National Semiconductor

July 1999

COP8FG Family
8-Bit CMOS ROM Based and OTP Microcontrollers with 8k to 32k Memory, Two Comparators and USART

General Description

Note: COP8FG devices are 15 MHz versions of the COP8SG devices.

The COP8FGx5 Family ROM based microcontrollers are highly integrated COP8™ Feature core devices with 8k to 32k memory and advanced features including Analog comparators, and zero external components. These single-chip CMOS devices are suited for more complex applications requiring a full featured controller with larger memory, low EMI, two comparators, and a full-duplex USART. COP8FGx7 devices are 100% form-fit-function compatible 8k or 32k OTP (One Time Programmable) versions for use in production or development.

Erasable windowed versions are available for use with a range of COP8 software and hardware development tools.

Family features include an 8-bit memory mapped architecture, 15 MHz CKI with 0.67 µs instruction cycle, 14 interrupts, three multi-function 16-bit timer/counters with PWM, full duplex USART, MICROWIRE/PLUS™, two analog comparators, two power saving HALT/IDLE modes, MIWU, idle timer, on-chip R/C oscillator, high current outputs, user selectable options (WATCHDOG™, 4 clock/oscillator modes, power-on-reset), 4.5V to 5.5V operation, program code security, and 28/40/44 pin packages.

Devices included in this datasheet are:

Device	Memory (bytes)	RAM (bytes)	I/O Pins	Packages	Temperature
COP8FGE5	8k ROM	256	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGG5	16k ROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGH5	20k ROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGK5	24k ROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGR5	32k ROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGE7	8k OTP EPROM	256	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGR7	32k OTP EPROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	-40 to +85˚C
COP8FGR7-Q3	32k EPROM	512	24/36/40	28 DIP/SOIC, 40 DIP, 44 PLCC/QFP	Room Temp.

Key Features

- Low cost 8-bit microcontroller
- Quiet Design (low radiated emissions)
- Multi-Input Wakeup pins with optional interrupts (8 pins)
- Mask selectable clock options
  - — Crystal oscillator
  - — Crystal oscillator option with on-chip bias resistor
  - — External oscillator
  - — Internal R/C oscillator
- Internal Power-On-Reset — user selectable
- WATCHDOG and Clock Monitor Logic — user selectable
- Eight high current outputs
- 256 or 512 bytes on-board RAM
- 8k to 32k ROM or OTP EPROM with security feature

CPU Features

- Versatile easy to use instruction set
- 0.67 µs instruction cycle time
- Fourteen multi-source vectored interrupts servicing
  - — External interrupt / Timers T0 — T3
  - — MICROWIRE/PLUS Serial Interface
  - — Multi-Input Wake Up
  - — Software Trap
  - — USART (2; 1 receive and 1 transmit)
  - — Default VIS (default interrupt)
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers
- True bit manipulation
- BCD arithmetic instructions

Peripheral Features

- Multi-Input Wakeup Logic
- Three 16-bit timers (T1 — T3), each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event Counter mode
  - — Input Capture mode
- Idle Timer (T0)
- MICROWIRE/PLUS Serial Interface (SPI Compatible)
- Full Duplex USART
- Two Analog Comparators

I/O Features

- Software selectable I/O options (TRI-STATE® Output,Push-Pull Output, Weak Pull-Up Input, and High Impedance Input)
- Schmitt trigger inputs on ports G and L
- Eight high current outputs
- Packages: 28 SO with 24 I/O pins, 40 DIP with 36 I/O pins, 44 PLCC and PQFP with 40 I/O pins

Fully Static CMOS Design

- Low current drain (typically < 4 µA)
- Two power saving modes: HALT and IDLE

Temperature Range

- −40˚C to +85˚C

Development Support

- Windowed packages for DIP and PLCC
- Real time emulation and full program debug offered by MetaLink Development System

Block Diagram

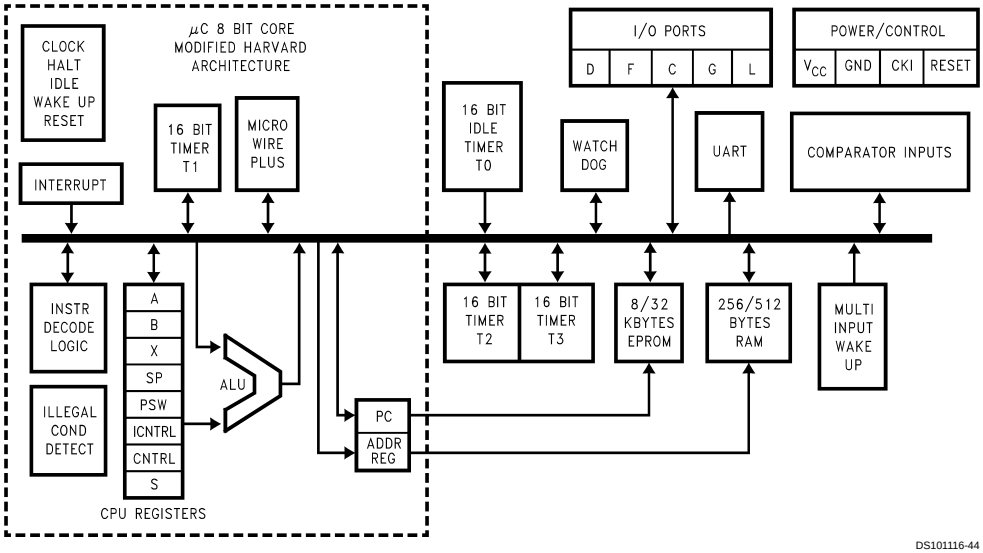


FIGURE 1. COP8FGx Block Diagram

1.0 Device Description

1.1 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 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.

1.2 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 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. 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.

1.2.1 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.

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.

1.2.2 Many Single-Byte, Multifunction Instructions

The COP8 instruction set utilizes many single-byte, multifunction 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 (analogous to "FOR NEXT" in higher level languages).

1.2.3 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.

1.2.4 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). And 15 memory-maped registers allow designers to optimize the precise implementation of certain specific instructions.

1.3 EMI REDUCTION

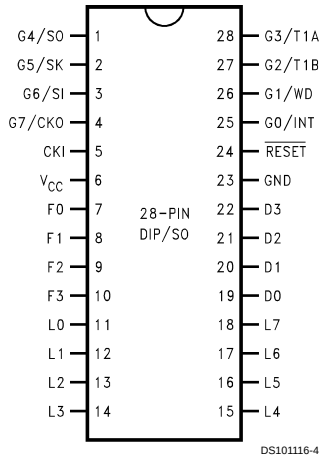
The COP8FGx5 family of devices incorporates circuitry that guards against electromagnetic interference — an increasing problem in today's microcontroller board designs. National's patented EMI reduction technology offers low EMI clock circuitry, gradual turn-on output drivers (GTOs) and internal I_{CC} 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.

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

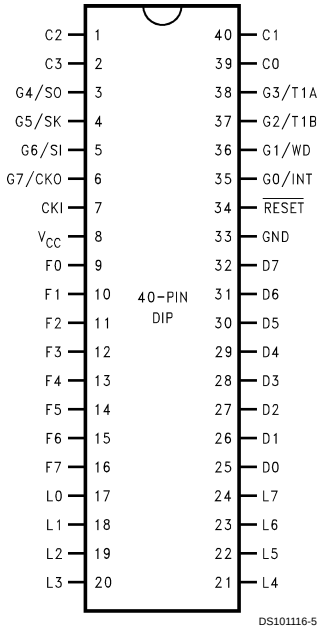
The COP8 family offers a wide range of packages and do not waste pins: up to 90.9% (or 40 pins in the 44-pin package) are devoted to useful I/O.

