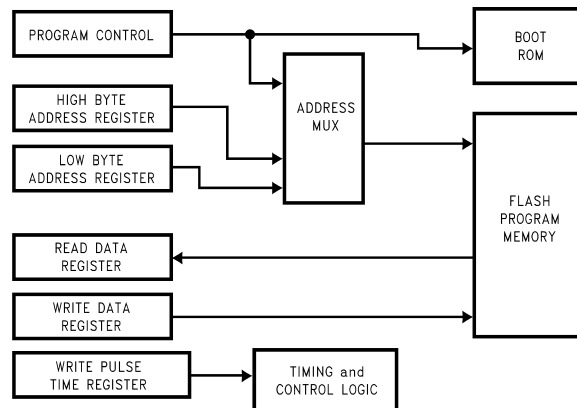
## 11.0 In-System Programming

### 11.1 INTRODUCTION

This device provides the capability to program the program memory while installed in an application board. This feature is called In System Programming (ISP). It provides a means of ISP by using the MICROWIRE/PLUS, or the user can provide his own, customized ISP routine. The factory installed ISP uses the MICROWIRE/PLUS port. The user can provide his own ISP routine that uses any of the capabilities of the device, such as ACCESS.Bus, parallel port, etc. The user is cautioned, however, to remove all calls to Boot ROM functions prior to submission of code for ROM generation and production in COP8TAx5 devices.

### 11.2 FUNCTIONAL DESCRIPTION

The organization of the ISP feature consists of the user Flash program memory, the factory boot ROM, and some registers dedicated to performing the ISP function. See *Figure 18* for a simplified block diagram. The factory installed ISP that uses MICROWIRE/PLUS is located in the Boot ROM. The size of the Boot ROM is 1k bytes and also contains code to facilitate in system emulation capability. If a user chooses to write an application specific ISP routine, it must be located in the Flash program memory.



20047529

FIGURE 18. Block Diagram of ISP

As described in *Section 10.4 OPTION REGISTER*, there is a bit, FLEX, that controls whether the device exits RESET executing from the Flash memory or the Boot ROM. The user must program the FLEX bit as appropriate for the application. In the erased state, the FLEX bit = 0 and the device will power-up executing from Boot ROM. When FLEX = 0, this assumes that either the MICROWIRE/PLUS ISP

## 11.0 In-System Programming

(Continued)

routine or external programming is being used to program the device. If using the MICROWIRE/PLUS ISP routine, the software in the boot ROM will monitor the MICROWIRE/PLUS for commands to program the Flash memory. When programming the Flash program memory is complete, the FLEX bit will have to be programmed to a 1 and the device will have to be reset, either by pulling external Reset to ground or by a MICROWIRE/PLUS ISP EXIT command, before execution from Flash program memory will occur.

If FLEX = 1, upon exiting Reset, the device will begin executing from location 0000 in the Flash program memory. The assumption, here, is that either the application is not using ISP, is using MICROWIRE/PLUS ISP by jumping to it within the application code, or is using a customized ISP routine. If a customized ISP routine is being used, then it must be programmed into the Flash memory by means of the MICROWIRE/PLUS ISP or external programming as described in the preceding paragraph.

### 11.3 REGISTERS

There are six registers required to support ISP: Address Register Hi byte (ISPADHI), Address Register Low byte (ISPADLO), Read Data Register (ISPRD), Write Data Register (ISPWR), Write Timing Register (PGMTIM), and the Control Register (ISPCNTRL). The ISPCNTRL Register is not available to the user. None of these six registers, which support ISP, have been implemented in the COP8TAx5 ROM based devices.

#### 11.3.1 ISP Address Registers

The address registers (ISPADHI & ISPADLO) are used to specify the address of the byte of data being written or read. For page erase operations, the address of the beginning of the page should be loaded. For mass erase operations, 0000 must be placed into the address registers. When reading the Option register, 07FF (hex) should be placed into the address registers of COP8TAB9 devices and 0FFF (hex) should be placed into the address registers of COP8TAC9 devices. Registers ISPADHI and ISPADLO are cleared to 00 on Reset.

Note: The actual memory address of the Option Register is 0x0FFF (hex), however the MICROWIRE/PLUS ISP routines require the address FFFF (hex) to be used to read the Option Register when the Flash Memory is secured.

**TABLE 6. High Byte of ISP Address**

ISPADHI							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Addr 15	Addr 14	Addr 13	Addr 12	Addr 11	Addr 10	Addr 9	Addr 8

**TABLE 7. Low Byte of ISP Address**

ISPADLO							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Addr 7	Addr 6	Addr 5	Addr 4	Addr 3	Addr 2	Addr 1	Addr 0

#### 11.3.2 ISP Read Data Register

The Read Data Register (ISPRD) contains the value read back from a read operation. This register is undefined on Reset.

**TABLE 8. ISP Read Data Register**

ISPRD							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

#### 11.3.3 ISP Write Data Register

The Write Data Register (ISPWR) contains the data to be written into the specified address. This register is undetermined on Reset.

**TABLE 9. ISP Write Data Register**

ISPWR							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

#### 11.3.4 ISP Write Timing Register

The Write Timing Register (PGMTIM) is used to control the width of the timing pulses for write and erase operations. The value to be written into this register is dependent on the frequency of CKI and is shown in *Table 10*. This register must be written before any write or erase operation can take place. It only needs to be loaded once, for each value of CKI frequency. The MICROWIRE/PLUS ISP routine that is resident in the boot ROM requires that this register be defined prior to any access to the Flash memory. Refer to *Section 11.7 MICROWIRE/PLUS ISP* for more information on available ISP commands. On Reset, the PGMTIM register is loaded with the value that corresponds to 10 MHz frequency for CKI. The best choice for value of PGMTIM will center the operating frequency in the CKI Frequency Range.

**TABLE 10. PGMTIM Register Format**

PGMTIM								
Register Bit								CKI Frequency Range
7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	25 kHz–50 kHz
0	0	0	0	0	0	0	1	50 kHz–100 kHz
0	0	0	0	0	0	1	0	75 kHz–150 kHz
0	0	0	0	0	1	0	0	125 kHz–250 kHz
0	0	0	0	0	1	1	1	200 kHz–400 kHz



## 11.0 In-System Programming (Continued)

TABLE 10. PGMTIM Register Format (Continued)

PGMTIM								
Register Bit								CKI Frequency Range
7	6	5	4	3	2	1	0	
0	0	0	0	1	0	1	0	275 kHz–550 kHz
0	0	0	1	0	0	0	0	425 kHz–850 kHz
0	0	0	1	1	0	0	0	625 kHz–1.25 MHz
0	0	1	0	1	0	0	0	1.025 MHz–2.05 MHz
0	1	0	0	0	1	0	1	1.5 MHz–3 MHz
0	1	0	0	1	0	0	1	2.5 MHz–5 MHz
0	1	0	0	1	1	1	0	3.75 MHz–7.5 MHz
0	1	0	1	1	0	1	0	6.75 MHz–13.5 MHz
0	1	1	0	1	1	0	0	11.25 MHz–22.5 MHz
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### 11.4 MANEUVERING BACK AND FORTH BETWEEN FLASH MEMORY AND BOOT ROM

When using ISP, at some point, it will be necessary to maneuver between the Flash program memory and the Boot ROM, even when using customized ISP routines. This is because it's not possible to execute from the Flash program memory while it's being programmed.

Two instructions are available to perform the jumping back and forth: Jump to Boot (JSRB) and Return to Flash (RETF). The JSRB instruction is used to jump from Flash memory to Boot ROM, and the RETF is used to return from the Boot ROM back to the Flash program memory. See *Section 19.0 Instruction Set* for specific details on the operation of these instructions.

The JSRB instruction must be used in conjunction with the Key register. This is to prevent jumping to the Boot ROM in the event of run-away software. For the JSRB instruction to actually jump to the Boot ROM, the Key bit must be set. This is done by writing the value shown in *Table 11* to the Key register. The Key is a 6 bit key and if the key matches, the KEY bit will be set for 8 instruction cycles. The JSRB instruction must be executed while the KEY bit is set. If the KEY does not match, then the KEY bit will not be set and the JSRB will jump to the specified location in the Flash memory. In emulation mode, if a breakpoint is encountered while the KEY is set, the counter that counts the instruction cycles will be frozen until the breakpoint condition is cleared. If an interrupt occurs while the key is set, the key will expire before interrupt service is complete. **It is recommended that the software globally disable interrupts before setting the key and re-enable interrupts on completion of Boot ROM execution.** The Key register is a memory mapped register. Its format when writing is shown in *Table 11*.

TABLE 11. KEY Register Write Format

KEY When Writing							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	1	1	0	X	X

**Bits 7–2:** Key value that must be written to set the KEY bit.

**Bits 1–0:** Don't care.

### 11.5 FORCED EXECUTION FROM BOOT ROM

When the user is developing a customized ISP routine, code lockups due to software errors may be encountered. The normal, and preferred, method to recover from these conditions is to reprogram the device with the corrected code by either an external parallel programmer or the emulation tools. As a last resort, when this equipment is not available, there is a hardware method to get out of these lockups and force execution from the Boot ROM MICROWIRE/PLUS routine. The customer will then be able to erase the Flash Memory code and start over.

The method to force this condition is to shift a 24 bit code into the G0 pin, using the G2 pin as a shift clock, while Reset is activated. This special condition will bypass checking the state of the Flex bit in the Option Register and will start execution from location 0000 in the Boot ROM. In this state, the user can input the appropriate commands, using MICROWIRE/PLUS, to erase the Flash program memory and reprogram it. If the device is subsequently reset before the Flex bit has been erased by specific Page Erase or Mass Erase ISP commands, execution will start from location 0000 in the Flash program memory. The forced entry to MICROWIRE/PLUS ISP mode will not erase either the Flex or the Security bit in the Option Register. The Security bit, if set, can only be erased by a Mass Erase of the entire contents of the Flash Memory unless under the control of User ISP routines in the Application Program.

The MICROWIRE/PLUS routine in Boot ROM monitors the G6 input, waits for it to go low, debounces it, and then enables the ISP routine. The user may wish to disconnect other circuitry while  $\overline{\text{RESET}}$ , G0, G2 and the MICROWIRE/PLUS pins are in use for ISP.

The correct sequence to be used to force execution from Boot ROM is :

1. Apply  $V_{CC}$  to the device.
2. Pull RESET Low.
3. Using G2 as a shift clock, shift a value of 5E38AC(hex) into the G0 pin least significant bit first.
4. Pull RESET High.
5. Pull G6 low and initiate the MICROWIRE/PLUS communication.

## 11.0 In-System Programming

(Continued)

### 11.6 RETURN TO FLASH MEMORY WITHOUT HARDWARE RESET

After programming the entire program memory, including options, it is necessary to exit the Boot ROM and return to the Flash program memory for program execution. Upon receipt and completion of the EXIT command through the MICROWIRE/PLUS ISP, the ISP code will reset the part and begin execution from the Flash program memory as described in the Reset section. This assumes that the FLEX bit in the Option register was programmed to 1.

### 11.7 MICROWIRE/PLUS ISP

National Semiconductor provides a program, which is available from our web site at [www.national.com/cop8](http://www.national.com/cop8), that is capable of programming a device from the parallel port of a PC. The software accepts manually input commands and is capable of downloading standard Intel HEX Format files.

Users who wish to write their own MICROWIRE/PLUS ISP host software should refer to the COP8 FLASH ISP User

Manual, available from the same web site. This document includes details of command format and delays necessary between command bytes.

The MICROWIRE/PLUS ISP supports the following features and commands:

- Write a value to the ISP Write Timing Register. NOTE: This must be the first command after entering MICROWIRE/PLUS ISP mode.
- Erase the entire Flash program memory (mass erase).
- Erase a page at a specified address.
- Read Option register.
- Read a byte from a specified address.
- Write a byte to a specified address.
- Read multiple bytes starting at a specified address.
- Write multiple bytes starting at a specified address.
- Exit ISP and return execution to Flash program memory.

The following table lists the MICROWIRE/PLUS ISP commands and provides information on required parameters and return values.

**TABLE 12. MICROWIRE/PLUS ISP Commands**

Command	Function	Command Value (Hex)	Parameters	Return Data
<b>PGMTIM_SET</b>	Write Pulse Timing Register	0x3B	Value	N/A
<b>PAGE_ERASE</b>	Page Erase	0xB3	Starting Address of Page	N/A
<b>MASS_ERASE</b>	Mass Erase	0xBF	Confirmation Code	N/A (The entire Flash Memory will be erased)
<b>READ_BYTE</b>	Read Byte	0x1D	Address High, Address Low	Data Byte if Security not set. 0xFF if Security set. Option Register if address = 0xFFFF, regardless of Security
<b>BLOCKR</b>	Block Read	0xA3	Address High, Address Low, Byte Count (n) High, Byte Count (n) Low $0 \leq n \leq 32767$	n Data Bytes if Security not set. n Bytes of 0xFF if Security set.
<b>WRITE_BYTE</b>	Write Byte	0x71	Address High, Address Low, Data Byte	N/A
<b>BLOCKW</b>	Block Write	0x8F	Address High, Address Low, Byte Count ( $0 \leq n \leq 16$ ), n Data Bytes	N/A
<b>EXIT</b>	EXIT	0xD3	N/A	N/A (Device will Reset)
<b>INVALID</b>	N/A		Any other invalid command will be ignored	N/A

**Note:** The user must ensure that Block Writes do not cross a 64 byte boundary within one operation.



## 12.0 Timers

The device contains a very versatile set of timers (T0 and T1). Timer T1 and associated autoreload/capture registers power up containing random data.

### 12.1 TIMER T0 (IDLE TIMER)

The device supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE Timer T0, which is a 16-bit timer. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The clock to the IDLE Timer is the instruction cycle clock (one-tenth of the CKI frequency).

In addition to its time base function, the Timer T0 supports the following functions:

- Exit out of the Idle Mode (See Idle Mode description)
- WATCHDOG logic (See WATCHDOG description)
- Start up delay out of the HALT mode
- Start up delay from POR

Figure 19 is a functional block diagram showing the structure of the IDLE Timer and its associated interrupt logic.

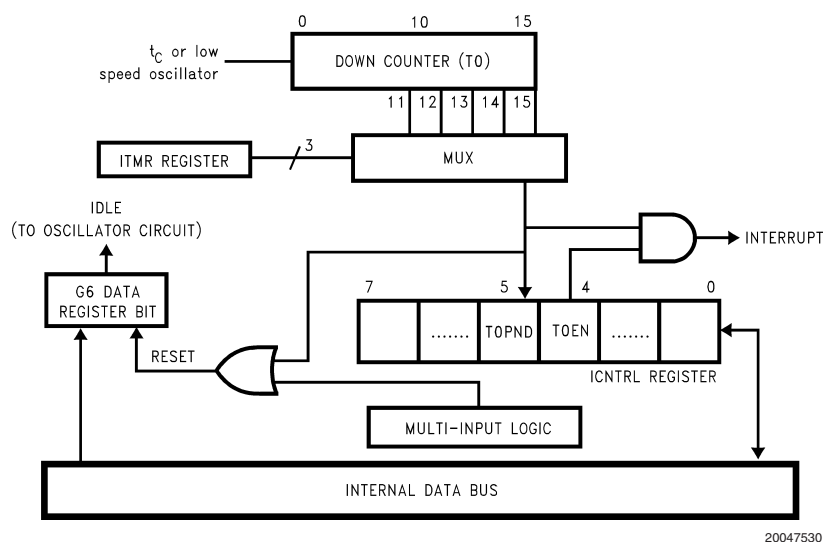


FIGURE 19. Functional Block Diagram for Idle Timer T0

TABLE 13. Idle Timer Window Length

ITSEL2	ITSEL1	ITSEL0	Idle Timer Period
0	0	0	4,096 inst. cycles
0	0	1	8,192 inst. cycles
0	1	0	16,384 inst. cycles
0	1	1	32,768 inst. cycles
1	0	0	65,536 inst. cycles
1	0	1	Reserved - Undefined
1	1	0	Reserved - Undefined
1	1	1	Reserved - Undefined

The ITSEL bits of the ITMR register are cleared on Reset and the Idle Timer period is reset to 4,096 instruction cycles.

Bits 11 through 15 of the ITMR register can be selected for triggering the IDLE Timer interrupt. Each time the selected bit underflows (every 4k, 8k, 16k, 32k or 64k selected clocks), the IDLE Timer interrupt pending bit TOPND is set, thus generating an interrupt (if enabled), and bit 6 of the Port G data register is reset, thus causing an exit from the IDLE mode if the device is in that mode.