Connection Diagrams

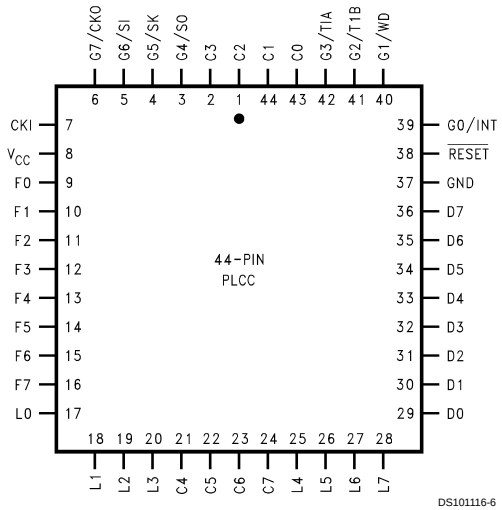


Note 1: X = E for 8k, G for 16k,
H for 20k, K for 24k, R for 32k
Y = 5 for ROM, 7 for OTP

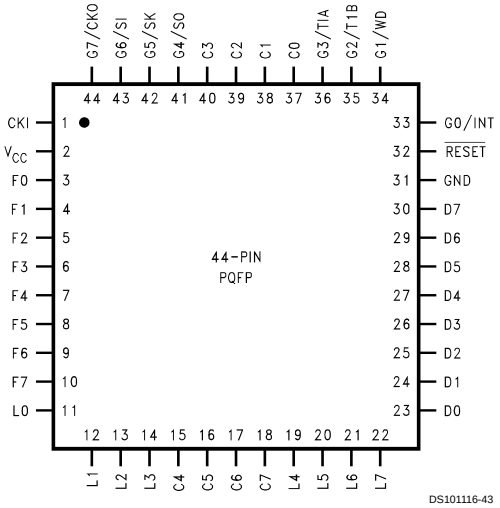
Top View
Order Number COP8FGXY28M8
See NS Package Number M28B
Order Number COP8FGXY28N8
See NS Package Number N28A
Order Number COP8FGR728Q3
See NS Package Number D28JQ



Top View
Order Number COP8FGXY40N8
See NS Package Number N40A
Order Number COP8FGR540Q3
See NS Package Number D40KQ



Top View
Order Number COP8FGXY44V8
See NS Package Number V44A
Order Number COP8FGR744J3
See NS Package Number EL44C



Top View
Order Number COP8FGXYVEJ8
See NS Package Number VEJ44A

FIGURE 2. Connection Diagrams

Connection Diagrams (Continued)

Pinouts for 28 -, 40- and 44-Pin Packages

Port	Type	Alt. Fun	28-Pin SO	40-Pin DIP	44-Pin PLCC	44-Pin PQFP
L0	I/O	MIWU	11	17	17	11
L1	I/O	MIWU or CKX	12	18	18	12
L2	I/O	MIWU or TDX	13	19	19	13
L3	I/O	MIWU or RDX	14	20	20	14
L4	I/O	MIWU or T2A	15	21	25	19
L5	I/O	MIWU or T2B	16	22	26	20
L6	I/O	MIWU or T3A	17	23	27	21
L7	I/O	MIWU or T3B	18	24	28	22
G0	I/O	INT	25	35	39	33
G1	I/O	WDOUT*	26	36	40	34
G2	I/O	T1B	27	37	41	35
G3	I/O	T1A	28	38	42	36
G4	I/O	SO	1	3	3	41
G5	I/O	SK	2	4	4	42
G6	I	SI	3	5	5	43
G7	I	CKO	4	6	6	44
D0	O		19	25	29	23
D1	O		20	26	30	24
D2	O		21	27	31	25
D3	O		22	28	32	26
D4	O		29	33	27	
D5	O		30	34	28	
D6	O		31	35	29	
D7	O		32	36	30	
F0	I/O		7	9	9	3
F1	I/O	COMP1IN−	8	10	10	4
F2	I/O	COMP1IN+	9	11	11	5
F3	I/O	COMP1OUT	10	12	12	6
F4	I/O	COMP2IN−		13	13	7
F5	I/O	COMP2IN+		14	14	8
F6	I/O	COMP2OUT		15	15	9
F7	I/O			16	16	10
C0	I/O			39	43	37
C1	I/O			40	44	38
C2	I/O			1	1	39
C3	I/O			2	2	40
C4	I/O				21	15
C5	I/O				22	16
C6	I/O				23	17
C7	I/O				24	18
V_{CC}			6	8	8	2
GND			23	33	37	31
CKI	I		5	7	7	1
$\overline{\text{RESET}}$	I		24	34	38	32

* G1 operation as WDOUT is controlled by ECON bit 2.

2.1 Ordering Information

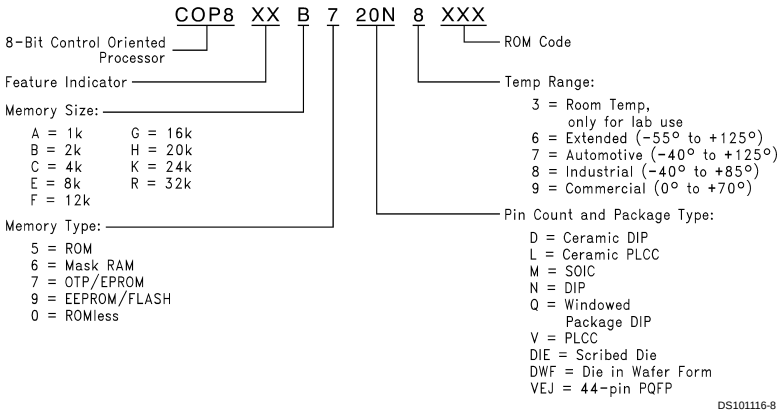


FIGURE 3. Part Numbering Scheme

3.0 Electrical Characteristics

Absolute Maximum Ratings (Note 2)

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

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	−0.3V to V_{CC} +0.3V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65˚C to +140˚C
ESD Protection Level	2kV (Human Body Model)

Note 2: *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.*

DC Electrical Characteristics

−40˚C ≤ T_A ≤ +85˚C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Rise Time		10		50 x 10^6	ns
V_{CC} Start Voltage to Guarantee POR		0		0.25	V
Power Supply Ripple (Note 4)	Peak-to-Peak			0.1 V_{cc}	V
Supply Current (Note 5)					
CKI = 15 MHz	V_{CC} = 5.5V, t_C = 0.67 µs			9.0	mA
CKI = 10 MHz	V_{CC} = 5.5V, t_C = 1 µs			6.0	mA
CKI = 4 MHz	V_{CC} = 4.5V, t_C = 2.5 µs			2.1	mA
HALT Current (Note 6)	V_{CC} = 5.5V, CKI = 0 MHz		<4	10	µA
IDLE Current (Note 5)					
CKI = 15 MHz	V_{CC} = 5.5V, t_C = 0.67 µs			2.25	mA
CKI = 10 MHz	V_{CC} = 5.5V, t_C = 1 µs			1.5	mA
CKI = 4 MHz	V_{CC} = 4.5V, t_C = 2.5 µs			0.8	mA
Input Levels (V_{IH}, V_{IL})					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{cc}			V
Logic Low				0.2 V_{cc}	V
CKI, All Other Inputs					
Logic High		0.7 V_{cc}			V
Logic Low				0.2 V_{cc}	V
Internal Bias Resistor for the Crystal/Resonator Oscillator		0.5	1	2	MΩ
CKI Resistance to V_{CC} or GND when R/C Oscillator is selected	V_{CC} = 5.5V	5	8	11	kΩ
Hi-Z Input Leakage	V_{CC} = 5.5V	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V, V_{IN} = 0V	−40		−250	µA
G and L Port Input Hysteresis	V_{CC} = 5.5V	0.25 V_{cc}			V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	−0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	−10.0		−110	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	−0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage	V_{CC} = 5.5V	−2		+2	µA
Allowable Sink Current per Pin (Note 9)					
D Outputs and L0 to L3	15			mA	
All Others	3			mA	

DC Electrical Characteristics (Continued)

$-40^{\circ}C \le T_A \le +85^{\circ}C$ unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Maximum Input Current without Latchup (Note 7)	Room Temp.			±200	mA
RAM Retention Voltage, Vr		2.0			V
V_{CC} Rise Time from a $V_{CC} \ge 2.0V$	(Note 10)	12			µs
Input Capacitance	(Note 9)			7	pF
Load Capacitance on D2	(Note 9)			1000	pF