In order for an interrupt to be generated, the IDLE Timer interrupt enable bit TOEN must be set, and the GIE (Global Interrupt Enable) bit must also be set. The TOPND flag and TOEN bit are bits 5 and 4 of the ICNTRL register, respectively. The interrupt can be used for any purpose. Typically, it is used to perform a task upon exit from the IDLE mode. For more information on the IDLE mode, refer to *Section 13.2 IDLE MODE*.

The Idle Timer period is selected by bits 0–2 of the ITMR register. Bit 3 of the ITMR Register is reserved and should not be used as a software flag. Bits 4 through 7 of the ITMR Register are reserved and must be zero.

#### 12.1.1 ITMR Register

RSVD					ITSEL2	ITSEL1	ITSEL0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

**RSVD:** These bits are reserved and must be set to 0.

**ITSEL2:0:** Selects the Idle Timer period as described in *Table 13*

Any time the IDLE Timer period is changed there is the possibility of generating a spurious IDLE Timer interrupt by setting the TOPND bit. The user is advised to disable IDLE Timer interrupts prior to changing the value of the ITSEL bits of the ITMR Register and then clear the TOPND bit before attempting to synchronize operation to the IDLE Timer.

## 12.0 Timers (Continued)

### 12.2 TIMER T1

One of the main functions of a microcontroller is to provide timing and counting capability for real-time control tasks. The COP8 family offers a very versatile 16-bit timer/counter structure, and two supporting 16-bit autoreload/capture registers (R1A and R1B), optimized to reduce software burdens in real-time control applications. The timer block has two pins associated with it, T1A and T1B. Pin T1A supports I/O required by the timer block, while pin T1B is an input to the timer block.

The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits T1C3, T1C2, and T1C1 allow selection of the different modes of operation.

### 12.3 MODE 1. PROCESSOR INDEPENDENT PWM MODE

One of the timer's operating modes is the Processor Independent PWM mode. In this mode, the timer generates a "Processor Independent" PWM signal because once the timer is setup, no more action is required from the CPU which translates to less software overhead and greater throughput. The user software services the timer block only when the PWM parameters require updating. This capability is provided by the fact that the timer has two separate 16-bit reload registers. One of the reload registers contains the "ON" timer while the other holds the "OFF" time. By contrast, a microcontroller that has only a single reload register requires an additional software to update the reload value (alternate between the on-time/off-time).

The timer can generate the PWM output with the width and duty cycle controlled by the values stored in the reload registers. The reload registers control the countdown values and the reload values are automatically written into the timer when it counts down through 0, generating interrupt on each reload. Under software control and with minimal overhead, the PWM outputs are useful in controlling motors, triacs, the intensity of displays, and in providing inputs for data acquisition and sine wave generators.

In this mode, the timer T1 counts down at a fixed rate of  $t_C$ . Upon every underflow the timer is alternately reloaded with the contents of supporting registers, R1A and R1B. The very first underflow of the timer causes the timer to reload from the register R1A. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register R1B.

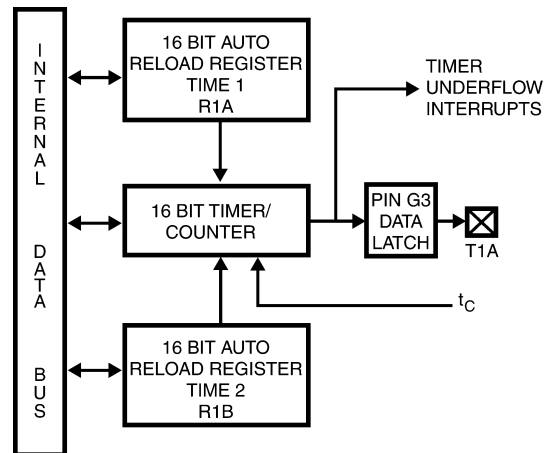
The T1 Timer control bits, T1C3, T1C2 and T1C1 set up the timer for PWM mode operation.

Figure 20 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the T1A output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, T1PNDA and T1PNDB. The user must reset these pending flags under software control. Two control enable flags, T1ENA and T1ENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag T1ENA will cause an interrupt when a timer underflow causes the R1A register to be reloaded into the timer. Setting the timer enable flag T1ENB will cause an

interrupt when a timer underflow causes the R1B register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.



20047531

FIGURE 20. Timer in PWM Mode

### 12.4 MODE 2. EXTERNAL EVENT COUNTER MODE

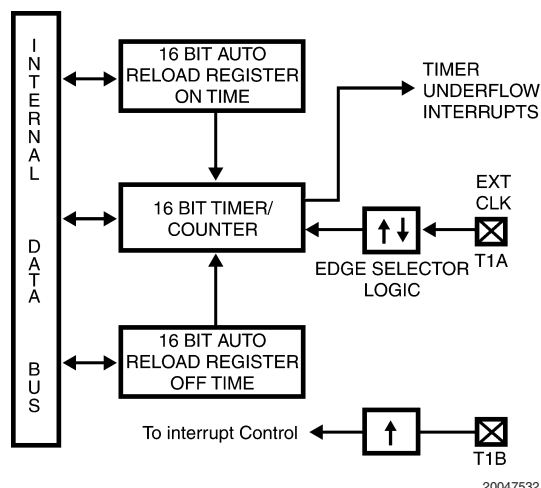
This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, T1, is clocked by the input signal from the T1A pin. The T1 timer control bits, T1C3, T1C2 and T1C1 allow the timer to be clocked either on a positive or negative edge from the T1A pin. Underflows from the timer are latched into the T1PNDA pending flag. Setting the T1ENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin T1B can be used as an independent positive edge sensitive interrupt input if the T1ENB control flag is set. The occurrence of a positive edge on the T1B input pin is latched into the T1PNDB flag.

Figure 21 shows a block diagram of the timer in External Event Counter mode.

**Note:** The PWM output is not available in this mode since the T1A pin is being used as the counter input clock.

## 12.0 Timers (Continued)



### FIGURE 21. Timer in External Event Counter Mode

## 12.5 MODE 3. INPUT CAPTURE MODE

The device can precisely measure external frequencies or time external events by placing the timer block, T1, in the input capture mode. In this mode, the reload registers serve as independent capture registers, capturing the contents of the timer when an external event occurs (transition on the timer input pin). The capture registers can be read while maintaining count, a feature that lets the user measure elapsed time and time between events. By saving the timer value when the external event occurs, the time of the external event is recorded. Most microcontrollers have a latency time because they cannot determine the timer value when the external event occurs. The capture register eliminates the latency time, thereby allowing the applications program to retrieve the timer value stored in the capture register.

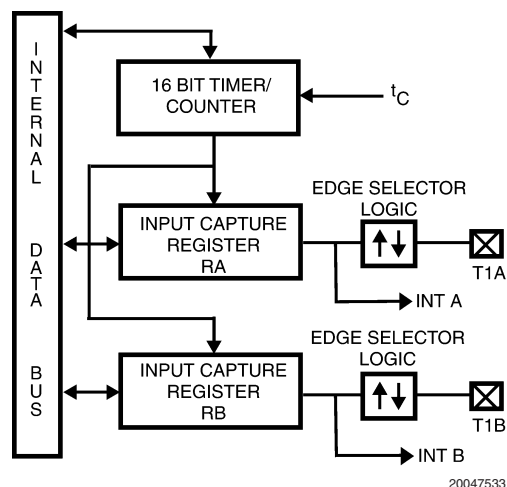
In this mode, the timer T1 is constantly running at the fixed  $t_C$  rate. The two registers, R1A and R1B, act as capture registers. Each register acts in conjunction with a pin. The register R1A acts in conjunction with the T1A pin and the register R1B acts in conjunction with the T1B pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, T1C3, T1C2 and T1C1, allow the trigger events to be speci-

fied either as a positive or a negative edge. The trigger condition for each input pin can be specified independently. The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the T1A and T1B pins will be respectively latched into the pending flags, T1PNDA and T1PNDB. The control flag T1ENA allows the interrupt on T1A to be either enabled or disabled. Setting the T1ENA flag enables interrupts to be generated when the selected trigger condition occurs on the T1A pin. Similarly, the flag T1ENB controls the interrupts from the T1B pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer T1C0 pending flag (the T1C0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the T1C0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the T1ENA control flag. When a T1A interrupt occurs in the Input Capture mode, the user must check both the T1PND and T1C0 pending flags in order to determine whether a T1A input capture or a timer underflow (or both) caused the interrupt.

Figure 22 shows a block diagram of the timer in Input Capture mode.



**FIGURE 22. Timer in Input Capture Mode**

## 12.0 Timers (Continued)

### 12.6 TIMER CONTROL FLAGS

The control bits and their functions are summarized below.

T1C3	Timer mode control
T1C2	Timer mode control
T1C1	Timer mode control
T1C0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop
	Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)

T1PNDA Timer Interrupt Pending Flag

T1ENA Timer Interrupt Enable Flag

1 = Timer Interrupt Enabled

0 = Timer Interrupt Disabled

T1PNDB Timer Interrupt Pending Flag

T1ENB Timer Interrupt Enable Flag

1 = Timer Interrupt Enabled

0 = Timer Interrupt Disabled

The timer mode control bits (T1C3, T1C2 and T1C1) are detailed below:

Mode	T1C3	T1C2	T1C1	Description	Interrupt A Source	Interrupt B Source	Timer Counts On
1	1	0	1	PWM: T1A Toggle	Autoreload RA	Autoreload RB	$t_c$
	1	0	0	PWM: No T1A Toggle	Autoreload RA	Autoreload RB	$t_c$
2	0	0	0	External Event Counter	Timer Underflow	Pos. T1B Edge	Pos. T1A Edge
	0	0	1	External Event Counter	Timer Underflow	Pos. T1B Edge	Pos. T1A Edge
3	0	1	0	Captures: T1A Pos. Edge T1B Pos. Edge	Pos. T1A Edge or Timer Underflow	Pos. T1B Edge	$t_c$
	1	1	0	Captures: T1A Pos. Edge T1B Neg. Edge	Pos. T1A Edge or Timer Underflow	Neg. T1B Edge	$t_c$
	0	1	1	Captures: T1A Neg. Edge T1B Neg. Edge	Neg. T1A Edge or Timer Underflow	Neg. T1B Edge	$t_c$
	1	1	1	Captures: T1A Neg. Edge T1B Neg. Edge	Neg. T1A Edge or Timer Underflow	Neg. T1B Edge	$t_c$

## 13.0 Power Save Modes

Today, the proliferation of battery-operated based applications has placed new demands on designers to drive power consumption down. Battery-operated systems are not the only type of applications demanding low power. The power budget constraints are also imposed on those consumer/industrial applications where well regulated and expensive power supply costs cannot be tolerated. Such applications rely on low cost and low power supply voltage derived directly from the “mains” by using voltage rectifier and passive components. Low power is demanded even in automotive applications, due to increased vehicle electronics content. This is required to ease the burden from the car battery. Low power 8-bit microcontrollers supply the smarts to control battery-operated, consumer/industrial, and automotive applications.

The COP8TAX devices offer system designers a variety of low-power consumption features that enable them to meet the demanding requirements of today's increasing range of low-power applications. These features include low voltage operation, low current drain, and power saving features such as HALT, IDLE, and Multi-Input Wake-Up (MIWU).

The devices offer the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller

activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

Clock Monitor if enabled can be active in both modes.

### 13.1 HALT MODE

The device can be placed in the HALT mode by writing a “1” to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WATCHDOG logic on the device is disabled during the HALT mode. However, the clock monitor circuitry, if enabled, remains active and will cause the WATCHDOG output pin (WDOUT) to go low. If the HALT mode is used and the user does not want to activate the WDOUT pin, the Clock Monitor should be disabled after the device comes out of reset (resetting the Clock Monitor control bit with the first write to the WDSVR register). In the HALT mode, the power requirements of the device are minimal and the applied voltage ( $V_{CC}$ ) may be decreased to  $V_r$  ( $V_r = 2.0V$ ) without altering the state of the machine.

The device supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wake-Up feature on Port L and Port C. The

## 13.0 Power Save Modes (Continued)

second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may only be used with an R/C clock configuration. The third method of exiting the HALT mode is by pulling the  $\overline{\text{RESET}}$  pin low.

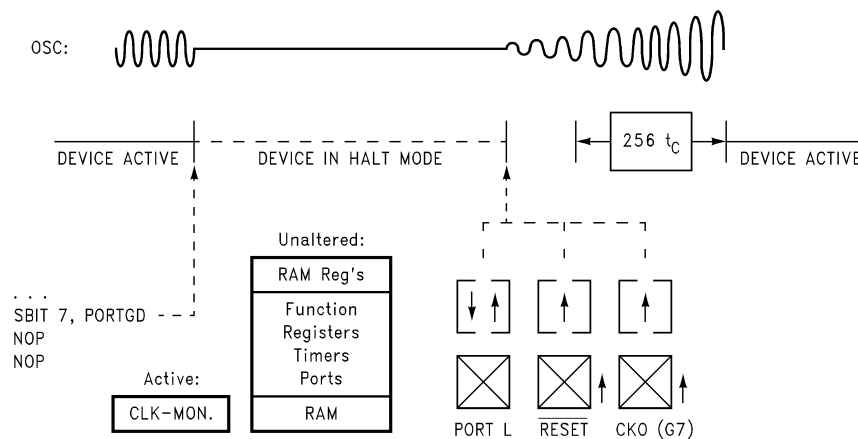
Since a crystal or ceramic resonator may be selected as the oscillator, the Multi-Input Wake-Up signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Multi-Input Wake-Up signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the  $t_C$  instruction cycle clock. The  $t_C$  clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The start-up time-out from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an R/C clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The device has two options associated with the HALT mode. The first option enables the HALT mode feature, while the second option disables the HALT mode selected through bit 0 of the Option Byte. With the HALT mode enable option, the device will enter and exit the HALT mode as described above. With the HALT disable option, the device cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect, the HALT flag will remain "0").

The WATCHDOG detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the device inadvertently enters the HALT mode as a result of a runaway program or power glitch.

If the device is placed in the HALT mode, with the R/C oscillator selected, the clock input pin (CKI) is forced to a logic high internally. With the crystal or external oscillator the CKI pin is TRI-STATE.



20047534

FIGURE 23. Multi-Input Wake-Up from HALT

### 13.2 IDLE MODE

The device is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry and the IDLE Timer T0, are stopped.

As with the HALT mode, the device can be returned to normal operation with a reset, or with a Multi-Input Wake-Up from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the twelfth bit (representing 4.096 ms at internal clock frequency of 10 MHz,  $t_C = 1 \mu s$ ) of the IDLE Timer toggles.

This toggle condition of the twelfth bit of the IDLE Timer T0 is latched into the TOPND pending flag.

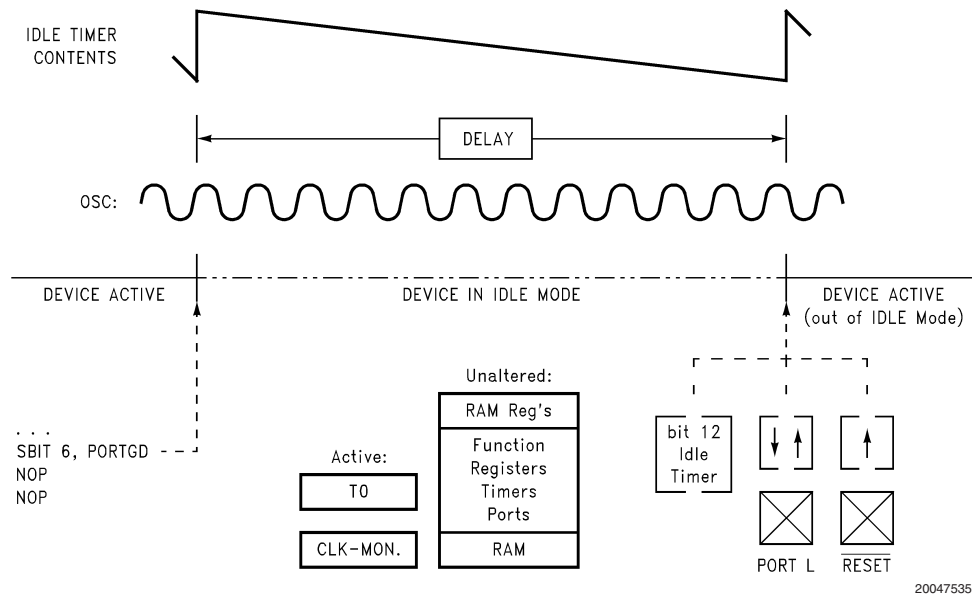
The user has the option of being interrupted with a transition on the twelfth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the TOPND bit gets set, the device will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the device will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

**Note:** It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

## 13.0 Power Save Modes (Continued)



20047535

FIGURE 24. Wake-Up from IDLE

### 13.3 MULTI-INPUT WAKE-UP

The Multi-Input Wake-Up feature is used to exit from the HALT and IDLE modes. In addition, the Multi-Input Wake-Up/Interrupt feature may be used to generate up to 16 edge-selectable external interrupts on the 44-pin devices or 8 interrupts on the 20- and 28-pin devices. Figure 25 shows the Multi-Input Wake-Up logic.

The Multi-Input Wake-Up feature uses the C and L ports. (The 20- and 28-pin devices only have the L port.) Software selects which port bit (or set of port bits) may cause the device to exit the HALT or IDLE modes. The selection is controlled by the CWKEN and LWKEN registers. These registers are 8-bit read/write registers, which contain control bits that correspond to the C and L port bits. Setting a CWKEN or LWKEN bit enables a Wake-Up event or interrupt from the associated C or L port pin.

If the ACCESS.Bus module is enabled, port pin L0 may also be used to generate a Wake-Up event on ACCESS.Bus activity. Please see the section on *Section 17.0 ACCESS.Bus Interface* for more information.

Software selects whether the trigger condition on the selected port pin is a positive edge (low-to-high transition) or a negative edge (high-to-low transition). The trigger conditions are selected in the CWKEDG and LWKEDG registers, which are 8-bit control registers with bits corresponding to the C and L port pins. Setting a trigger condition control bit selects the negative edge, while clearing the bit selects the positive edge.

The occurrence of a selected trigger condition is latched in the pending registers called CWKPND and LWKPND. The bits of these registers correspond to the C and L port pins. These bits are set on the occurrence of the selected trigger condition on the corresponding port pin, whether or not the trigger condition is enabled in CWKEN or LWKEN. Software has responsibility for clearing the pending bits before enabling them for Wake-Up events or interrupts. Any set pend-

ing bit in CWKPND or LWKPND remains set until cleared by software. The device will not enter HALT or IDLE mode if any Wake-Up input is both enabled and pending.

Changing a trigger condition control bit requires several steps to avoid generating a spurious Wake-Up event or interrupt as a side effect

- First, the corresponding CWKEN or LWKEN bit should be cleared to disable any Wake-Up event or interrupt for that port pin.
- Second, the trigger condition is selected in CWKEDG or LWKEDG.
- Third, any spurious pending event is removed by clearing the associated bit in CWKPND or LWKPND.
- Finally, the trigger is re-enabled by setting the associated bit in CWKEN or LWKEN.

An example shows how software performs this procedure. Assume the trigger condition for L port bit 5 is to be changed from positive (low-to-high transition) to negative (high-to-low transition), and bit 5 has previously been enabled for an input interrupt. Software would execute the following instructions:

```
RBIT 5, LWKEN    ; Disable MIWU Port L.5
SBIT 5, LWKEDG   ; Change edge polarity
RBIT 5, LWKPND   ; Reset pending flag
SBIT 5, LWKEN    ; Enable MIWU Port L.5
```

If the C or L port pins have been used as outputs and then changed to inputs using the Multi-Input Wake-Up feature, a safe procedure should be used to avoid generating a spurious Wake-Up event or interrupt. After the selected C or L port pins have been changed from output to input, the trigger conditions are selected in CWKEDG or LWKEDG and the pending bits in CWKPND or LWKPND are cleared. Finally, the CWKEN or LWKEN bits are set to enable the desired Wake-Up events or interrupts.

The same procedure should be used following reset, because the C and L port pins are left floating. The CWKPND



## 13.0 Power Save Modes (Continued)

and LWKPND registers contain undefined values after reset, so software should clear these bits after reset to ensure that no spurious Wake-Up events or interrupts are left pending.

CWKEN and LWKEN are read/write registers that are cleared at reset, so no Wake-Up events or interrupts are enabled following reset. CWKEDG and LWKEDG are also cleared at reset.

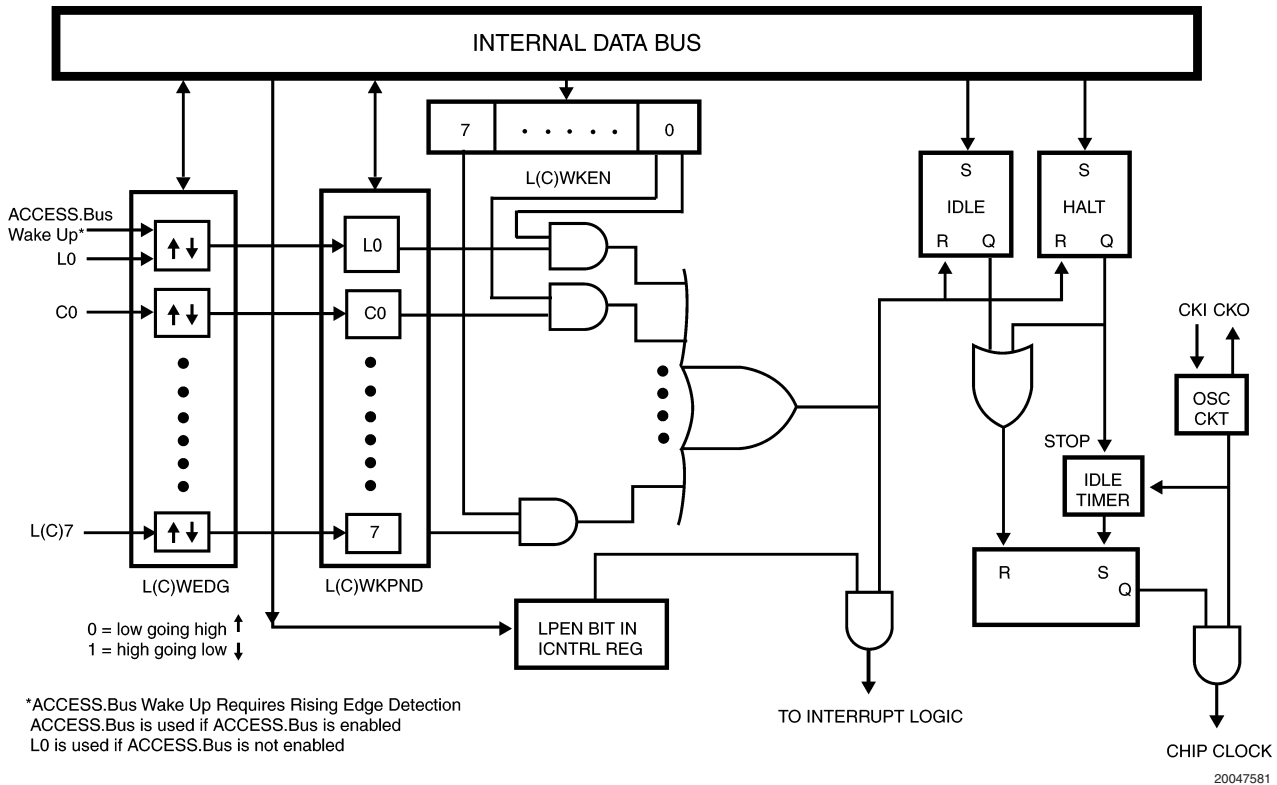


FIGURE 25. Multi-Input Wake-Up Logic

## 14.0 Interrupts

### 14.1 INTRODUCTION

The device supports eleven vectored interrupts. Interrupt sources include Timer 1, Timer 2, Timer T0, Multi-Input Wake-Up, Software Trap, MICROWIRE/PLUS and External Input.

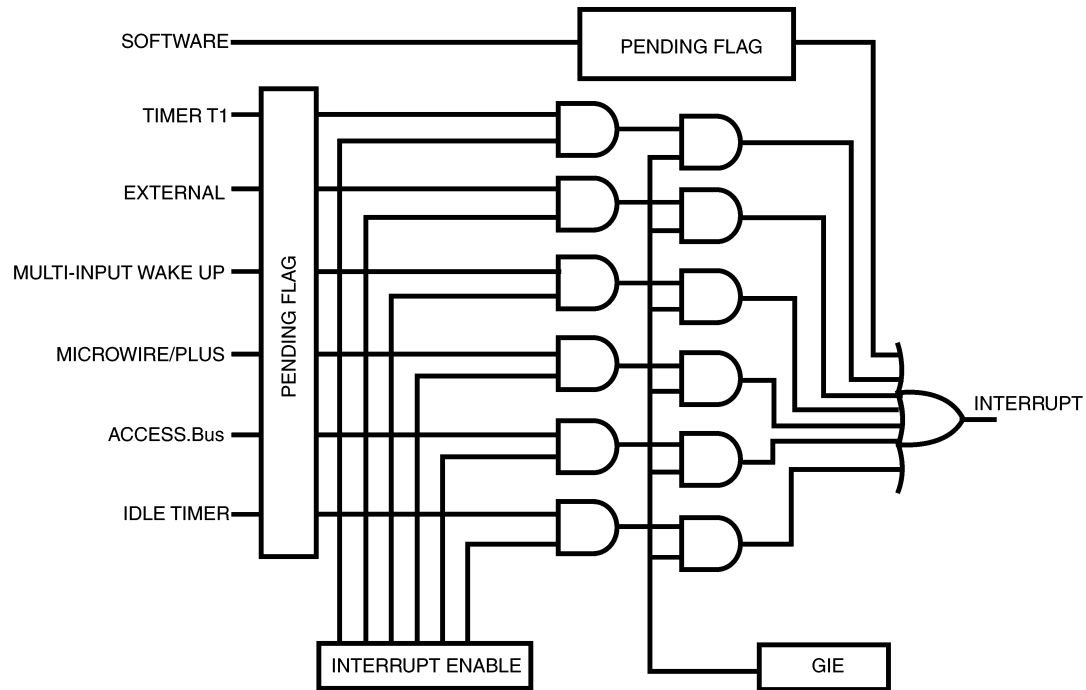
All interrupts force a branch to location 00FF Hex in program memory. The VIS instruction may be used to vector to the appropriate service routine from location 00FF Hex.

The Software trap has the highest priority while the default VIS has the lowest priority.

Each of the 13 maskable inputs has a fixed arbitration ranking and vector.

Figure 26 shows the Interrupt block diagram.

## 14.0 Interrupts (Continued)



20047576

FIGURE 26. Interrupt Block Diagram

### 14.2 MASKABLE INTERRUPTS

All interrupts other than the Software Trap are maskable. Each maskable interrupt has an associated enable bit and pending flag bit. The pending bit is set to 1 when the interrupt condition occurs. The state of the interrupt enable bit, combined with the GIE bit determines whether an active pending flag actually triggers an interrupt. All of the maskable interrupt pending and enable bits are contained in mapped control registers, and thus can be controlled by the software.

A maskable interrupt condition triggers an interrupt under the following conditions:

1. The enable bit associated with that interrupt is set.
2. The GIE bit is set.
3. The device is not processing a non-maskable interrupt. (If a non-maskable interrupt is being serviced, a maskable interrupt must wait until that service routine is completed.)

An interrupt is triggered only when all of these conditions are met at the beginning of an instruction. If different maskable interrupts meet these conditions simultaneously, the highest-priority interrupt will be serviced first, and the other pending interrupts must wait.

Upon Reset, all pending bits, individual enable bits, and the GIE bit are reset to zero. Thus, a maskable interrupt condition cannot trigger an interrupt until the program enables it by setting both the GIE bit and the individual enable bit. When enabling an interrupt, the user should consider whether or not a previously activated (set) pending bit should be acknowledged. If, at the time an interrupt is enabled, any previous occurrences of the interrupt should be ignored, the associated pending bit must be reset to zero prior to enabling the interrupt. Otherwise, the interrupt may be simply

enabled; if the pending bit is already set, it will immediately trigger an interrupt. A maskable interrupt is active if its associated enable and pending bits are set.

An interrupt is an asynchronous event which may occur before, during, or after an instruction cycle. Any interrupt which occurs during the execution of an instruction is not acknowledged until the start of the next normally executed instruction. If the next normally executed instruction is to be skipped, the skip is performed before the pending interrupt is acknowledged.

At the start of interrupt acknowledgment, the following actions occur:

1. The GIE bit is automatically reset to zero, preventing any subsequent maskable interrupt from interrupting the current service routine. This feature prevents one maskable interrupt from interrupting another one being serviced.
2. The address of the instruction about to be executed is pushed onto the stack.
3. The program counter (PC) is loaded with 00FF Hex, causing a jump to that program memory location.

The device requires seven instruction cycles to perform the actions listed above.

If the user wishes to allow nested interrupts, the interrupts service routine may set the GIE bit to 1 by writing to the PSW register, and thus allow other maskable interrupts to interrupt the current service routine. If nested interrupts are allowed, caution must be exercised. The user must write the program in such a way as to prevent stack overflow, loss of saved context information, and other unwanted conditions.

The interrupt service routine stored at location 00FF Hex should use the VIS instruction to determine the cause of the interrupt, and jump to the interrupt handling routine corre-



## 14.0 Interrupts (Continued)

sponding to the highest priority enabled and active interrupt. Alternately, the user may choose to poll all interrupt pending and enable bits to determine the source(s) of the interrupt. If more than one interrupt is active, the user's program must decide which interrupt to service.

Within a specific interrupt service routine, the associated pending bit should be cleared. This is typically done as early as possible in the service routine in order to avoid missing the next occurrence of the same type of interrupt event. Thus, if the same event occurs a second time, even while the first occurrence is still being serviced, the second occurrence will be serviced immediately upon return from the current interrupt routine.

An interrupt service routine typically ends with an RETI instruction. This instruction set the GIE bit back to 1, pops the address stored on the stack, and restores that address to the program counter. Program execution then proceeds with the next instruction that would have been executed had there been no interrupt. If there are any valid interrupts pending, the highest-priority interrupt is serviced immediately upon return from the previous interrupt.

**Note:** While executing from the Boot ROM for ISP or virtual E2 operations, the hardware will disable interrupts from occurring. The hardware will leave the GIE bit in its current state, and if set, the hardware interrupts will occur when execution is returned to Flash Memory. Subsequent interrupts, during ISP operation, from the same interrupt source will be lost.

### 14.3 VIS INSTRUCTION

The general interrupt service routine, which starts at address 00FF Hex, must be capable of handling all types of interrupts. The VIS instruction, together with an interrupt vector table, directs the device to the specific interrupt handling routine based on the cause of the interrupt.

VIS is a single-byte instruction, typically used at the very beginning of the general interrupt service routine at address 00FF Hex, or shortly after that point, just after the code used for context switching. The VIS instruction determines which enabled and pending interrupt has the highest priority, and causes an indirect jump to the address corresponding to that interrupt source. The jump addresses (vectors) for all possible interrupts sources are stored in a vector table.

The vector table may be as long as 32 bytes (maximum of 16 vectors) and resides at the top of the 256-byte block containing the VIS instruction. However, if the VIS instruction is at the very top of a 256-byte block (such as at 00FF Hex), the vector table resides at the top of the next 256-byte block. Thus, if the VIS instruction is located somewhere between 00FF and 01DF Hex (the usual case), the vector table is located between addresses 01E0 and 01FF Hex. If the VIS instruction is located between 01FF and 02DF Hex, then the vector table is located between addresses 02E0 and 02FF Hex, and so on.

Each vector is 15 bits long and points to the beginning of a specific interrupt service routine somewhere in the 32-kbyte memory space. Each vector occupies two bytes of the vector table, with the higher-order byte at the lower address. The vectors are arranged in order of interrupt priority. The vector of the maskable interrupt with the lowest rank is located to 0yE0 (higher-order byte) and 0yE1 (lower-order byte). The

next priority interrupt is located at 0yE2 and 0yE3, and so forth in increasing rank. The Software Trap has the highest rank and its vector is always located at 0yFE and 0yFF. The number of interrupts which can become active defines the size of the table.

Table 14 shows the types of interrupts, the interrupt arbitration ranking, and the locations of the corresponding vectors in the vector table.

The vector table should be filled by the user with the memory locations of the specific interrupt service routines. For example, if the Software Trap routine is located at 0310 Hex, then the vector location 0yFE and -0yFF should contain the data 03 and 10 Hex, respectively. When a Software Trap interrupt occurs and the VIS instruction is executed, the program jumps to the address specified in the vector table.