AC Electrical Characteristics

$-40^{\circ}C \le T_A \le +85^{\circ}C$ unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_C)					
Crystal/Resonator, External	$4.5V \le V_{CC} \le 5.5V$	0.67			µs
R/C Oscillator (Internal)	$4.5V \le V_{CC} \le 5.5V$		2		µs
Frequency Variation (Note 9)	$4.5V \le V_{CC} \le 5.5V$		±35		%
External CKI Clock Duty Cycle (Note 9)	fr = Max	45		55	%
Rise Time (Note 9)	fr = 10 MHz Ext Clock			12	ns
Fall Time (Note 9)	fr = 10 MHz Ext Clock			8	ns
Output Propagation Delay (Note 8)	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4.5V \le V_{CC} \le 5.5V$			0.7	µs
All Others	$4.5V \le V_{CC} \le 5.5V$			1.0	µs
MICROWIRE Setup Time (t_{UWS}) (Note 11)		20			ns
MICROWIRE Hold Time (t_{UWH}) (Note 11)		56			ns
MICROWIRE Output Propagation Delay (t_{UPD}) (Note 11)				220	ns
Input Pulse Width (Note 9)					
Interrupt Input High Time		1			t_C
Interrupt Input Low Time		1			t_C
Timer 1, 2, 3, Input High Time		1			t_C
Timer 1 2, 3, Input Low Time		1			t_C
Reset Pulse Width		1			µs

Note 3: t_C = Instruction cycle time.

Note 4: Maximum rate of voltage change must be < 0.5 V/ms.

Note 5: Supply and IDLE currents are measured with CKI driven with a square wave Oscillator, External Oscillator, inputs connected to V_{CC} and outputs driven low but not connected to a load.

Note 6: The HALT mode will stop CKI from oscillating in the R/C and the Crystal configurations. In the R/C configuration, CKI is forced high internally. In the crystal or external configuration, 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}; clock monitor disabled. Parameter refers to HALT mode entered via setting bit 7 of the G Port data register.

Note 7: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network. These pins allow input voltages > V_{CC} and the pins will have sink current to V_{CC} when biased at voltages > V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to < 14V. WARNING: Voltages in excess of 14V will cause damage to the pins. This warning excludes ESD transients.

Note 8: The output propagation delay is referenced to the end of the instruction cycle where the output change occurs.

Note 9: Parameter characterized but not tested.

Note 10: Rise times faster than the minimum specification may trigger an internal power-on-reset.

Note 11: MICROWIRE Setup and Hold Times and Propagation Delays are referenced to the appropriate edge of the MICROWIRE clock. See and the MICROWIRE operation description.

Comparators AC and DC Characteristics

V_{CC} = 5V, −40˚C ≤ T_A ≤ +85˚C.

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage (Note 12)	0.4V ≤ V_{IN} ≤ V_{CC} − 1.5V		±5	±15	mV
Input Common Mode Voltage Range		0.4		V_{CC} − 1.5	V
Voltage Gain			100		dB
Low Level Output Current	V_{OL} = 0.4V	−1.6			mA
High Level Output Current	V_{OH} = V_{CC} − 0.4V	1.6			mA
DC Supply Current per Comparator (When Enabled)				150	µA
Response Time (Note 13)	200 mV step input 100 mV Overdrive, 100 pF Load			200	ns

Note 12: The comparator inputs are high impedance port inputs and, as such, input current is limited to port input leakage current.

Note 13: Response time is measured from a step input to a valid logic level at the comparator output. software response time is dependent of instruction execution.

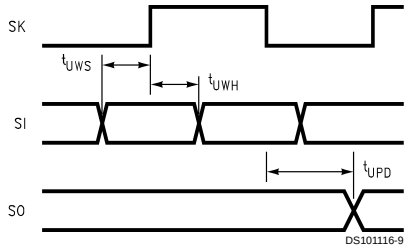
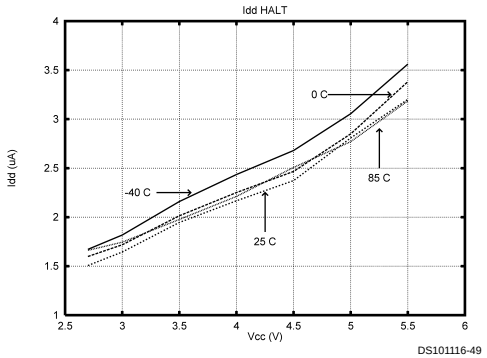
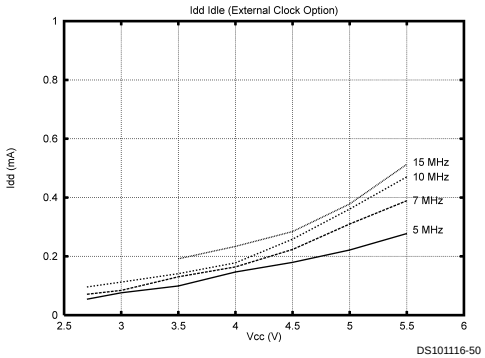


FIGURE 4. MICROWIRE/PLUS Timing

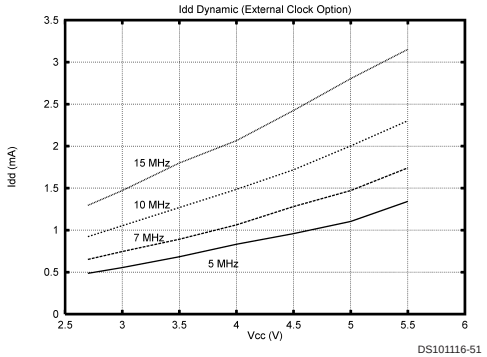
Typical Performance Characteristics T_A = 25˚C (unless otherwise specified)



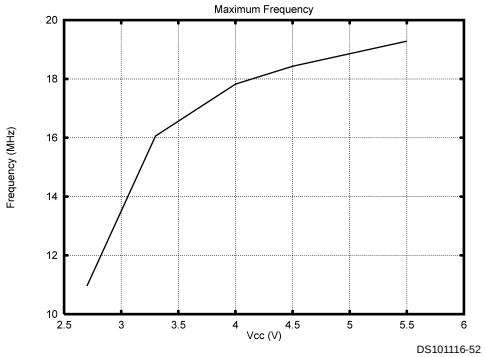
DS101116-49



DS101116-50



DS101116-51



DS101116-52

4.0 Pin Descriptions

The COP8FGx 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 less external components and lower costs. Below is the general description of all available pins.

V_{CC} and GND are the power supply pins. All V_{CC} and GND pins must be connected.

CKI is the clock input. This can come from the Internal R/C oscillator, external, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset description section.

Each device contains four bidirectional 8-bit I/O ports (C, G, L and F), where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports L and G), 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 two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the memory map for the various addresses associated with the I/O ports.) *Figure 5* 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 L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

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

L7 Multi-input Wakeup or T3B (Timer T3B Input)

L6 Multi-input Wakeup or T3A (Timer T3A Input)

L5 Multi-input Wakeup or T2B (Timer T2B Input)

L4 Multi-input Wakeup or T2A (Timer T2A Input)

L3 Multi-input Wakeup and/or RDX (USART Receive)

L2 Multi-input Wakeup or TDX (USART Transmit)

L1 Multi-input Wakeup and/or CKX (USART Clock)

L0 Multi-input Wakeup

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 pins have Schmitt Triggers on their inputs.**Pin G1 serves as the dedicated WATCHDOG output with weak pullup if WATCHDOG feature is selected by the Mask 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 zeroes.

Each 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 Serial Data Input)

G5 SK (MICROWIRE Serial Clock)

G4 SO (MICROWIRE 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)

Port C is an 8-bit I/O port. The 40-pin device does not have a full complement of Port C pins. The unavailable pins are not terminated. A read operation on these unterminated pins will return unpredictable values. The 28 pin device do not offer Port C. On this device, the associated Port C Data and Configuration registers should not be used.

Port F is an 8-bit I/O port. The 28--pin device does not have a full complement of Port F pins. The unavailable pins are not terminated. A read operation on these unterminated pins will return unpredictable values.

Port F1–F3 are used for Comparator 1. Port F4–F6 are used for Comparator 2.