The interrupt sources in the vector table are listed in order of rank, from highest to lowest priority. If two or more enabled and pending interrupts are detected at the same time, the one with the highest priority is serviced first. Upon return from the interrupt service routine, the next highest-level pending interrupt is serviced.

If the VIS instruction is executed, but no interrupts are enabled and pending, the lowest-priority interrupt vector is used, and a jump is made to the corresponding address in the vector table. This is an unusual occurrence and may be the result of an error. It can legitimately result from a change in the enable bits or pending flags prior to the execution of the VIS instruction, such as executing a single cycle instruction which clears an enable flag at the same time that the pending flag is set. It can also result, however, from inadvertent execution of the VIS command outside of the context of an interrupt.

The default VIS interrupt vector can be useful for applications in which time critical interrupts can occur during the servicing of another interrupt. Rather than restoring the program context (A, B, X, etc.) and executing the RETI instruction, an interrupt service routine can be terminated by returning to the VIS instruction. In this case, interrupts will be serviced in turn until no further interrupts are pending and the default VIS routine is started. After testing the GIE bit to ensure that execution is not erroneous, the routine should restore the program context and execute the RETI to return to the interrupted program.

This technique can save up to fifty instruction cycles ( $t_c$ ), or more, (100  $\mu$ s at 10 MHz oscillator) of latency for pending interrupts with a penalty of fewer than ten instruction cycles if no further interrupts are pending.

To ensure reliable operation, the user should always use the VIS instruction to determine the source of an interrupt. Although it is possible to poll the pending bits to detect the source of an interrupt, this practice is not recommended. The use of polling allows the standard arbitration ranking to be altered, but the reliability of the interrupt system is compromised. The polling routine must individually test the enable and pending bits of each maskable interrupt. If a Software Trap interrupt should occur, it will be serviced last, even though it should have the highest priority. Under certain conditions, a Software Trap could be triggered but not serviced, resulting in an inadvertent "locking out" of all maskable interrupts by the Software Trap pending flag. Problems such as this can be avoided by using VIS instruction.

## 14.0 Interrupts (Continued)

TABLE 14. Interrupt Vector Table

Arbitration Ranking	Source Description		Vector Address (Note 7) (Hi-Low Byte)
(1) Highest	Software	INTR Instruction	0yFE–0yFF
(2)	Reserved for NMI		0yFC–0yFD
(3)	External	G0	0yFA–0yFB
(4)	Timer T0	Underflow	0yF8–0yF9
(5)	Timer T1	T1A/Underflow	0yF6–0yF7
(6)	Timer T1	T1B	0yF4–0yF5
(7)	MICROWIRE/PLUS	BUSY Low	0yF2–0yF3
(8)	ACCESS.Bus	Address Match, Bus Error, Negative Acknowledge, Valid Sto or SDASt is set	0yF0–0yF1
(9)	Reserved		0yEE–0yEF
(10)	Reserved		0yEC–0yED
(11)	Reserved		0yEA–0yEB
(12)	Reserved		0yE8–0yE9
(13)	Reserved		0yE6–0yE7
(14)	Reserved		0yE4–0yE5
(15)	Port L/Wake-up Port C/Wake-up	Port L Edge Port C Edge	0yE2–0yE3
(16) Lowest	Default VIS	Reserved	0yE0–0yE1

**Note 7:** y is a variable which represents the VIS block. VIS and the vector table must be located in the same 256-byte block except if VIS is located at the last address of a block. In this case, the table must be in the next block.

### 14.3.1 VIS Execution

When the VIS instruction is executed it activates the arbitration logic. The arbitration logic generates an even number between E0 and FE (E0, E2, E4, E6 etc....) depending on which active interrupt has the highest arbitration ranking at the time of the 1st cycle of VIS is executed. For example, if the software trap interrupt is active, FE is generated. If the external interrupt is active and the software trap interrupt is not, then FA is generated and so forth. If no active interrupt is pending, then E0 is generated. This number replaces the lower byte of the PC. The upper byte of the PC remains unchanged. The new PC is therefore pointing to the vector of

the active interrupt with the highest arbitration ranking. This vector is read from program memory and placed into the PC which is now pointed to the 1st instruction of the service routine of the active interrupt with the highest arbitration ranking.

Figure 27 illustrates the different steps performed by the VIS instruction. Figure 28 shows a flowchart for the VIS instruction.

The non-maskable interrupt pending flag is cleared by the RPND (Reset Non-Maskable Pending Bit) instruction (under certain conditions) and upon RESET.

## 14.0 Interrupts (Continued)

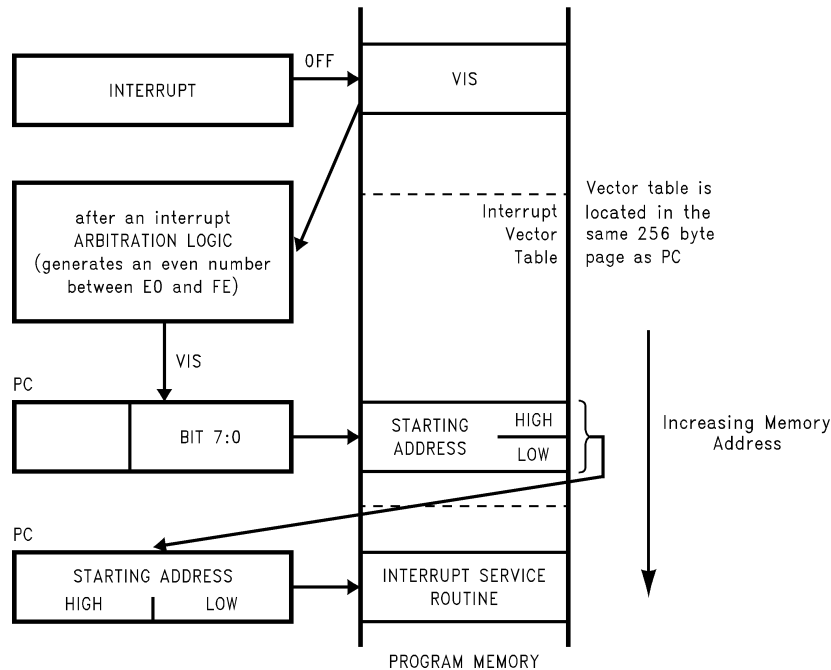


FIGURE 27. VIS Operation

### 14.4 NON-MASKABLE INTERRUPT

#### 14.4.1 Pending Flag

There is a pending flag bit associated with the non-maskable Software Trap interrupt, called STPND. This pending flag is not memory-mapped and cannot be accessed directly by the software.

The pending flag is reset to zero when a device Reset occurs. When the non-maskable interrupt occurs, the associated pending bit is set to 1. The interrupt service routine should contain an RPND instruction to reset the pending flag to zero. The RPND instruction always resets the STPND flag.

#### 14.4.2 Software Trap

The Software Trap is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from program memory and placed in the instruction register. This can happen in a variety of ways, usually because of an error condition. Some examples of causes are listed below.

If the program counter incorrectly points to a memory location beyond the programmed Flash memory space, the unused memory location returns zeros which is interpreted as the INTR instruction.

A Software Trap can be triggered by a temporary hardware condition such as a brownout or power supply glitch.

The Software Trap has the highest priority of all interrupts. When a Software Trap occurs, the STPND bit is set. The GIE bit is not affected and the pending bit (not accessible by the user) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction.

Nothing can interrupt a Software Trap service routine except for another Software Trap. The STPND can be reset only by the RPND instruction or a chip Reset.

The Software Trap indicates an unusual or unknown error condition. Generally, returning to normal execution at the point where the Software Trap occurred cannot be done reliably. Therefore, the Software Trap service routine should re-initialize the stack pointer and perform a recovery procedure that re-starts the software at some known point, similar to a device Reset, but not necessarily performing all the same functions as a device Reset. The routine must also execute the RPND instruction to reset the STPND flag. Otherwise, all other interrupts will be locked out. To the extent possible, the interrupt routine should record or indicate the context of the device so that the cause of the Software Trap can be determined.

If the user wishes to return to normal execution from the point at which the Software Trap was triggered, the user must first execute RPND, followed by RETSK rather than RETI or RET. This is because the return address stored on the stack is the address of the INTR instruction that triggered the interrupt. The program must skip that instruction in order to proceed with the next one. Otherwise, an infinite loop of Software Traps and returns will occur.

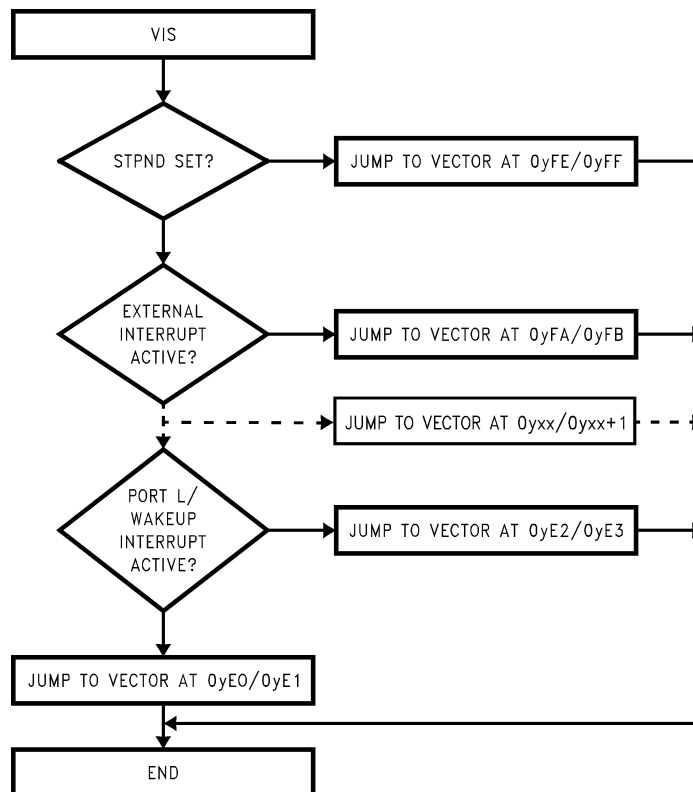
Programming a return to normal execution requires careful consideration. If the Software Trap routine is interrupted by another Software Trap, the RPND instruction in the service routine for the second Software Trap will reset the STPND flag; upon return to the first Software Trap routine, the STPND flag will have the wrong state. This will allow maskable interrupts to be acknowledged during the servicing of the first Software Trap. To avoid problems such as this, the

## 14.0 Interrupts (Continued)

user program should contain the Software Trap routine to perform a recovery procedure rather than a return to normal execution.

Under normal conditions, the STPND flag is reset by a RPND instruction in the Software Trap service routine. If a

programming error or hardware condition (brownout, power supply glitch, etc.) sets the STPND flag without providing a way for it to be cleared, all other interrupts will be locked out. To alleviate this condition, the user can use extra RPND instructions in the main program and in the Watchdog service routine (if present). There is no harm in executing extra RPND instructions in these parts of the program.



20047578

FIGURE 28. VIS Flow Chart

### 14.4.2.1 Programming Example: External Interrupt

```

PSW          =00EF
CNTRL        =00EE
RBIT         0,PORTGC
RBIT         0,PORTGD      ; G0 pin configured Hi-Z
SBIT         IEDG, CNTRL   ; Ext interrupt polarity; falling edge
SBIT         GIE, PSW      ; Set the GIE bit
SBIT         EXEN, PSW     ; Enable the external interrupt
WAIT:        JP           WAIT      ; Wait for external interrupt
.
.
.
.=0FF        ; The interrupt causes a
VIS          ; branch to address 0FF
              ; The VIS causes a branch to
              ; interrupt vector table
.
.
.
.=01FA      ; Vector table (within 256 byte
.ADDRW SERVICE ; of VIS inst.) containing the ext
              ; interrupt service routine
.
.

```

## 14.0 Interrupts (Continued)

```

SERVICE:      .           ; Interrupt Service Routine
               RBIT,EXPND,PSW ; Reset ext interrupt pend. bit
               .
               .
               .
               RET I        ; Return, set the GIE bit

```

### 14.5 PORT C AND PORT L INTERRUPTS

Ports C and L provides the user with an additional sixteen fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Ports C and L share logic with the wake-up circuitry. The registers CWKEN and LWKEN allow interrupts from Ports C and L to be individually enabled or disabled. The register CWKEDG and LWKEDG specify the trigger condition to be either a positive or a negative edge. Finally, the registers CWKPND and LWKPND latch in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port C and Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the registers CWKPND and LWKPND are adequate.

Since Ports C and L are also used for waking the device out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the device will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the device will first execute the interrupt service routine and then revert to normal operation.

### 14.6 INTERRUPT SUMMARY

The device uses the following types of interrupts, listed below in order of priority:

1. The Software Trap non-maskable interrupt, triggered by the INTR (00 opcode) instruction. The Software Trap is acknowledged immediately. This interrupt service routine can be interrupted only by another Software Trap. The Software Trap should end with two RPND instructions followed by a re-start procedure.
2. Maskable interrupts, triggered by an on-chip peripheral block or an external device connected to the device. Under ordinary conditions, a maskable interrupt will not interrupt any other interrupt routine in progress. A maskable interrupt routine in progress can be interrupted by the non-maskable interrupt request. A maskable interrupt routine should end with an RETI instruction or, prior to restoring context, should return to execute the VIS instruction. This is particularly useful when exiting long interrupt service routines if the time between interrupts is short. In this case the RETI instruction would only be executed when the default VIS routine is reached.
3. While executing from the Boot ROM for ISP or virtual E2 operations, the hardware will disable interrupts from occurring. The hardware will leave the GIE bit in its current state, and if set, the hardware interrupts will occur when execution is returned to Flash Memory. Subsequent in-

terrupts, during ISP operation, from the same interrupt source will be lost.

## 15.0 WATCHDOG/Clock Monitor

The devices contain a user selectable WATCHDOG and clock monitor. The following section is applicable only if WATCHDOG feature has been selected in the Option Byte. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs.

The WATCHDOG logic contains two separate service windows. While the user programmable upper window selects the WATCHDOG service time, the lower window provides protection against an infinite program loop that contains the WATCHDOG service instruction.

The COP8TAx devices provide the added feature of a software trap that provides protection against stack overpops and addressing locations outside valid user program space.

The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table 15 shows the WDSVR register.

**TABLE 15. WATCHDOG Service Register (WDSVR)**

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y

The lower limit of the service window is fixed at 256 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table 16 shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

**TABLE 16. WATCHDOG Service Window Select**

WDSVR Bit 7	WDSVR Bit 6	Clock Monitor	Service Window (Lower-Upper Limits)
0	0	x	256–8k $t_C$ Cycles
0	1	x	256–16k $t_C$ Cycles



## 15.0 WATCHDOG/Clock Monitor

(Continued)

**TABLE 16. WATCHDOG Service Window Select** (Continued)

WDSVR Bit 7	WDSVR Bit 6	Clock Monitor	Service Window (Lower-Upper Limits)
1	0	x	256–32k $t_C$ Cycles
1	1	x	256–64k $t_C$ Cycles
x	x	0	Clock Monitor Disabled
x	x	1	Clock Monitor Enabled

### 15.1 CLOCK MONITOR

The Clock Monitor aboard the device can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ( $1/t_C$ ) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

### 15.2 WATCHDOG/CLOCK MONITOR OPERATION

The WATCHDOG is enabled by bit 2 of the Option Byte. When this Option bit is 0, the WATCHDOG is enabled and pin G1 becomes the WATCHDOG output with a weak pullup. The WATCHDOG and Clock Monitor are disabled during reset. The device comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCH-

DOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. Table 17 shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOOUT pin, on pin 1 of the port G. WDOOUT is active low and must be externally connected to the RESET pin or to some other external logic which handles WATCHDOG event. The WDOOUT pin has a weak pullup in the inactive state. This pull-up is sufficient to serve as the connection to  $V_{CC}$  for systems which use the internal Power On Reset. Upon triggering the WATCHDOG, the logic will pull the WDOOUT (G1) pin low for an additional 16  $t_C$ –32  $t_C$  cycles after the signal level on WDOOUT pin goes below the lower Schmitt trigger threshold. After this delay, the device will stop forcing the WDOOUT output low. The WATCHDOG service window will restart when the WDOOUT pin goes high.

A WATCHDOG service while the WDOOUT signal is active will be ignored. The state of the WDOOUT pin is not guaranteed on reset, but if it powers up low then the WATCHDOG will time out and WDOOUT will go high.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will go high following 16  $t_C$ –32  $t_C$  clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

- $1/t_C > 10 \text{ kHz}$  — No clock rejection.
- $1/t_C < 10 \text{ Hz}$  — Guaranteed clock rejection.

**TABLE 17. WATCHDOG Service Actions**

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

### 15.3 WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and CLOCK MONITOR detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and CLOCK MONITOR enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.

## 15.0 WATCHDOG/Clock Monitor

(Continued)

- The CLOCK MONITOR detector circuit is active during both the HALT and IDLE modes. Consequently, the device inadvertently entering the HALT mode will be detected as a CLOCK MONITOR error (provided that the CLOCK MONITOR enable option has been selected by the program).
- With the single-pin R/C oscillator option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator option selected, or with the single-pin R/C oscillator option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 256 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with external RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the T0PND flag. The T0PND flag is set whenever the twelfth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the T0PND flag.
- A hardware WATCHDOG service occurs just as the device exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 256 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 256 instruction cycles without causing a WATCHDOG error.
- In order to RESET the device on the occurrence of a WATCH event, the user must connect the WDOUT pin (G1) pin to the RESET external to the device. The weak pull-up on the WDOUT pin is sufficient to provide the RESET connection to  $V_{CC}$  for devices which use both Power On Reset and WATCHDOG.

### 15.4 DETECTION OF ILLEGAL CONDITIONS

The device can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeroes. The opcode for software interrupt is 00. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), and all other segments (i.e., Segments 4 ... etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex.

This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

1. Executing from undefined ROM
2. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

## 16.0 MICROWIRE/PLUS

MICROWIRE/PLUS is a serial SPI compatible synchronous communications interface. The MICROWIRE/PLUS capability enables the device to interface with MICROWIRE/PLUS or SPI peripherals (i.e. A/D converters, display drivers, EEPROMs etc.) and with other microcontrollers which support the MICROWIRE/PLUS or SPI interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 29* shows a block diagram of the MICROWIRE/PLUS logic.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. *Table 18* details the different clock rates that may be selected.

**TABLE 18. MICROWIRE/PLUS  
Master Mode Clock Select**

SL1	SL0	SK Period
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where  $t_C$  is the instruction cycle clock

### 16.1 MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 29* shows how two microcontroller devices and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

#### WARNING

The SIO register should only be loaded when the SK clock is in the idle phase. Loading the SIO register while the SK clock is in the active phase, will result in undefined data in the SIO register.

Setting the BUSY flag when the input SK clock is in the active phase while in the MICROWIRE/PLUS is in the slave

## 16.0 MICROWIRE/PLUS (Continued)

mode may cause the current SK clock for the SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is in the idle phase.

### 16.2 MICROWIRE/PLUS MASTER MODE OPERATION

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally. The MICROWIRE

Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. In the slave mode, the shift clock stops after 8 clock pulses. *Table 19* summarizes the bit settings required for Master mode of operation.

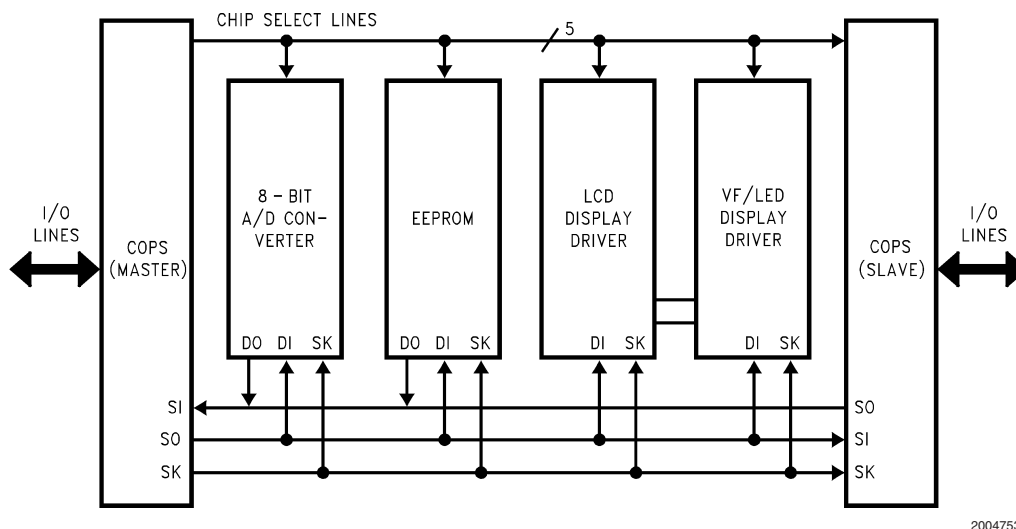


FIGURE 29. MICROWIRE/PLUS Application

### 16.3 MICROWIRE/PLUS SLAVE MODE OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bits in the Port G configuration register. *Table 19* summarizes the settings required to enter the Slave mode of operation.

This table assumes that the control flag MSEL is set.

TABLE 19. MICROWIRE/PLUS Mode Settings

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI- STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI- STATE	Ext. SK	MICROWIRE/PLUS Slave

The user must set the BUSY flag immediately upon entering the Slave mode. This ensures that all data bits sent by the

Master is shifted properly. After eight clock pulses the BUSY flag is clear, the shift clock is stopped, and the sequence may be repeated.

### 16.4 ALTERNATE SK PHASE OPERATION AND SK IDLE POLARITY

The device allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK idle polarity can be either high or low. The polarity is selected by bit 5 of Port G data register. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock. Bit 6 of Port G configuration register selects the SK edge.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.



## 16.0 MICROWIRE/PLUS (Continued)

TABLE 20. MICROWIRE/PLUS Shift Clock Polarity and Sample/Shift Phase

SK Phase	Port G		SO Clocked Out On:	SI Sampled On:	SK Idle Phase
	G6 (SKSEL) Config. Bit	G5 Data Bit			
Normal	0	0	SK Falling Edge	SK Rising Edge	Low
Alternate	1	0	SK Rising Edge	SK Falling Edge	Low
Alternate	0	1	SK Rising Edge	SK Falling Edge	High
Normal	1	1	SK Falling Edge	SK Rising Edge	High

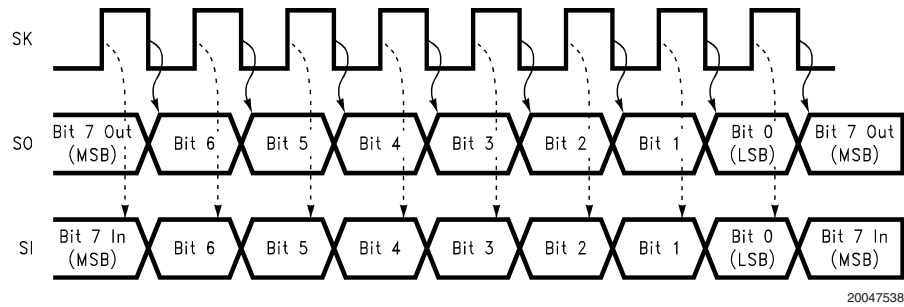


FIGURE 30. MICROWIRE/PLUS SPI Mode Interface Timing, Normal SK Mode, SK Idle Phase being Low

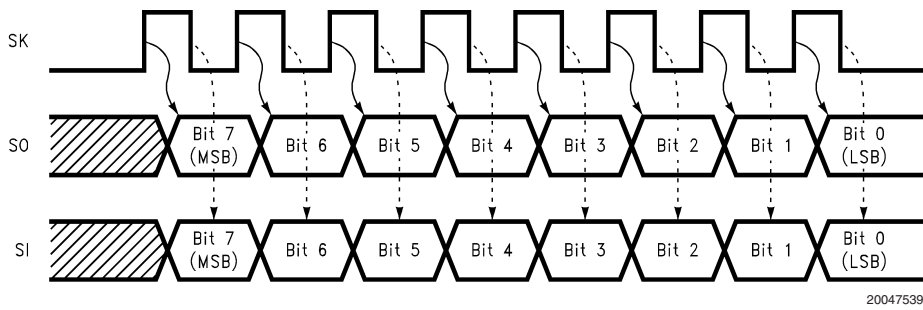


FIGURE 31. MICROWIRE/PLUS SPI Mode Interface Timing, Alternate SK Mode, SK Idle Phase being Low

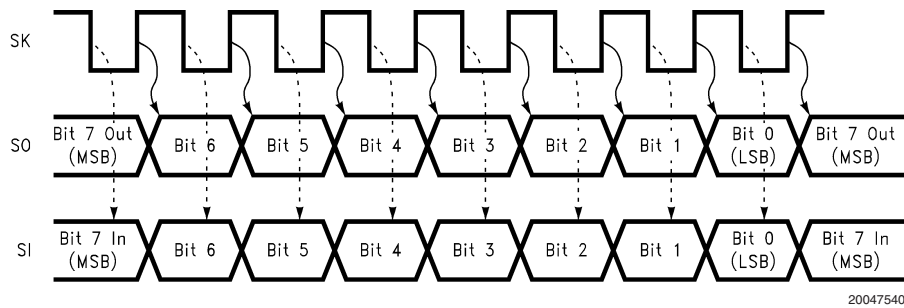
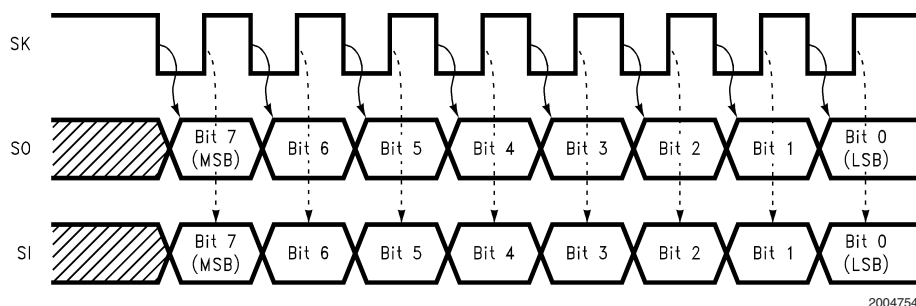


FIGURE 32. MICROWIRE/PLUS SPI Mode Interface Timing, Normal SK Mode, SK Idle Phase being High

## 16.0 MICROWIRE/PLUS (Continued)



**FIGURE 33. MICROWIRE/PLUS SPI Mode Interface Timing, Alternate SK Mode, SK Idle Phase being High**

## 17.0 ACCESS.Bus Interface

The ACCESS.Bus interface module (ACB) is a two-wire serial interface compatible with the ACCESS.Bus physical layer. It permits easy interfacing to a wide range of low-cost memories and I/O devices, including: EEPROMs, SRAMs, timers, A/D converters, D/A converters, clock chips, and peripheral drivers. It is compatible with Intel's SMBus and Philips' I<sup>2</sup>C bus. The module can be configured as a bus master or slave, and can maintain bidirectional communications with both multiple master and multiple slave devices.

- ACCESS.Bus master and slave
- Supports polling and interrupt-controlled operation
- Generate a wake-up signal on detection of a Start Condition, while in reduced-power mode
- Optional internal pull-up on SDA and SCL pins
- Optional 1.8V logic compatibility on SDA and SCL pins

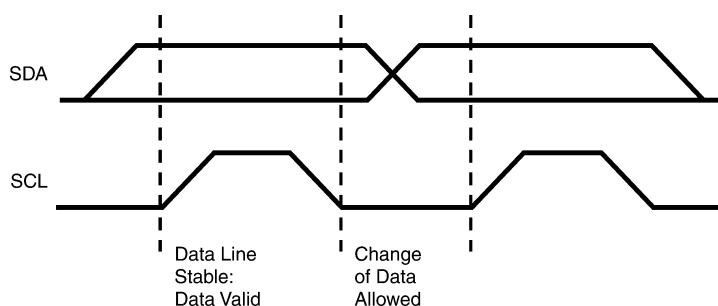
The ACCESS.Bus protocol uses a two-wire interface for bidirectional communication between the devices connected to the bus. The two interface signals are the Serial Data Line (SDA) and the Serial Clock Line (SCL). These signals should be connected to the positive supply, through pull-up resistors, to keep the signals high when the bus is idle. When the ACCESS.Bus module is enabled and Bit 7 of the Option Register (LVCMP) is set, the SDA and SCL inputs, along with input L2, provide compatibility with 1.8V logic levels.

The ACCESS.Bus protocol supports multiple master and slave transmitters and receivers. Each bus device has a unique address and can operate as a transmitter or a receiver (though some peripherals are only receivers).

### 17.1 DATA TRANSACTIONS

During data transactions, the master device initiates the transaction, generates the clock signal and terminates the transaction. For example, when the ACB initiates a data transaction with an ACCESS.Bus peripheral, the ACB becomes the master. When the peripheral responds and transmits data to the ACB, their master/slave (data transaction initiator and clock generator) relationship is unchanged, even though their transmitter/receiver functions are reversed.

One data bit is transferred during each clock period. Data is sampled during the high phase of the serial clock (SCL). Consequently, throughout the clock high phase, the data must remain stable (see Figure 34). Any change on the SDA signal during the high phase of the SCL clock and in the middle of a transaction aborts the current transaction. New data must be driven during the low phase of the SCL clock. This protocol permits a single data line to transfer both command/control information and data using the synchronous serial clock.



**FIGURE 34. Bit Transfer**

Each data transaction is composed of a Start Condition, a number of byte transfers (programmed by software) and a

Stop Condition to terminate the transaction. Each byte is transferred with the most significant bit first, and after each byte, an Acknowledge signal must follow.

## 17.0 ACCESS.Bus Interface

(Continued)

At each clock cycle, the slave can stall the master while it handles the previous data, or prepares new data. The slave can hold SCL low, to extend the clock-low period, on each bit transfer, or on a byte boundary, to accomplish this. Typically, slaves extend the first clock cycle of a transfer if a byte read has not yet been stored, or if the next byte to be transmitted is not yet ready. Some microcontrollers, with limited hardware support for ACCESS.Bus, extend the access after each bit, to allow software time to handle this bit.

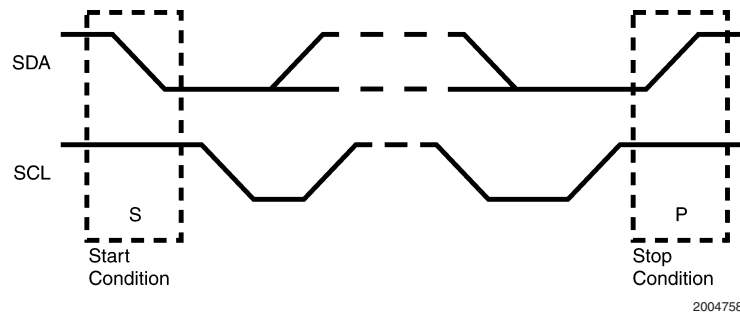


FIGURE 35. Start and Stop Conditions

In addition to the first Start Condition, a repeated Start Condition can be generated in the middle of a transaction. This allows another device to be accessed, or a change in the direction of the data transfer.

### 17.1.2 Acknowledge Cycle

The Acknowledge Cycle consists of two signals: the acknowledge clock pulse the master sends with each byte transferred, and the acknowledge signal sent by the receiving device.

The master generates an acknowledge clock pulse after each byte transfer. The receiver sends an acknowledge signal after every byte received. There are two exceptions to the "acknowledge after every byte" rule.

- When the master is the receiver, it must indicate to the transmitter an end-of-data condition by not-acknowledging ("negative acknowledge") the last byte clocked out of the slave. This "negative acknowledge" still includes the acknowledge clock pulse (generated by the master), but the SDA line is not pulled down.
- When the receiver is full, otherwise occupied, or a problem has occurred, it sends a negative acknowledge to indicate that it cannot accept additional data bytes.

### 17.1.3 Addressing Transfer Formats

Each device on the bus has a unique address. Before any data is transmitted, the master transmits the address of the slave being addressed. The slave device should send an acknowledge signal on the SDA signal, once it recognizes its address.

### 17.2 BUS ARBITRATION

Arbitration is required when multiple master devices attempt to gain control of the bus simultaneously. Control of the bus is initially determined according to the address bits and clock cycle. If the masters are trying to address the same bus device, data comparisons determine the outcome of this

### 17.1.1 Start and Stop

The ACCESS.Bus master generates Start and Stop Conditions (control codes). After a Start Condition is generated, the bus is considered busy and it retains this status until a certain time after a Stop Condition is generated. A high-to-low transition of the data line (SDA) while the clock (SCL) is high indicates a Start Condition. A low-to-high transition of the SDA line while the SCL is high indicates a Stop Condition (Figure 35).

arbitration. In master mode, the device immediately aborts a transaction if the value sampled on the SDA line differs from the value driven by the device.