The Port F has the following alternate features:

F6 COMP2OUT (Comparator 2 Output)

F5 COMP2+IN (Comparator 2 Positive Input)

F4 COMP2-IN (Comparator 2 Negative Input)

F3 COMP1OUT (Comparator 1 Output)

F2 COMP1+IN (Comparator 1 Positive Input)

F1 COMP1-IN (Comparator 1 Negative Input)

Note: For compatibility with existing software written for COP888xG devices and with existing Mask ROM devices, a read of the Port I input pins

4.0 Pin Descriptions (Continued)

(address xxD7) will return the same data as reading the Port F input pins (address xx96). It is recommended new applications which will go to production with the COP8FGx use the Port F addresses. Note that compatible ROM devices contains the input only Port I instead of the bi-directional Port F.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs (except D2) together in order to get a higher drive.

Note: Care must be exercised with the D2 pin operation. At RESET, the external loads on this pin must ensure that the output voltages stay above 0.7 V_{CC} to prevent the chip from entering special modes. Also keep the external loading on D2 to less than 1000 pF.

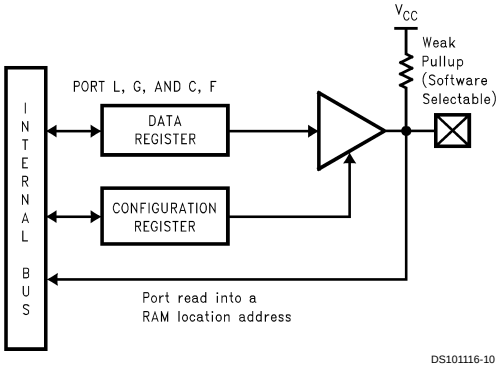


FIGURE 5. I/O Port Configurations

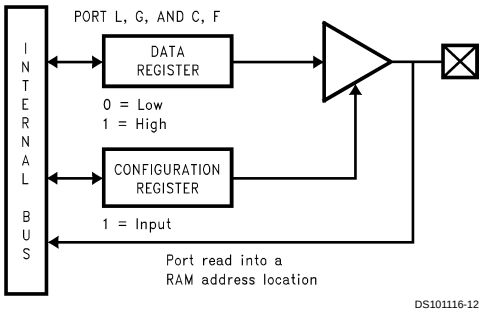


FIGURE 6. I/O Port Configurations — Output Mode

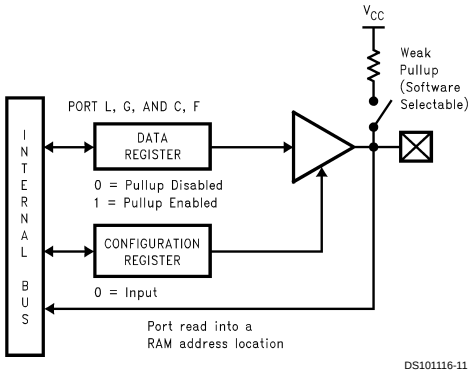


FIGURE 7. I/O Port Configurations — Input Mode

5.0 Functional Description

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

5.1 CPU REGISTERS

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

There are six 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.

S is the 8-bit Segment Address Register used to extend the lower half of the address range (00 to 7F) into 256 data segments of 128 bytes each.

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 02F Hex (devices with 64 bytes of RAM), or initialized to RAM address 06F Hex (devices with 128 bytes of RAM).

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

5.2 PROGRAM MEMORY

The program memory consists of varies sizes of ROM. 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 0FF Hex. The contents of the program memory read 00 Hex in the erased state. Program execution starts at location 0 after RESET.

5.3 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, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The data memory consists of 256 or 512 bytes of RAM. Sixteen bytes of RAM are mapped as “registers” at addresses 0F0 to 0FE 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 (except 0FF) 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

5.0 Functional Description (Continued)

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.

5.4 DATA MEMORY SEGMENT RAM EXTENSION

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S).

The data store memory is either addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B, X, or SP pointers (each contains a single-byte address). This single-byte address allows an addressing range of 256 locations from 00 to FF hex. The upper bit of this single-byte address divides the data store memory into two separate sections as outlined previously. With the exception of the RAM register memory from address locations 00F0 to 00FF, all RAM memory is memory mapped with the upper bit of the single-byte address being equal to zero. This allows the upper bit of the single-byte address to determine whether or not the base address range (from 0000 to 00FF) is extended. If this upper bit equals one (representing address range 0080 to 00FF), then address extension does not take place. Alternatively, if this upper bit equals zero, then the data segment extension register S is used to extend the base address range (from 0000 to 007F) from XX00 to XX7F, where XX represents the 8 bits from the S register. Thus the 128-byte data segment extensions are located from addresses 0100 to 017F for data segment 1, 0200 to 027F for data segment 2, etc., up to FF00 to FF7F for data segment 255. The base address range from 0000 to 007F represents data segment 0.

Figure 8 illustrates how the S register data memory extension is used in extending the lower half of the base address range (00 to 7F hex) into 256 data segments of 128 bytes each, with a total addressing range of 32 kbytes from XX00 to XX7F. This organization allows a total of 256 data segments of 128 bytes each with an additional upper base segment of 128 bytes. Furthermore, all addressing modes are available for all data segments. The S register must be changed under program control to move from one data segment (128 bytes) to another. However, the upper base segment (containing the 16 memory registers, I/O registers, control registers, etc.) is always available regardless of the contents of the S register, since the upper base segment (address range 0080 to 00FF) is independent of data segment extension.

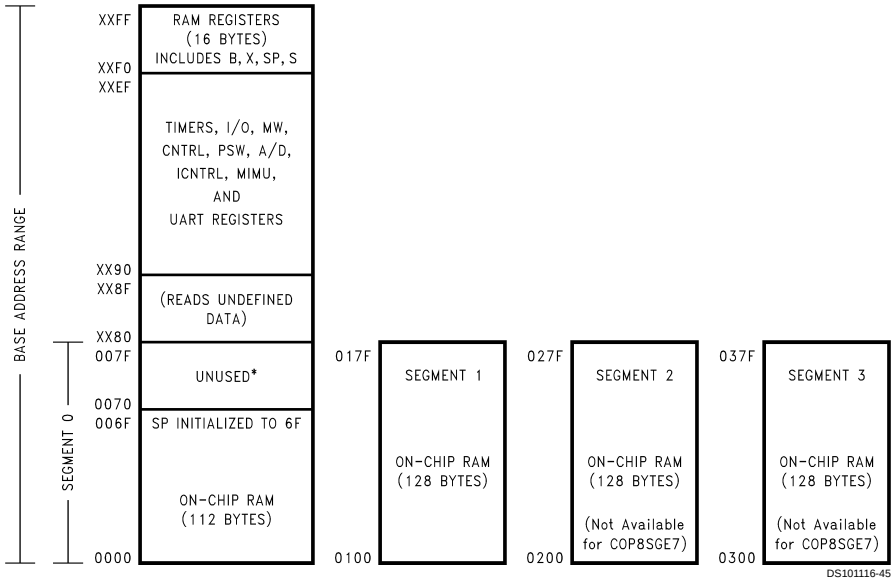


FIGURE 8. RAM Organization

The instructions that utilize the stack pointer (SP) always reference the stack as part of the base segment (Segment 0), regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupts) is always located in the base segment. The stack pointer will be initialized to point at data memory location 006F as a result of reset.

The 128 bytes of RAM contained in the base segment are split between the lower and upper base segments. The first 112 bytes of RAM are resident from address 0000 to 006F in the lower base segment, while the remaining 16 bytes of RAM represent the 16 data memory registers located at addresses 00F0 to 00FF of the upper base segment. No RAM is located at the upper sixteen addresses (0070 to 007F) of the lower base segment.

Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 384 bytes of RAM in this device are memory mapped at address locations 0100 to 017F, 0200 to 027F and 0300 to 037F hex.

5.0 Functional Description (Continued)

Memory address ranges 0200 to 027F and 0300 to 037F are unavailable on the COP8FGx5 and, if read, will return undefined data.

5.5 ECON (CONFIGURATION) REGISTER

For compatibility with COP8FGx7 devices, mask options are defined by an ECON Configuration Register which is programmed at the same time as the program code. Therefore, the register is programmed at the same time as the program memory.