When an abort occurs during the address transmission, the master that identifies the conflict should give up the bus, switch to slave mode, and continue to sample SDA to see if it is being addressed by the winning master on the ACCESS.Bus.

### 17.3 POWER SAVE MODES

When this device is placed in HALT or IDLE mode, the ACB module is effectively disabled. Registers ACBST, ACBCST and ACBCTL1 are reset, however ACBSDA, ACBADDR and ACBCTL2 are unaffected. If the ACB is enabled (ACBCTL2.ENABLE = 1) on detection of a Start Condition, a wake-up signal is issued to the Multi-Input Wake-Up module. The byte transfer which causes the Wake-Up event will not be acknowledged by the COP8 ACCESS.Bus and thus must be retransmitted. The Multi-Input Wake-Up logic must be configured, by the user, to enable Wake-Up on ACCESS.Bus transfer. The ACCESS.Bus SDA signal is an alternate function of the one of the Multi-Input Wake-Up pins, and thus the associated bit of the LWKEN or CWKEN and LWKEDG or CWKEDG registers must be configured to cause a Wake-Up event on a rising edge. See Figure 25 and the pinout table for determination of the Multi-Input Wake-Up channel associated with the ACCESS.Bus.

### 17.4 SDA AND SCL DRIVER CONFIGURATION

SDA and SCL are driven as open-drain signals on Port L signals L0 and L1. If the ACB interface is not being used, these pins are available for use as general-purpose port pins or Multi-Input Wake-Up inputs.

## 17.0 ACCESS.Bus Interface

(Continued)

### 17.5 ACB SERIAL DATA REGISTER (ACBSDA)

The ACBSDA register is a byte-wide, read/write shift register used to transmit and receive data. The most significant bit is transmitted (received) first and the least significant bit is transmitted (received) last.

7	0
DATA	

### 17.6 ACB STATUS REGISTER (ACBST)

The ACBST register is a byte-wide, read-only register that reports the current ACB status.

7	6	5	4	3	2	1	0
SLVSTP	SDAST	BER	NEGACK	RSVD	NMATCH	MASTER	XMIT

SLVSTP	The Slave Stop bit is set when a Stop Condition was detected after a slave transfer (i.e., after a slave transfer in which MATCH or GCMTCH is set).
SDAST	The SDA Status bit is set when the SDA data register is waiting for data (transmit, as master or slave) or holds data that should be read (receive, as master or slave)
BER	The Bus Error bit is set when a Start or Stop Condition is detected during data transfer or when an arbitration problem is detected.
NEGACK	The Negative Acknowledge bit is set when a transmission is not acknowledged.
RSVD	This bit is reserved and will be zero.
NMATCH	The New Match bit is set when the address byte following a Start Condition, or repeated starts, causes a match or a global-call match.
MASTER	The Master bit indicates that the module is currently in master mode. It is set when a request for bus mastership succeeds. It is cleared upon arbitration loss (BER is set) or the recognition of a Stop Condition.
XMIT	The Direction bit is set when the ACB module is currently in master/slave transmit mode. Otherwise, it is clear.

### 17.7 ACB CONTROL STATUS REGISTER (ACBCST)

The ACBCST register is a byte-wide, read/write register that reports the current ACB status. At reset and when the module is disabled, the non-reserved bits of ACBCST are cleared.

7	6	5	4	3	2	1	0
RSVD	TGSCL	TSDA	GCMTCH	MATCH	BB	BUSY	

**TGSCL** The Toggle SCL bit enables toggling the SCL signal during error recovery. When the SDA signal is low, writing 1 to this bit drives the SCL signal high for one cycle. Writing 1 to TGSCL when the SDA signal is high is ignored.

**TSDA** The Test SDA bit samples the state of the SDA signal. This bit can be used while recovering from an error condition in which the SDA signal is constantly pulled low by a slave that has lost synchronization. This bit is a read-only bit.

**GCMTCH** The Global Call Match bit is set in slave mode when the ACBCTL1.GCMEN bit is set and the address byte (the first byte transferred after a Start Condition) is 00.

**MATCH** The Address Match bit indicates in slave mode when ACBADDR.SAEN is set and the first seven bits of the address byte (the first byte transferred after a Start Condition) matches the 7-bit address in the ACBADDR register.

**BB** The Bus Busy bit indicates the bus is busy. It is set when the bus is active (i.e., a low level on either SDA or SCL) or by a Start Condition. It is cleared when the module is disabled or a Stop Condition is detected.

**BUSY** The BUSY bit indicates that the ACB module is:

- Generating a Start Condition
- In Master mode (ACBST.MASTER is set)
- In Slave mode (ACBCST.MATCH or ACBCST.GCMTCH is set)
- In the period between detecting a Start and completing the reception of the address byte. After this, the ACB either becomes not busy or enters slave mode. The BUSY bit is cleared by the completion of any of the above states, or by disabling the module. BUSY is a read only bit.

### 17.8 ACB CONTROL 1 REGISTER (ACBCTL1)

The ACBCTL1 register is a byte-wide, read/write register that configures and controls the ACB module. At reset and while the module is disabled (ACBCTL2.ENABLE = 0), the ACBCTL1 register is cleared.

7	6	5	4	3	2	1	0
CLRST	NMINTE	GCMEN	ACK	RSVD	INTEN	STOP	START

**CLRST** The Clear Status bit clears the NMATCH, BER, NEGACK and SLVSTP bits when 1 is written to this bit.

**NMINTE** The New Match Interrupt Enable controls whether ACB interrupts are generated on new matches.

## 17.0 ACCESS.Bus Interface

(Continued)

GCMEN	The Global Call Match Enable bit enables the match of an incoming address byte to the general call address (Start Condition followed by address byte of 00) while the ACB is in slave mode.
ACK	The Acknowledge bit holds the value this device sends in master or slave mode during the next acknowledge cycle. Setting this bit to 1 instructs the transmitting device to stop sending data, because the receiver either does not need, or cannot receive, any more data.
INTEN	The Interrupt Enable bit controls generating ACB interrupts. When the INTEN bit is set, interrupts are enabled. An interrupt is generated on any of the following events: <ul style="list-style-type: none"> <li>An address MATCH is detected (ACBST.NMATCH = 1) and the NMINTEN bit is set.</li> <li>A Bus Error occurs (ACBST.BERR = 1).</li> <li>Negative acknowledge after sending a byte (ACBST.NEGACK = 1).</li> <li>An interrupt is generated on acknowledge of each transaction (same as hardware setting the ACBST.SDAST bit).</li> <li>Detection of a Stop Condition while in slave receive mode (ACBST.SLVSTP = 1).</li> </ul>
STOP	The Stop bit in master mode generates a Stop Condition that completes or aborts the current message transfer.
START	The Start bit is set to generate a Start Condition on the ACCESS.Bus. This bit should be set only when in Master mode or when requesting Master mode. An address send sequence should then be performed.

### 17.9 ACB CONTROL REGISTER 2 (ACBCTL2)

The ACBCTL2 register is a byte-wide, read/write register that controls the module and selects the ACB clock rate. At reset, the ACBCTL2 register is cleared.

7	1	0
SCLFRQ		ENABLE

SCLFRQ	The SCL Frequency field specifies the SCL period (low time and high time) in master mode. The clock low time and high time are defined as follows: $t_{SCLK1} = t_{SCLKh} = 2 \times SCLFRQ \times t_{SCLK}$ Where $t_{CLK}$ is this device's clock period when in Active mode. The SCLFRQ field may be programmed to values in the range of 0001000 through 1111111.
--------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ENABLE

The Enable bit controls the ACB module. When this bit is set, the ACB module is enabled. When the Enable bit is clear, the ACB module is disabled, the ACBCTL1, ACBST, and ACBCST registers are cleared, and the ACB module clocks are halted.

### 17.10 ACB OWN ADDRESS REGISTER (ACBADDR)

The ACBADDR register is a byte-wide, read/write register that holds the module's first ACCESS.Bus address.

7	6	0
SEAN	ADDR	

SAEN

The Slave Address Enable bit controls whether address matching is performed in slave mode. When set, the SAEN bit indicates that the ADDR field holds a valid address and enables the match of ADDR to an incoming address byte.

ADDR

The Own Address field holds the 7-bit ACCESS.Bus address of this device. In slave mode, the 7 bits received after a Start Condition are compared to this field (first bit received to bit 6, and the last to bit 0). If the address field matches the received data and the SAEN bit is set, a match is detected.

## 18.0 Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address ADD REG	Contents
00 to 6F	On-Chip RAM bytes (112 bytes)
70 to 7F	Unused RAM Address Space (Reads As All Ones)
80 to 83	Unused RAM Address Space (Reads Undefined Data)
84	Port C MIWU Edge Select Register (Reg: CWKEDG)
85	Port C MIWU Enable Register (Reg: CWKEN)
86	Port C MIWU Pending Register (Reg: CWKPND)
87 to 8F	Reserved
90 to 93	Reserved
94	Port F Data Register
95	Port F Configuration Register
96	Port F Input pins (Read Only)
97	Reserved for Port F
98 to 9F	Reserved
A0 to A7	Reserved

## 18.0 Memory Map (Continued)

Address ADD REG	Contents
A8	ISP Address Register Low Byte (ISPADLO)
A9	ISP Address Register High Byte (ISPADHI)
AA	ISP Read Data Register (ISPRD)
AB	ISP Write Data Register (ISPWR)
AC to AF	Reserved
B0 to B7	Reserved
B8	ACB Serial Data Register (ACBSDA)
B9	ACB Status Register (ACBST)
BA	ACB Control And Status (ACBCST)
BB	ACB Control Register 1 (ACBCTL1)
BC	ACB Own Address Register (ACBADDR)
BD	ACB Control Register 2(ACBCTL2)
BE to BF	Reserved
C0 to C6	Reserved
C7	WATCHDOG Service Register (Reg:WDSVR)
C8	Port L MIWU Edge Select Register (Reg:LWKEDG)
C9	Port L MIWU Enable Register (Reg:LWKEN)
CA	Port L MIWU Pending Register (Reg:LWKPND)
CB to CE	Reserved
CF	Idle Timer Control Register (ITMR)
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Reserved
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved
DC	Port J Data Register
DD	Port J Configuration Register
DE	Port J Input Pins (Read Only)
DF	CPU Clock Prescale Register (CLKPS)
E0	Reserved
E1	Flash Memory Write Timing Register (PGMTIM)
E2	ISP Key Register (ISPKEY)
E3 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte

Address ADD REG	Contents
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE/PLUS Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	On-Chip RAM Mapped as Register

**Note:** Reading memory locations 70H–7FH will return all ones. Reading unused memory locations 80H–83H, 87H–93H will return undefined data.

## 19.0 Instruction Set

### 19.1 INTRODUCTION

This section defines the instruction set of the COP8 Family members. It contains information about the instruction set features, addressing modes and types.

### 19.2 INSTRUCTION FEATURES

The strength of the instruction set is based on the following features:

- Mostly single-byte opcode instructions minimize program size.
- One instruction cycle for the majority of single-byte instructions to minimize program execution time.
- Many single-byte, multiple function instructions such as DRSZ.
- Three memory mapped pointers: two for register indirect addressing, and one for the software stack.
- Sixteen memory mapped registers that allow an optimized implementation of certain instructions.
- Ability to set, reset, and test any individual bit in data memory address space, including the memory-mapped I/O ports and registers.
- Register-Indirect LOAD and EXCHANGE instructions with optional automatic post-incrementing or decrementing of the register pointer. This allows for greater efficiency (both in cycle time and program code) in loading, walking across and processing fields in data memory.
- Unique instructions to optimize program size and throughput efficiency. Some of these instructions are: DRSZ, IFBNE, DCOR, RETSK, VIS and RRC.

### 19.3 ADDRESSING MODES

The instruction set offers a variety of methods for specifying memory addresses. Each method is called an addressing mode. These modes are classified into two categories: op-



## 19.0 Instruction Set (Continued)

operand addressing modes and transfer-of-control addressing modes. Operand addressing modes are the various methods of specifying an address for accessing (reading or writing) data. Transfer-of-control addressing modes are used in conjunction with jump instructions to control the execution sequence of the software program.

### 19.3.1 Operand Addressing Modes

The operand of an instruction specifies what memory location is to be affected by that instruction. Several different operand addressing modes are available, allowing memory locations to be specified in a variety of ways. An instruction can specify an address directly by supplying the specific address, or indirectly by specifying a register pointer. The contents of the register (or in some cases, two registers) point to the desired memory location. In the immediate mode, the data byte to be used is contained in the instruction itself.

Each addressing mode has its own advantages and disadvantages with respect to flexibility, execution speed, and program compactness. Not all modes are available with all instructions. The Load (LD) instruction offers the largest number of addressing modes.

The available addressing modes are:

- Direct
- Register B or X Indirect
- Register B or X Indirect with Post-Incrementing/Decrementing
- Immediate
- Immediate Short
- Indirect from Program Memory

The addressing modes are described below. Each description includes an example of an assembly language instruction using the described addressing mode.

**Direct.** The memory address is specified directly as a byte in the instruction. In assembly language, the direct address is written as a numerical value (or a label that has been defined elsewhere in the program as a numerical value).

Example: Load Accumulator Memory Direct

LD A,#05

Reg/Data	Contents	Contents
Memory	Before	After
Accumulator	XX Hex	A6 Hex
Memory Location 0005 Hex	A6 Hex	A6 Hex

**Register B or X Indirect.** The memory address is specified by the contents of the B Register or X register (pointer register). In assembly language, the notation [B] or [X] specifies which register serves as the pointer.

Example: Exchange Memory with Accumulator, B Indirect

X A,[B]

Reg/Data	Contents	Contents
Memory	Before	After
Accumulator	01 Hex	87 Hex
Memory Location 0005 Hex	87 Hex	01 Hex
B Pointer	05 Hex	05 Hex

**Register B or X Indirect with Post-Incrementing/Decrementing.** The relevant memory address is specified by the contents of the B Register or X register (pointer register). The pointer register is automatically incremented or decremented after execution, allowing easy manipulation of memory blocks with software loops. In assembly language, the notation [B+], [B-], [X+], or [X-] specifies which register serves as the pointer, and whether the pointer is to be incremented or decremented.

Example: Exchange Memory with Accumulator, B Indirect with Post-Increment

X A,[B+]

Reg/Data	Contents	Contents
Memory	Before	After
Accumulator	03 Hex	62 Hex
Memory Location 0005 Hex	62 Hex	03 Hex
B Pointer	05 Hex	06 Hex

**Intermediate.** The data for the operation follows the instruction opcode in program memory. In assembly language, the number sign character (#) indicates an immediate operand.

Example: Load Accumulator Immediate

LD A,#05

Reg/Data	Contents	Contents
Memory	Before	After
Accumulator	XX Hex	05 Hex

**Immediate Short.** This is a special case of an immediate instruction. In the "Load B immediate" instruction, the 4-bit immediate value in the instruction is loaded into the lower nibble of the B register. The upper nibble of the B register is reset to 0000 binary.

Example: Load B Register Immediate Short

LD B,#7

Reg/Data	Contents	Contents
Memory	Before	After
B Pointer	12 Hex	07 Hex

**Indirect from Program Memory.** This is a special case of an indirect instruction that allows access to data tables stored in program memory. In the "Load Accumulator Indirect" (LAID) instruction, the upper and lower bytes of the Program Counter (PCU and PCL) are used temporarily as a pointer to program memory. For purposes of accessing program memory, the contents of the Accumulator and PCL are exchanged. The data pointed to by the Program Counter is loaded into the Accumulator, and simultaneously, the original contents of PCL are restored so that the program can resume normal execution.

Example: Load Accumulator Indirect

LAID

Reg/Data	Contents	Contents
Memory	Before	After
PCU	04 Hex	04 Hex
PCL	35 Hex	36 Hex
Accumulator	1F Hex	25 Hex
Memory Location 041F Hex	25 Hex	25 Hex



## 19.0 Instruction Set (Continued)

### 19.3.2 Transfer-of-Control Addressing Modes

Program instructions are usually executed in sequential order. However, Jump instructions can be used to change the normal execution sequence. Several transfer-of-control addressing modes are available to specify jump addresses.

A change in program flow requires a non-incremental change in the Program Counter contents. The Program Counter consists of two bytes, designated the upper byte (PCU) and lower byte (PCL). The most significant bit of PCU is not used, leaving 15 bits to address the program memory.

Different addressing modes are used to specify the new address for the Program Counter. The choice of addressing mode depends primarily on the distance of the jump. Farther jumps sometimes require more instruction bytes in order to completely specify the new Program Counter contents.

The available transfer-of-control addressing modes are:

- Jump Relative
- Jump Absolute
- Jump Absolute Long
- Jump Indirect

The transfer-of-control addressing modes are described below. Each description includes an example of a Jump instruction using a particular addressing mode, and the effect on the Program Counter bytes of executing that instruction.

**Jump Relative.** In this 1-byte instruction, six bits of the instruction opcode specify the distance of the jump from the current program memory location. The distance of the jump can range from -31 to +32. A JP+1 instruction is not allowed. The programmer should use a NOP instead.

Example: Jump Relative  
JP 0A

Reg	Contents Before	Contents After
PCU	02 Hex	02 Hex
PCL	05 Hex	0F Hex

**Jump Absolute.** In this 2-byte instruction, 12 bits of the instruction opcode specify the new contents of the Program Counter. The upper three bits of the Program Counter remain unchanged, restricting the new Program Counter address to the same 4-kbyte address space as the current instruction. (This restriction is relevant only in devices using more than one 4-kbyte program memory space.)

Example: Jump Absolute  
JMP 0125

Reg	Contents Before	Contents After
PCU	0C Hex	01 Hex
PCL	77 Hex	25 Hex

**Jump Absolute Long.** In this 3-byte instruction, 15 bits of the instruction opcode specify the new contents of the Program Counter.

Example: Jump Absolute Long  
JMP 03625

Reg/ Memory	Contents Before	Contents After
PCU	42 Hex	36 Hex
PCL	36 Hex	25 Hex

**Jump Indirect.** In this 1-byte instruction, the lower byte of the jump address is obtained from a table stored in program memory, with the Accumulator serving as the low order byte of a pointer into program memory. For purposes of accessing program memory, the contents of the Accumulator are written to PCL (temporarily). The data pointed to by the Program Counter (PCH/PCL) is loaded into PCL, while PCH remains unchanged.

Example: Jump Indirect  
JID

Reg/ Memory	Contents Before	Contents After
PCU	01 Hex	01 Hex
PCL	C4 Hex	32 Hex
Accumulator	26 Hex	26 Hex
Memory Location	32 Hex	32 Hex
0126 Hex		

The VIS instruction is a special case of the Indirect Transfer of Control addressing mode, where the double-byte vector associated with the interrupt is transferred from adjacent addresses in program memory into the Program Counter in order to jump to the associated interrupt service routine.

### 19.4 INSTRUCTION TYPES

The instruction set contains a wide variety of instructions. The available instructions are listed below, organized into related groups.

Some instructions test a condition and skip the next instruction if the condition is not true. Skipped instructions are executed as no-operation (NOP) instructions.

#### 19.4.1 Arithmetic Instructions

The arithmetic instructions perform binary arithmetic such as addition and subtraction, with or without the Carry bit.

Add (ADD)  
Add with Carry (ADC)  
Subtract with Carry (SUBC)  
Increment (INC)  
Decrement (DEC)  
Decimal Correct (DCOR)  
Clear Accumulator (CLR)  
Set Carry (SC)  
Reset Carry (RC)

#### 19.4.2 Transfer-of-Control Instructions

The transfer-of-control instructions change the usual sequential program flow by altering the contents of the Program Counter. The Jump to Subroutine instructions save the Program Counter contents on the stack before jumping; the Return instructions pop the top of the stack back into the Program Counter.

Jump Relative (JP)  
Jump Absolute (JMP)

## 19.0 Instruction Set (Continued)

Jump Absolute Long (JMPL)  
 Jump Indirect (JID)  
 Jump to Subroutine (JSR)  
 Jump to Subroutine Long (JSRL)  
 Jump to Boot ROM Subroutine (JSRB)  
 Return from Subroutine (RET)  
 Return from Subroutine and Skip (RETSK)  
 Return from Interrupt (RETI)  
 Software Trap Interrupt (INTR)  
 Vector Interrupt Select (VIS)

### 19.4.3 Load and Exchange Instructions

The load and exchange instructions write byte values in registers or memory. The addressing mode determines the source of the data.

Load (LD)  
 Load Accumulator Indirect (LAID)  
 Exchange (X)

### 19.4.4 Logical Instructions

The logical instructions perform the operations AND, OR, and XOR (Exclusive OR). Other logical operations can be performed by combining these basic operations. For example, complementing is accomplished by exclusive-ORing the Accumulator with FF Hex.

Logical AND (AND)  
 Logical OR (OR)  
 Exclusive OR (XOR)

### 19.4.5 Accumulator Bit Manipulation Instructions

The Accumulator bit manipulation instructions allow the user to shift the Accumulator bits and to swap its two nibbles.

Rotate Right Through Carry (RRC)  
 Rotate Left Through Carry (RLC)  
 Swap Nibbles of Accumulator (SWAP)

### 19.4.6 Stack Control Instructions

Push Data onto Stack (PUSH)  
 Pop Data off of Stack (POP)

### 19.4.7 Memory Bit Manipulation Instructions

The memory bit manipulation instructions allow the user to set and reset individual bits in memory.

Set Bit (SBIT)  
 Reset Bit (RBIT)  
 Reset Pending Bit (RPND)

### 19.4.8 Conditional Instructions

The conditional instruction test a condition. If the condition is true, the next instruction is executed in the normal manner; if the condition is false, the next instruction is skipped.

If Equal (IFEQ)  
 If Not Equal (IFNE)  
 If Greater Than (IFGT)

If Carry (IFC)  
 If Not Carry (IFNC)  
 If Bit (IFBIT)  
 If B Pointer Not Equal (IFBNE)  
 And Skip if Zero (ANDSZ)  
 Decrement Register and Skip if Zero (DRSZ)

### 19.4.9 No-Operation Instruction

The no-operation instruction does nothing, except to occupy space in the program memory and time in execution.

No-Operation (NOP)

**Note:** The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

## 19.5 REGISTER AND SYMBOL DEFINITION

The following abbreviations represent the nomenclature used in the instruction description and the COP8 cross-assembler.

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
S	8-Bit Segment Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

## 19.0 Instruction Set (Continued)

### 19.6 INSTRUCTION SET SUMMARY

ADD	A,Meml	ADD	$A \leftarrow A + \text{Meml}$
ADC	A,Meml	ADD with Carry	$A \leftarrow A + \text{Meml} + C$ , $C \leftarrow \text{Carry}$ , $HC \leftarrow \text{Half Carry}$
SUBC	A,Meml	Subtract with Carry	$A \leftarrow A - \overline{\text{Meml}} + C$ , $C \leftarrow \text{Carry}$ , $HC \leftarrow \text{Half Carry}$
AND	A,Meml	Logical AND	$A \leftarrow A \text{ and } \overline{\text{Meml}}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if $(A \text{ and } \text{Imm}) = 0$
OR	A,Meml	Logical OR	$A \leftarrow A \text{ or } \text{Meml}$
XOR	A,Meml	Logical EXclusive OR	$A \leftarrow A \text{ xor } \text{Meml}$
IFEQ	MD,Imm	IF EQUAL	Compare MD and Imm, Do next if $MD = \text{Imm}$
IFEQ	A,Meml	IF EQUAL	Compare A and Meml, Do next if $A = \text{Meml}$
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if $A \neq \text{Meml}$
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if $A > \text{Meml}$
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B $\neq$ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	$\text{Reg} \leftarrow \text{Reg} - 1$ , Skip if $\text{Reg} = 0$
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit #, A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	$A \leftrightarrow \text{Mem}$
X	A,[X]	EXchange A with Memory [X]	$A \leftrightarrow [X]$
LD	A,Meml	LoaD A with Memory	$A \leftarrow \text{Meml}$
LD	A,[X]	LoaD A with Memory [X]	$A \leftarrow [X]$
LD	B,Imm	LoaD B with Immed.	$B \leftarrow \text{Imm}$
LD	Mem,Imm	LoaD Memory Immed.	$\text{Mem} \leftarrow \text{Imm}$
LD	Reg,Imm	LoaD Register Memory Immed.	$\text{Reg} \leftarrow \text{Imm}$
X	A, [B $\pm$ ]	EXchange A with Memory [B]	$A \leftrightarrow [B]$ , $(B \leftarrow B \pm 1)$
X	A, [X $\pm$ ]	EXchange A with Memory [X]	$A \leftrightarrow [X]$ , $(X \leftarrow X \pm 1)$
LD	A, [B $\pm$ ]	LoaD A with Memory [B]	$A \leftarrow [B]$ , $(B \leftarrow B \pm 1)$
LD	A, [X $\pm$ ]	LoaD A with Memory [X]	$A \leftarrow [X]$ , $(X \leftarrow X \pm 1)$
LD	[B $\pm$ ],Imm	LoaD Memory [B] Immed.	$[B] \leftarrow \text{Imm}$ , $(B \leftarrow B \pm 1)$
CLR	A	CLeaR A	$A \leftarrow 0$
INC	A	INCrement A	$A \leftarrow A + 1$
DEC	A	DECrement A	$A \leftarrow A - 1$
LAID		Load A InDirect from ROM	$A \leftarrow \text{ROM}(\text{PU}, A)$
DCOR	A	Decimal CORrect A	$A \leftarrow \text{BCD correction of A (follows ADC, SUBC)}$
RRC	A	Rotate A Right thru C	$C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$
RLC	A	Rotate A Left thru C	$C \leftarrow A7 \leftarrow \dots \leftarrow A0 \leftarrow C$ , $HC \leftarrow A0$
SWAP	A	SWAP nibbles of A	$A7 \dots A4 \leftrightarrow A3 \dots A0$
SC		Set C	$C \leftarrow 1$ , $HC \leftarrow 1$
RC		Reset C	$C \leftarrow 0$ , $HC \leftarrow 0$
IFC		IF C	If C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	$\text{SP} \leftarrow \text{SP} + 1$ , $A \leftarrow [\text{SP}]$
PUSH	A	PUSH A onto the stack	$[\text{SP}] \leftarrow A$ , $\text{SP} \leftarrow \text{SP} - 1$
VIS		Vector to Interrupt Service Routine	$\text{PU} \leftarrow [\text{VU}]$ , $\text{PL} \leftarrow [\text{VL}]$
JMPL	Addr.	Jump absolute Long	$\text{PC} \leftarrow ii$ (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	$\text{PC}9 \dots 0 \leftarrow i$ (i = 12 bits)
JP	Disp.	Jump relative short	$\text{PC} \leftarrow \text{PC} + r$ (r is -31 to +32, except 1)

## 19.0 Instruction Set (Continued)

JSRL	Addr.	Jump SubRoutine Long	$[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow ii$
JSR	Addr.	Jump SubRoutine	$[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC9...0 \leftarrow i$
JSRB	Addr	Jump SubRoutine Boot ROM	$[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2,$ $PL \leftarrow Addr, PU \leftarrow 00, \text{switch to flash}$
JID		Jump InDirect	$PL \leftarrow ROM (PU, A)$
RET		RETurn from subroutine	$SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1]$
RETSK		RETurn and SKip	$SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1],$ skip next instruction
RETI		RETurn from Interrupt	$SP + 2, PL \leftarrow [SP], PU \leftarrow [SP-1], GIE \leftarrow 1$
INTR		Generate an Interrupt	$[SP] \leftarrow PL, [SP-1] \leftarrow PU, SP-2, PC \leftarrow 0FF$
NOP		No OPeration	$PC \leftarrow PC + 1$

### 19.7 INSTRUCTION EXECUTION TIME

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time to execute.

Skipped instructions require x number of cycles to be skipped, where x equals the number of bytes in the skipped instruction opcode.

See the BYTES and CYCLES per INSTRUCTION table for details.

#### Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle.

#### Arithmetic and Logic Instructions

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1
------	-----

#### Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

#### Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JSRB	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

## 19.0 Instruction Set (Continued)

### Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B-]	[X+, X-]	
X A, (Note 8)	1/1	1/3	2/3		1/2	1/3	
LD A, (Note 8)	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/2			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm			2/3				
IFEQ MD,Imm			3/3				

**Note 8:** = > Memory location addressed by B or X or directly.

## 19.0 Instruction Set (Continued)

OPCODE TABLE

Upper Nibble															
F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
JP-15	JP-31	LD 0F0,#i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	IFBIT 0,[B]	ANDSZ A,#i	LD B,#0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP+17	INTR 0
JP-14	JP-30	LD 0F1,#i	DRSZ 0F1	*	SC	SUBC A,#i	SUBC A,[B]	IFBIT 1,[B]	JSRB B,#0E	LD B,#0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP+18	JP+2 1
JP-13	JP-29	LD 0F2,#i	DRSZ 0F2	X A,[X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	IFBIT 2,[B]	Re-served	LD B,#0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP+19	JP+3 2
JP-12	JP-28	LD 0F3,#i	DRSZ 0F3	X A,[X-]	X A,[B-]	IFGT A,#i	IFGT A,[B]	IFBIT 3,[B]	Re-served	LD B,#0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP+20	JP+4 3
JP-11	JP-27	LD 0F4,#i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP+21	JP+5 4
JP-10	JP-26	LD 0F5,#i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP+22	JP+6 5
JP-9	JP-25	LD 0F6,#i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP+23	JP+7 6
JP-8	JP-24	LD 0F7,#i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP+24	JP+8 7
JP-7	JP-23	LD 0F8,#i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP+25	JP+9 8
JP-6	JP-22	LD 0F9,#i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP+26	JP+10 9
JP-5	JP-21	LD 0FA,#i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	[B+],#i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP+27	JP+11 A
JP-4	JP-20	LD 0FB,#i	DRSZ 0FB	LD A,[X-]	LD A,[B-]	[B-],#i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP+28	JP+12 B
JP-3	JP-19	LD 0FC,#i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP+29	JP+13 C
JP-2	JP-18	LD 0FD,#i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP+30	JP+14 D
JP-1	JP-17	LD 0FE,#i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP+31	JP+15 E
JP-0	JP-16	LD 0FF,#i	DRSZ 0FF	*	*	LD B,#i	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP+32	JP+16 F

Lower Nibble

\* is an unused opcode

i is the immediate data

Md is a directly addressed memory location

The opcode 60 Hex is also the opcode for IFBIT #i,A



## 20.0 Development Support

### 20.1 TOOLS ORDERING NUMBERS FOR THE COP8TA 2.5V FLASH FAMILY DEVICES

This section provides specific tools ordering information for the devices in this datasheet, followed by a summary of the tools and development kits available at print time. Up-to-date information, device selection guides, demos, updates, and purchase information can be obtained at our web site at: [www.national.com/cop8](http://www.national.com/cop8).