The format of the ECON register is as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	POR	SECURITY	CKI 2	CKI 1	WATCH DOG	F-Port	HALT

Bit 7 = x This is for factory test. The polarity is "Don't Care."

Bit 6 = 1 Power-on reset enabled.

= 0 Power-on reset disabled.

Bit 5 = 1 Security enabled.

Bits 4, 3 = 0, 0 External CKI option selected. G7 is available as a HALT restart and/or general purpose input. CKI is clock input.

= 0, 1 R/C oscillator option selected. G7 is available as a HALT restart and/or general purpose input. CKI clock input. Internal R/C components are supplied for maximum R/C frequency.

= 1, 0 Crystal oscillator with on-chip crystal bias resistor disabled. G7 (CKO) is the clock generator output to crystal/resonator.

= 1, 1 Crystal oscillator with on-chip crystal bias resistor enabled. G7 (CKO) is the clock generator output to crystal/resonator.

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 Force port I compatibility. Disable port F outputs and pull-ups. This is intended for compatibility with existing code and Mask ROMMed devices only. This bit should be programmed to 0 for all other applications.

= 0 Enable full port F capability.

Bit 0 = 1 HALT mode disabled.

= 0 HALT mode enabled.

5.6 USER STORAGE SPACE IN EPROM

The ECON register is outside of the normal address range of the ROM and can not be accessed by the executing software.

The COP8 assembler defines a special ROM section type, CONF, into which the ECON may be coded. Both ECON and User Data are programmed automatically by programmers that are certified by National.

The following examples illustrate the declaration of ECON and the User information.

Syntax:

```
[label:] .sect   econ, conf
         .db     value  ;1 byte,
                        ;configures options
         .db     <user information>
         .endsect       ; up to 8 bytes
```

Example: The following sets a value in the ECON register and User Identification for a COP8FGR728M7. The ECON bit values shown select options: Power-on enabled, Security disabled, Crystal oscillator with on-chip bias disabled, WATCHDOG enabled and HALT mode enabled.

```
.sect  econ, conf
.db    0x55       ;por, xtal, wd, halt
.db    'my v1.00' ;user data declaration
.endsect
```

5.7 RESET

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

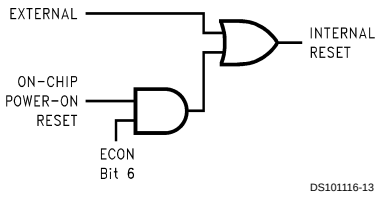


FIGURE 9. Reset Logic

The following occurs upon initialization:

Port L: TRI-STATE (High Impedance Input)

Port C: TRI-STATE (High Impedance Input)

Port G: TRI-STATE (High Impedance Input)

Port F: TRI-STATE (High Impedance Input)

Port D: HIGH

PC: CLEARED to 0000

PSW, CNTRL and ICNTRL registers: CLEARED

SIOR:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

T2CNTRL: CLEARED

T3CNTRL: CLEARED

Accumulator, Timer 1, Timer 2 and Timer 3:

RANDOM after RESET with crystal clock option (power already applied)

UNAFFECTED after RESET with R/C clock option (power already applied)

RANDOM after RESET at power-on

WKEN, WKEDG: CLEARED

WKPND: RANDOM

SP (Stack Pointer):

Initialized to RAM address 06F Hex

B and X Pointers:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

S Register: CLEARED

5.0 Functional Description (Continued)

RAM:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

USART:

PSR, ENU, ENUR, ENUI: Cleared except the TBMT bit which is set to one.

COMPARATORS:

CMPSL; CLEARED

WATCHDOG (if enabled):

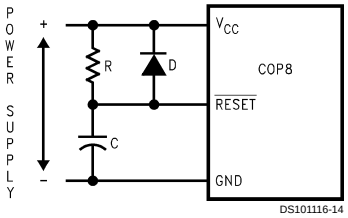
The device comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of 64k t_C 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 error output on pin G1. This error output will continue until 16 t_C–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will go high.

5.7.1 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. During Power-Up initialization, the user must ensure that the $\overline{\text{RESET}}$ pin is held low until the device is within the specified V_{CC} voltage. An R/C circuit on the $\overline{\text{RESET}}$ pin with a delay 5 times (5x) greater than the power supply rise time or 15 µs whichever is greater, is recommended. Reset should also be wide enough to ensure crystal start-up upon Power-Up.

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

A recommended reset circuit for this device is shown in *Figure 10*.



RC >5x power supply rise time or 15 µs, whichever is greater.

FIGURE 10. Reset Circuit Using External Reset

5.7.2 On-Chip Power-On Reset

The on-chip reset circuit is selected by a bit in the ECON register. When enabled, 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 10 ns and 50 ms).To guarantee an on-chip power-on-reset, V_{CC}must start at a voltage less than the start voltage specified in the DC characteristics. Also, if V_{CC} be lowered to the start voltage before powering back up to the operating range. If this is not possible, it is recommended that external reset be used.

Under no circumstances should the $\overline{\text{RESET}}$ pin be allowed to float. If the on-chip Power-On Reset feature is being used, $\overline{\text{RESET}}$ pin should be connected directly to V_{CC}. The output of the power-on reset detector will **always** preset the Idle timer to 0FFF(4096 t_C). At this time, the internal reset will be generated.

If the Power-On Reset feature is enabled, 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 the operating frequency within the 4096 t_C. After the underflow, the logic is designed such that no additional internal resets occur as long as V_{CC} remains above 2.0V.

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

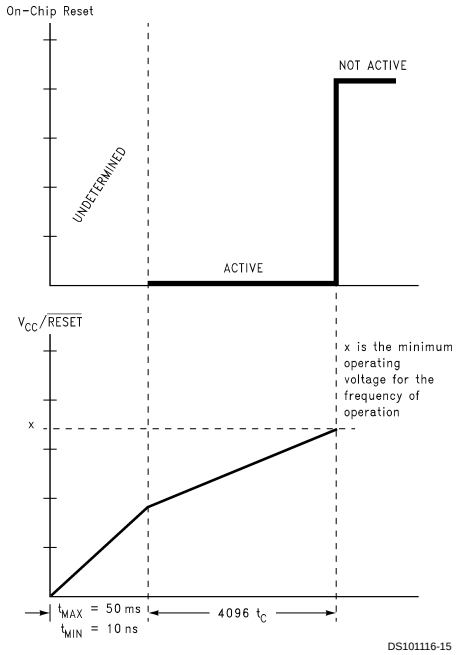


FIGURE 11. Reset Timing (Power-On Reset Enabled) with V_{CC} Tied to $\overline{\text{RESET}}$

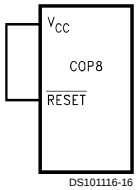


FIGURE 12. Reset Circuit Using Power-On Reset

5.0 Functional Description (Continued)

5.8 OSCILLATOR CIRCUITS

There are four clock oscillator options available: Crystal Oscillator with or without on-chip bias resistor, R/C Oscillator with on-chip resistor and capacitor, and External Oscillator. The oscillator feature is selected by programming the ECON register, which is summarized in *Table 1*.

TABLE 1. Oscillator Option

ECON4	ECON3	Oscillator Option
0	0	External Oscillator
1	0	Crystal Oscillator without Bias Resistor
0	1	R/C Oscillator
1	1	Crystal Oscillator with Bias Resistor

5.8.1 Crystal Oscillator

The crystal Oscillator mode can be selected by programming ECON 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 ECON Bit 3 to 1 with the crystal oscillator option selection. The value of the resistor is in the range of 0.5M to 2M (typically 1.0M). *Table 2* shows the component values required for various standard crystal values. Resistor R2 is only used when the on-chip bias resistor is disabled. *Figure 13* shows the crystal oscillator connection diagram.

TABLE 2. Crystal Oscillator Configuration, T_A = 25˚C, V_{CC} = 5V

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq. (MHz)
0	1	18	18	15
0	1	20	20	10
0	1	25	25	4
5.6	1	100	100–156	0.455

5.8.2 External Oscillator

The External Oscillator mode can be selected by programming ECON Bit 3 to 0 and ECON 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. G7/CKO is available as a general purpose input G7 and/or Halt control. *Figure 14* shows the external oscillator connection diagram.

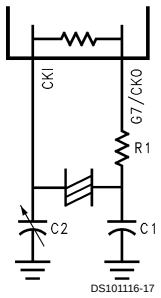
5.8.3 R/C Oscillator

The R/C Oscillator mode can be selected by programming ECON Bit 3 to 1 and ECON Bit 4 to 0. In R/C oscillation mode, CKI is left floating, while G7/CKO is available as a general purpose input G7 and/or HALT control. The R/C controlled oscillator has on-chip resistor and capacitor for maximum R/C oscillator frequency operation. The maximum frequency is 5 MHz ± 35% for V_{CC} between 4.5V to 5.5V and temperature range of −40˚C to +85˚C. For max frequency operation, the CKI pin should be left floating. For lower frequencies, an external capacitor should be connected between CKI and either V_{CC} or GND. Immunity of the R/C oscillator to external noise can be improved by connecting one half the external capacitance to V_{CC} and one half to 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 external capacitance on the CKI pin. *Figure 15* shows the R/C oscillator configuration.

TABLE 3. R/C Oscillator Configuration, −40˚C to +85˚C, V_{CC} = 4.5V to 5.5V, OSC Freq. Variation of ± 35%

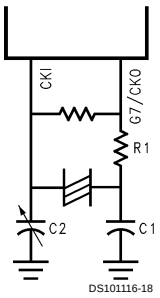
External Capacitor (pF)*	R/C OSC Freq (MHz)	Instr. Cycle (µs)
0	5	2.0
9	4	2.5
52	2	5.0
125	1	10
6100	32 kHz	312.5
* Assumes 3-5 pF board capacitance.		

With On-Chip Bias Resistor



DS101116-17

Without On-Chip Bias Resistor



DS101116-18

FIGURE 13. Crystal Oscillator

5.0 Functional Description (Continued)

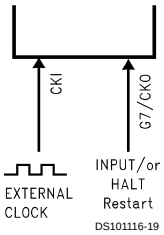
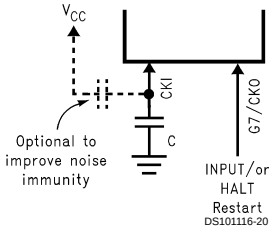
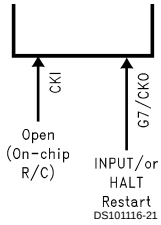


FIGURE 14. External Oscillator



For operation at lower than maximum R/C oscillator frequency.



For operation at maximum R/C oscillator frequency.

FIGURE 15. R/C Oscillator

5.0 Functional Description (Continued)

5.9 CONTROL REGISTERS

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)

PSW Register (Address X'00EF)

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE
Bit 7							Bit 0

The PSW register contains the following select bits:

- HC Half Carry Flag
- C Carry Flag
- T1PNDA 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 R/C (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and R/C instructions, ADC, SUBC, RRC and RLC instructions affect the Carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

Reserved	LPEN	T0PND	T0EN	µWPND	µWEN	T1PNDB	T1ENB
Bit 7							Bit 0

The ICNTRL register contains the following bits:

- Reserved This bit is reserved and must be zero
- LPEN L Port Interrupt Enable (Multi-Input Wakeup/ Interrupt)
- T0PND Timer T0 Interrupt pending
- T0EN Timer T0 Interrupt Enable (Bit 12 toggle)
- µWPND MICROWIRE/PLUS interrupt pending
- µWEN Enable MICROWIRE/PLUS interrupt
- T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge
- T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

T2CNTRL Register (Address X'00C6)

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

The T2CNTRL control register contains the following bits:

- T2C3 Timer T2 mode control bit
- T2C2 Timer T2 mode control bit
- T2C1 Timer T2 mode control bit
- T2C0 Timer T2 Start/Stop control in timer modes 1 and 2, T2 Underflow Interrupt Pending Flag in timer mode 3
- T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
- T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
- T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge
- T2ENB Timer T2 Interrupt Enable for Timer Underflow or T2B Input capture edge

T3CNTRL Register (Address X'00B6)

T3C3	T3C2	T3C1	T3C0	T3PNDA	T3ENA	T3PNDB	T3ENB
Bit 7							Bit 0

The T3CNTRL control register contains the following bits:

- T3C3 Timer T3 mode control bit
- T3C2 Timer T3 mode control bit
- T3C1 Timer T3 mode control bit
- T3C0 Timer T3 Start/Stop control in timer modes 1 and 2, T3 Underflow Interrupt Pending Flag in timer mode 3
- T3PNDA Timer T3 Interrupt Pending Flag (Autoreload RA in mode 1, T3 Underflow in mode 2, T3A capture edge in mode 3)
- T3ENA Timer T3 Interrupt Enable for Timer Underflow or T3A Input capture edge
- T3PNDB Timer T3 Interrupt Pending Flag for T3B capture edge
- T3ENB Timer T3 Interrupt Enable for Timer Underflow or T3B Input capture edge

6.0 Timers

Each device contains a very versatile set of timers (T0, T1, T2 and T3). Timer T1, T2 and T3 and associated autoreload/ capture registers power up containing random data.

6.1 TIMER T0 (IDLE TIMER)

Each 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. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

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

6.0 Timers (Continued)

- Timing the width of the internal power-on-reset

The IDLE Timer T0 can generate an interrupt when the twelfth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 2.731 ms at the maximum clock frequency (t_C = 0.67 µs). A control flag T0EN allows the interrupt from the twelfth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

6.2 TIMER T1, TIMER T2 and TIMER T3

Each device have a set of three powerful timer/counter blocks, T1, T2 and T3. Since T1, T2, and T3 are identical, all comments are equally applicable to any of the three timer blocks which will be referred to as Tx.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB 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 TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

6.2.1 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 PMW 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 Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

Figure 16 shows a block diagram of the timer in PWM mode.

The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB 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.

6.2.2 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, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

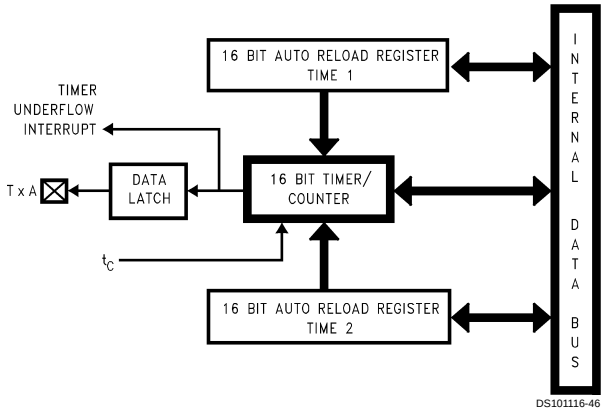


FIGURE 16. Timer in PWM Mode

6.0 Timers (Continued)

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

Figure 17 shows a block diagram of the timer in External Event Counter mode.

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

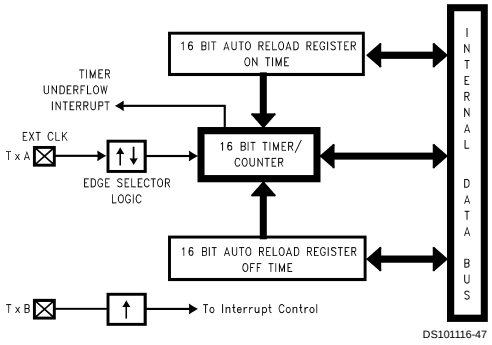


FIGURE 17. Timer in External Event Counter Mode

6.2.3 Mode 3. Input Capture Mode

Each device can precisely measure external frequencies or time external events by placing the timer block, Tx, 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 Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified 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 TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 18 shows a block diagram of the timer T1 in Input Capture mode. Timer T2 and T3 are identical to T1.

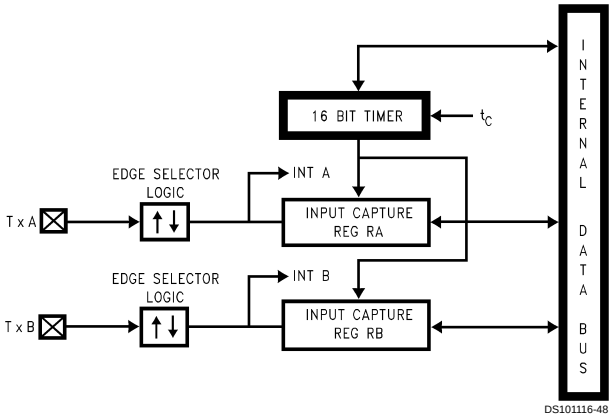


FIGURE 18. Timer in Input Capture Mode

6.0 Timers (Continued)

6.3 TIMER CONTROL FLAGS

The control bits and their functions are summarized below.