**Unless otherwise noted, tools can be purchased for worldwide delivery from National's e-store:** <http://www.national.com/store/>

Tool	Order Number	Cost*	Notes/Includes
<b>Evaluation Software and Reference Designs</b>			
Software and Utilities	Web Downloads: <a href="http://www.national.com/cop8">www.national.com/cop8</a>	Free	<b>Assembler/ Linker/ Simulators/ Library Manager/ Compiler Demos/ Flash ISP and NiceMon Debugger Utilities/ Example Code/ etc.</b> (Flash Emulator support requires licensed COP8-NSDEV CD-ROM).
Hardware Reference Designs	None		
<b>Starter Kits and Hardware Target Boards</b>			
Starter Development Kits	COP8-DB-TAC	L	<b>Supports COP8TA</b> - Target board with 44LLP and 28SOIC sockets, LEDs, Test Points, and Breadboard Area. Development CD, ISP Cable and Source Code. No p/s. Also supports COP8 Low Voltage Flash Emulator and Kanda ISP Tool.
<b>Software Development Languages, and Integrated Development Environments</b>			
National's WCOP8 IDE and Assembler on CD	COP8-NSDEV		<b>Fully Licensed IDE with Assembler and Emulator/Debugger Support.</b> Assembler/ Linker/ Simulator/ Utilities/ Documentation. Updates from web. <b>Included with COP8-DB-TAC, SKFlash, COP8 Emulators.</b>
COP8 Library Manager from KKD	<a href="http://www.kkd.dk/libman.htm">www.kkd.dk/libman.htm</a>	Eval	<b>The ultimate information source for COP8 developers</b> - Integrates with WCOP8 IDE. Organize and manage code, notes, datasheets, etc.
<b>Hardware Emulation and Debug Tools</b>			
Hardware Emulators	COP8-IMFlash-LV	M	Includes 110v/220v p/s, target cable with 2x7 connector, manuals and software on CD.
<b>Development and Production Programming Tools</b>			
Programming Adapters (For any programmer supporting flash adapter base pinout)	COP8-PGMA-20SF2	L	For programming 20SOIC COP8TA only.
	COP8-PGMA-28SF2	L	For programming 28SOIC COP8TA only.
	COP8-PGMA-44CF2	L	For programming 44LLP COP8TAC only.
KANDA's Flash ISP Programmer	COP8 USB ISP <a href="http://www.kanda.com">www.kanda.com</a>	L	USB connected Dongle, with target cable and Control Software; Updateable from the web; <b>Purchase from <a href="http://www.kanda.com">www.kanda.com</a></b>
Development Devices	COP8TAB9 COP8TAC9	Free	All packages. Obtain samples from: <a href="http://www.national.com">www.national.com</a>
<b>*Cost: Free; VL=&lt;\$100; L=\$100-\$300; M=\$300-\$1k; H=\$1k-\$3k; VH=\$3k-\$5k</b>			

## 20.0 Development Support (Continued)

### 20.2 COP8 TOOLS OVERVIEW

#### COP8 Evaluation Software and Reference Designs -

**Software and Hardware for: Evaluation of COP8 Development Environments; Learning about COP8 Architecture and Features; Demonstrating Application Specific Capabilities.**

Product	Description	Source
WCOP8 IDE and Software Downloads	Software Evaluation downloads for Windows. Includes WCOP8 IDE evaluation version, Full COP8 Assembler/Linker, COP8-SIM Instruction Level Simulator or Unis Simulator, Byte Craft COP8C Compiler Demo, IAR Embedded Workbench (Assembler version), Manuals, Applications Software, and other COP8 technical information.	<a href="http://www.national.com/cop8">www.national.com/cop8</a> FREE Download

#### COP8 Starter Kits and Hardware Target Solutions -

**Hardware Kits for: In-depth Evaluation and Testing of COP8 capabilities; Developing and Testing Code; Implementing Target Design.**

Product	Description	Source
COP8TAC Starter Kits	COP8-DB-TAC - A complete Code Development Tool for 2.5V COP8Flash Families. A Windows IDE with Assembler, Simulator, and Debug Monitor, combined with a simple realtime target environment. Quickly design and simulate your code, then download to the target COP8flash device for execution and simple debugging. Includes a library of software routines, and source code. No power supply. (Add a COP8-EMFlash Emulator for advanced emulation and debugging)	NSC Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>

#### COP8 Software Development Languages and Integrated Environments -

**Integrated Software for: Project Management; Code Development; Simulation and Debug.**

Product	Description	Source
WCOP8 IDE from National on CD-ROM	National's COP8 Software Development package for Windows on CD. Fully licensed versions of our WCOP8 IDE and Emulator Debugger, with Assembler/Linker/ Simulators/ Library Manager/ Compiler Demos/ Flash ISP and NiceMon Debugger Utilities/ Example Code/ etc. Includes all COP8 datasheets and documentation. Included with most tools from National.	NSC Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
Unis Processor Expert	Processor Expert( from Unis Corporation - COP8 Code Generation and Simulation tool with Graphical and Traditional user interfaces. Automatically generates customized source code "Beans" (modules) containing working code for all on-chip features and peripherals, then integrates them into a fully functional application code design, with all documentation.	Unis, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
Byte Craft COP8C Compiler	ByteCraft COP8C- C Cross-Compiler and Code Development System. Includes BCLIDE (Integrated Development Environment) for Win32, editor, optimizing C Cross-Compiler, macro cross assembler, BC-Linker, and MetaLinktools support. (DOS/SUN versions available; Compiler is linkable under WCOP8 IDE)	ByteCraft Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
IAR Embedded Workbench	IAR EWCOP8 - ANSI C-Compiler and Embedded Workbench. A fully integrated Win32 IDE, ANSI C-Compiler, macro assembler, editor, linker, librarian, and C-Spy high-level simulator/debugger. (EWCOP8-M version includes COP8Flash Emulator support) (EWCOP8-BL version is limited to 4k code limit; no FP).	IAR Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>

#### COP8 Hardware Emulation/Debug Tools -

**Hardware Tools for: Real-time Emulation; Target Hardware Debug; Target Design Test.**

Product	Description	Source
COP8Flash Emulators - COP8-DMFlash	COP8 In-Circuit Emulator for Flash Families. Windows based development and real-time in-circuit emulation tool with 32k, trace 32k s/w breakpoints, source/symbolic debugger, and device programming. Includes COP8-NSDEV CD, Null Target, emulation cable with 2x7 connector, and power supply.	NSC Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
NiceMon Debug Monitor Utility	A simple, single-step debug monitor with one breakpoint. MICROWIRE interface.	Download from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>

## 20.0 Development Support (Continued)

Development and Production Programming Tools - Programmers for: Design Development; Hardware Test; Pre-Production; Full Production.		
Product	Description	Source
COP8 Flash Emulators	COP8 Flash Emulators include in-circuit device programming capability during development.	NSC Distributor, or Order from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
NiceMon Debugger, KANDAFIash	National's software Utilities "KANDAFIash" and "NiceMon" provide development In-System-Programming for our Flash Starter Kit, our Prototype Development Board, or any other target board with appropriate connectors.	Download from: <a href="http://www.national.com/cop8">www.national.com/cop8</a>
KANDA COP8-ISP	The COP8-ISP programmer from KANDA is available for engineering, and small volume production use. PC parallel or serial interface.	<a href="http://www.kanda.com">www.kanda.com</a>
SofTec Micro inDart COP8	The inDart COP8 programmer from SofTec is available for engineering and small volume production use. PC serial interface only.	<a href="http://www.softecmicro.com">www.softecmicro.com</a>
Third-Party Programmers	Third-party programmers and automatic handling equipment are approved for non-ISP engineering and production use.	
Factory Programming	Factory programming available for high-volume requirements and LLP production.	National Representative

### 20.3 WHERE TO GET TOOLS

Tools can be ordered directly from National, National's e-store (Worldwide delivery: <http://www.national.com/store/>) , a National Distributor, or from the tool vendor. Go to the vendor's web site for current listings of distributors.

Vendor	Home Office	Electronic Sites	Other Main Offices
Byte Craft Limited	421 King Street North Waterloo, Ontario Canada N2J 4E4 Tel: 1-(519) 888-6911 Fax: (519) 746-6751	<a href="http://www.bytecraft.com">www.bytecraft.com</a> <a href="mailto:info@bytecraft.com">info@bytecraft.com</a>	Distributors Worldwide
IAR Systems AB	PO Box 23051 S-750 23 Uppsala Sweden Tel: +46 18 16 78 00 Fax +46 18 16 78 38	<a href="http://www.iar.se">www.iar.se</a> <a href="mailto:info@iar.se">info@iar.se</a> <a href="mailto:info@iar.com">info@iar.com</a> <a href="mailto:info@iarsys.co.uk">info@iarsys.co.uk</a> <a href="mailto:info@iar.de">info@iar.de</a>	USA:: San Francisco Tel: +1-415-765-5500 Fax: +1-415-765-5503 UK: London Tel: +44 171 924 33 34 Fax: +44 171 924 53 41 Germany: Munich Tel: +49 89 470 6022 Fax: +49 89 470 956
Embedded Results Ltd.	P.O. Box 200, Aberystwyth, SY23 2WD, UK Tel/Fax: +44 (0)8707 446 807	<a href="http://www.kanda.com">www.kanda.com</a> <a href="mailto:sales@kanda.co">sales@kanda.co</a>	USA: Tel: 408-441-1300 Fax: 408-437-8970 <a href="mailto:sales@allamerican.com">sales@allamerican.com</a> <a href="http://www.allamerican.com">www.allamerican.com</a> Tel/Fax: 800-510-3609 <a href="mailto:info@ucpros.com">info@ucpros.com</a> <a href="http://www.ucpros.com">www.ucpros.com</a>
K and K Development ApS	Kaergaardsvej 42 DK-8355 Solbjerg Denmark Fax: +45-8692-8500	<a href="http://www.kkd.dk">www.kkd.dk</a> <a href="mailto:kkd@kkd.dk">kkd@kkd.dk</a>	
National Semiconductor	2900 Semiconductor Dr. Santa Clara, CA 95051 USA Tel: 1-800-272-9959 Fax: 1-800-737-7018	<a href="http://www.national.com/cop8">www.national.com/cop8</a> <a href="mailto:support@nsc.com">support@nsc.com</a> <a href="mailto:europe.support@nsc.com">europe.support@nsc.com</a>	Europe: Tel: 49(0) 180 530 8585 Fax: 49(0) 180 530 8586 Hong Kong: Distributors Worldwide

## 20.0 Development Support (Continued)

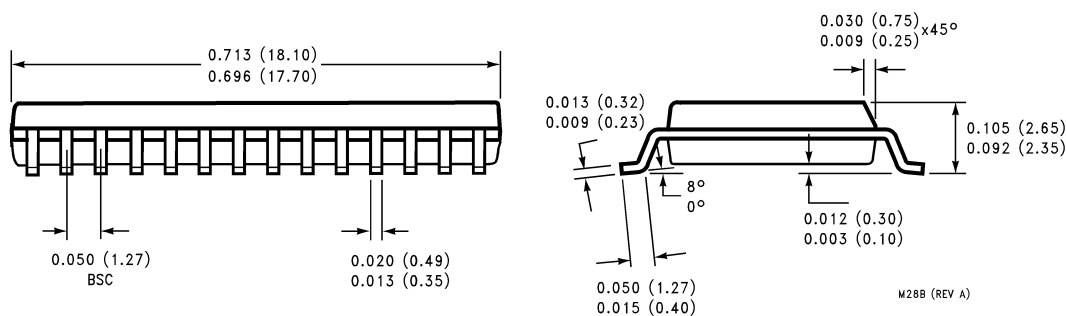
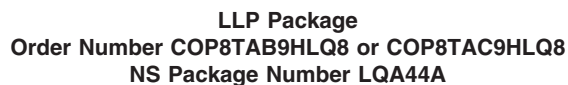
Vendor	Home Office	Electronic Sites	Other Main Offices
SofTec Microsystems	Via Roma, 1 33082 Azzano Decimo (PN) Italy Tel: +39 0434 640113 Fax: +39 0434 631598	info@softecmicro.com www.softecmicro.com support@softecmicro.com	Germany: Tel.: +49 (0) 8761 63705 France: Tel: +33 (0) 562 072 954 UK: Tel: +44 (0) 1970 621033

The following companies have approved COP8 programmers in a variety of configurations. **Contact your vendor's local office or distributor and request a COP8FLASH update.** You can link to their web sites and get the latest listing of approved programmers at: [www.national.com/cop8](http://www.national.com/cop8).

Advantech; BP Microsystems; Data I/O; Dataman; Hi-Lo Systems; KANDA, Lloyd Research; MQP; Needhams; Phytion; SofTec Microsystems; System General; and Tribal Microsystems.

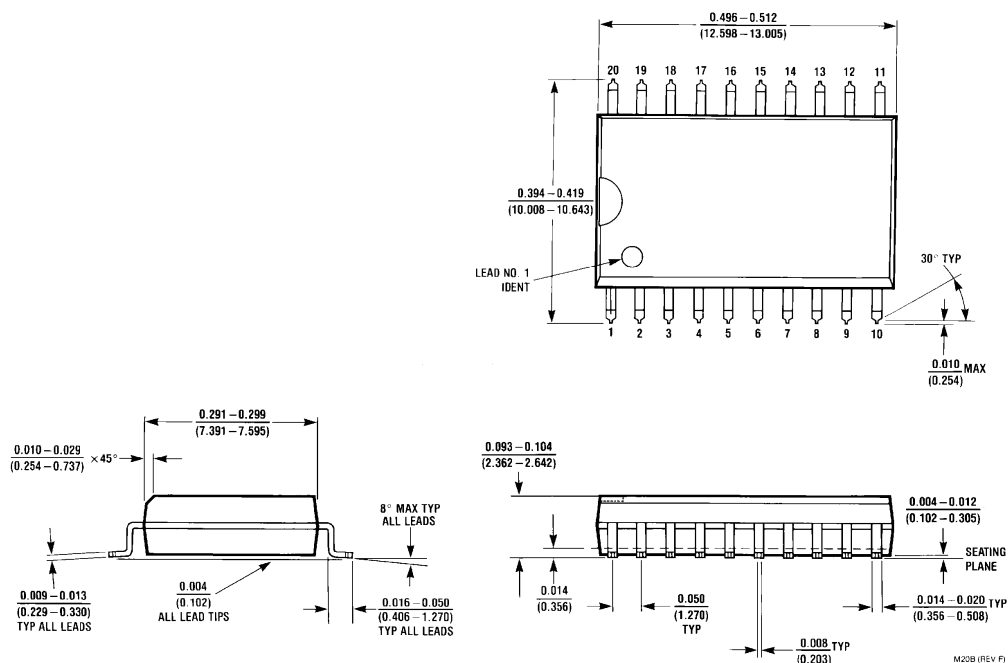
## 21.0 Revision History

Date	Section	Summary of Changes
December, 2002		Preliminary Datasheet Release.
April, 2003	User ISP and Virtual E <sup>2</sup>	Deleted
	Restrictions on Software When Calling ISP Routines in Boot ROM	Deleted
June, 2003	Multi-Input Wake-Up and ACCESS.Bus Interface	Updated to account for C Port Wake-Up.
August, 2003	Electrical Specifications, Option Register and ACCESS.Bus	Added 1.8V input compatibility on L0 (SDA), L1 (SCL) and L2. Added Electrical Specifications for ACCESS.Bus
February, 2004	Electrical Specifications, Development Support	Updated Electrical Specifications, Updated Development Support
May, 2004	Reset	Clarified need for external Reset in Brownout conditions.
	Memory Map	Modified Recommended Reset Circuit to include external Brownout detector. Added missing Port C.
February, 2005	General	Clarify differences between COP8TAx9 and COP8TAx5 to minimize problems when generating ROM code for COP8TAx5 production. General rework, through <i>Section 11.0 In-System Programming</i> , for clarity.
	Electrical Specifications	Increase Input Capacitance specification. Add Oscillator Frequency specification. Increase Typical R/C Oscillator specification. Add Figure references to ACCESS.Bus timing specifications. Correct SCL edge reference for SDA Valid Time. Reduce Halt Current Specification.
April, 2005	Functional Description	Strengthening of recommendation for external power supervisory circuit and addition of summary timing diagram for $\overline{\text{RESET}}$ operation. Addition of missing Port F in <i>Section 9.0 Pin Descriptions</i> .



**SOIC Wide Package**  
**Order Number COP8TAB9EMW8 or COP8TAC9EMW8**  
**NS Package Number M28B**

## 22.0 Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



**SOIC Wide Package**  
**Order Number COP8TAB9CMW8 or COP8TAC9CMW8**  
**NS Package Number M20B**

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

For the most current product information visit us at [www.national.com](http://www.national.com).

### LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

### BANNED SUBSTANCE COMPLIANCE

National Semiconductor manufactures products and uses packing materials that meet the provisions of the Customer Products Stewardship Specification (CSP-9-111C2) and the Banned Substances and Materials of Interest Specification (CSP-9-111S2) and contain no "Banned Substances" as defined in CSP-9-111S2.



**National Semiconductor**  
**Americas Customer**  
**Support Center**  
 Email: [new.feedback@nsc.com](mailto:new.feedback@nsc.com)  
 Tel: 1-800-272-9959

[www.national.com](http://www.national.com)

**National Semiconductor**  
**Europe Customer Support Center**  
 Fax: +49 (0) 180-530 85 86  
 Email: [europe.support@nsc.com](mailto:europe.support@nsc.com)  
 Deutsch Tel: +49 (0) 69 9508 6208  
 English Tel: +44 (0) 870 24 0 2171  
 Français Tel: +33 (0) 1 41 91 8790

**National Semiconductor**  
**Asia Pacific Customer**  
**Support Center**  
 Email: [ap.support@nsc.com](mailto:ap.support@nsc.com)

**National Semiconductor**  
**Japan Customer Support Center**  
 Fax: 81-3-5639-7507  
 Email: [jpn.feedback@nsc.com](mailto:jpn.feedback@nsc.com)  
 Tel: 81-3-5639-7560