TxC3 Timer mode control

TxC2 Timer mode control

TxC1 Timer mode control

TxC0 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)

TxPNDA Timer Interrupt Pending Flag

TxENA Timer Interrupt Enable Flag
1 = Timer Interrupt Enabled
0 = Timer Interrupt Disabled

TxPNDB Timer Interrupt Pending Flag

TxENB Timer Interrupt Enable Flag
1 = Timer Interrupt Enabled
0 = Timer Interrupt Disabled

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

Mode	TxC3	TxC2	TxC1	Description	Interrupt A Source	Interrupt B Source	Timer Counts On
1	1	0	1	PWM: TxA Toggle	Autoreload RA	Autoreload RB	t_C
	1	0	0	PWM: No TxA Toggle	Autoreload RA	Autoreload RB	t_C
2	0	0	0	External Event Counter	Timer Underflow	Pos. TxB Edge	Pos. TxA Edge
	0	0	1	External Event Counter	Timer Underflow	Pos. TxB Edge	Pos. TxA Edge
3	0	1	0	Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
	1	1	0	Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
	0	1	1	Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
	1	1	1	Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

7.0 Power Saving Features

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.

Each device offers 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 wakeup (MIWU).

Each device offers 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.

7.1 HALT MODE

Each 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 devices are 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 devices come 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 devices 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.

Each device supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on Port L. The 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.

On wakeup from G7 or Port L, the devices resume execution from the HALT point. On wakeup from RESET execution will resume from location PC=0 and all RESET conditions apply.

If a crystal or ceramic resonator may be selected as the oscillator, the Wakeup 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 Wakeup 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 9. 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.

Each 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 ECON register. 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.

It is recommended that the user not halt the device by merely stopping the clock in external oscillator mode. If this method is used, there is a possibility of greater than specified HALT current.

If the user wishes to stop an external clock, it is recommended that the CPU be halted by setting the Halt flag first and the clock be stopped only after the CPU has halted.

7.0 Power Saving Features (Continued)

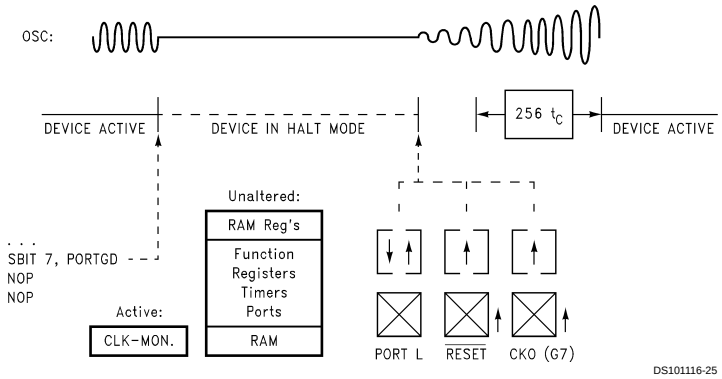


FIGURE 19. Wakeup from HALT

7.2 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 Wakeup 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 µs) of the IDLE Timer toggles.

This toggle condition of the twelfth bit of the IDLE Timer T0 is latched into the T0PND 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 T0PND 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.

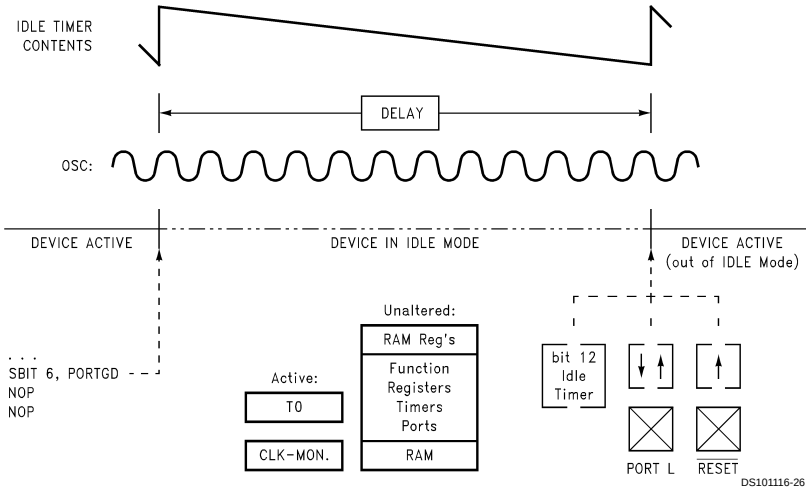


FIGURE 20. Wakeup from IDLE

7.0 Power Saving Features (Continued)

7.3 MULTI-INPUT WAKEUP

The Multi-Input Wakeup feature is used to return (wakeup) the device from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 21 shows the Multi-Input Wakeup logic.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the device to exit the HALT or IDLE modes. The selection is done through the register WKEN. The register WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the register WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN   ; Disable MIWU
SBIT  5, WKEDG  ; Change edge polarity
RBIT  5, WKPND  ; Reset pending flag
SBIT  5, WKEN   ; Enable MIWU
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user must clear the pending flags before attempting to enter the HALT mode.

WKEN and WKEDG are all read/write registers, and are cleared at reset. WKPND register contains random value after reset.

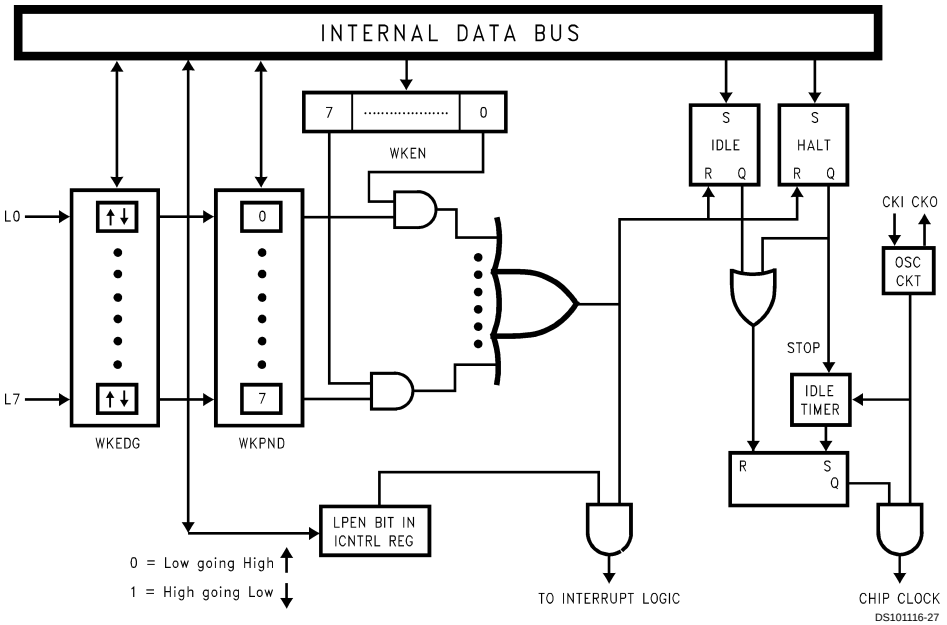


FIGURE 21. Multi-Input Wake Up Logic

8.0 USART

Each device contains a full-duplex software programmable USART. The USART (*Figure 22*) consists of a transmit shift register, a receive shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a USART control and status register (ENU), a USART receive control and status register (ENUR), a USART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the USART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the USART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the USART mode of operation: asynchronous or synchronous.

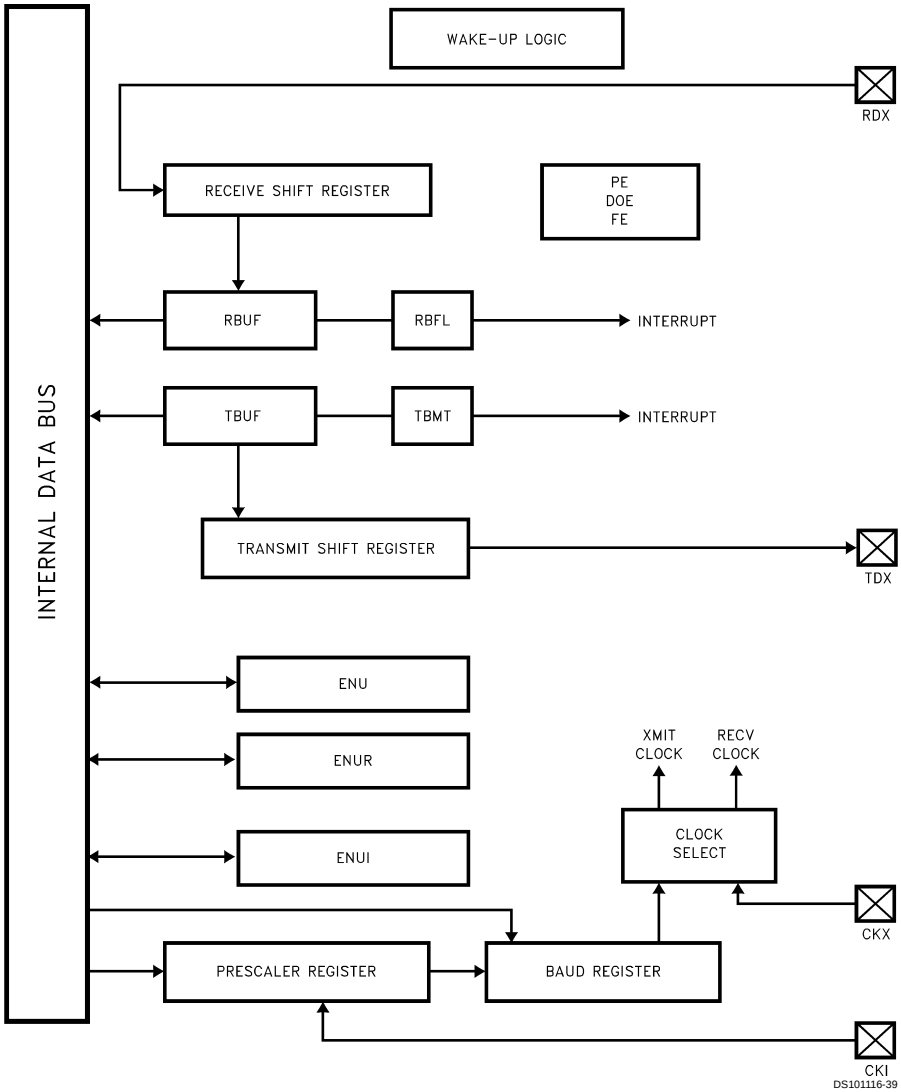


FIGURE 22. USART Block Diagram

8.0 USART (Continued)

8.1 USART CONTROL AND STATUS REGISTERS

The operation of the USART is programmed through three registers: ENU, ENUR and ENUI.

8.2 DESCRIPTION OF USART REGISTER BITS

ENU-USART Control and Status Register (Address at 0BA)

PEN	PSEL1	XBIT9/ PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
Bit 7							Bit 0

PEN: This bit enables/disables Parity (7- and 8-bit modes only). Read/Write, cleared on reset.

PEN = 0 Parity disabled.

PEN = 1 Parity enabled.

PSEL1, PSEL0: Parity select bits. Read/Write, cleared on reset.

PSEL1 = 0, PSEL0 = 0 Odd Parity (if Parity enabled)

PSEL1 = 0, PSEL0 = 1 Even Parity (if Parity enabled)

PSEL1 = 1, PSEL0 = 0 Mark(1) (if Parity enabled)

PSEL1 = 1, PSEL0 = 1 Space(0) (if Parity enabled)

XBIT9/PSEL0: Programs the ninth bit for transmission when the USART is operating with nine data bits per frame. For seven or eight data bits per frame, this bit in conjunction with PSEL1 selects parity. Read/Write, cleared on reset.

CHL1, CHL0: These bits select the character frame format. Parity is not included and is generated/verified by hardware. Read/Write, cleared on reset.

CHL1 = 0, CHL0 = 0 The frame contains eight data bits.

CHL1 = 0, CHL0 = 1 The frame contains seven data bits.

CHL1 = 1, CHL0 = 0 The frame contains nine data bits.

CHL1 = 1, CHL0 = 1 Loopback Mode selected. Transmitter output internally looped back to receiver input. Nine bit framing format is used.

ERR: This bit is a global USART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur. Read only; it cannot be written by software, cleared on reset.

RBFL: This bit is set when the USART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF. Read only; it cannot be written by software, cleared on reset.

TBMT: This bit is set when the USART transfers a byte of data from the TBUF register into the TSFT register for transmission. It is automatically reset when software writes into the TBUF register. Read only, bit is set to “one” on reset; it cannot be written by software.

ENUR-USART Receive Control and Status Register

(Address at 0BB)

DOE	FE	PE	Reserved (Note 14)	RBIT9	ATTN	XMTG	RCVG
Bit 7							Bit 0

Note 14: Bit is reserved for future use. User must set to zero.

DOE: Flags a Data Overrun Error. Read only, cleared on read, cleared on reset.

DOE = 0 Indicates no Data Overrun Error has been detected since the last time the ENUR register was read.

DOE = 1 Indicates the occurrence of a Data Overrun Error.

FE: Flags a Framing Error. Read only, cleared on read, cleared on reset.

FE = 0 Indicates no Framing Error has been detected since the last time the ENUR register was read.

FE = 1 Indicates the occurrence of a Framing Error.

PE: Flags a Parity Error. Read only, cleared on read, cleared on reset.

PE = 0 Indicates no Parity Error has been detected since the last time the ENUR register was read.

PE = 1 Indicates the occurrence of a Parity Error.

SPARE: Reserved for future use. Read/Write, cleared on reset.

RBIT9: Contains the ninth data bit received when the USART is operating with nine data bits per frame. Read only, cleared on reset.

ATTN: ATTENTION Mode is enabled while this bit is set. This bit is cleared automatically on receiving a character with data bit nine set. Read/Write, cleared on reset.

XMTG: This bit is set to indicate that the USART is transmitting. It gets reset at the end of the last frame (end of last Stop bit). Read only, cleared on reset.

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high. Read only, cleared on reset.

ENUI-USART Interrupt and Clock Source Register

(Address at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
Bit 7							Bit 0

STP2: This bit programs the number of Stop bits to be transmitted. Read/Write, cleared on reset.

STP2 = 0 One Stop bit transmitted.

STP2 = 1 Two Stop bits transmitted.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length. Read/Write, cleared on reset.

ETDX: TDX (USART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit. To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers. Read/Write, cleared on reset.

SSEL: USART mode select. Read/Write, cleared on reset.

SSEL = 0 Asynchronous Mode.

SSEL = 1 Synchronous Mode.

XRCLK: This bit selects the clock source for the receiver section. Read/Write, cleared on reset.

XRCLK = 0 The clock source is selected through the PSR and BAUD registers.

XRCLK = 1 Signal on CKX (L1) pin is used as the clock.

XTCLK: This bit selects the clock source for the transmitter section. Read/Write, cleared on reset.

XTCLK = 0 The clock source is selected through the PSR and BAUD registers.

XTCLK = 1 Signal on CKX (L1) pin is used as the clock.

ERI: This bit enables/disables interrupt from the receiver section. Read/Write, cleared on reset.

ERI = 0 Interrupt from the receiver is disabled.

ERI = 1 Interrupt from the receiver is enabled.

8.0 USART (Continued)

ETI: This bit enables/disables interrupt from the transmitter section. Read/Write, cleared on reset.

ETI = 0 Interrupt from the transmitter is disabled.

ETI = 1 Interrupt from the transmitter is enabled.

8.3 Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUI register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the USART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

8.4 USART Operation

The USART has two modes of operation: asynchronous mode and synchronous mode.

8.4.1 ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the USART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the USART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the USART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The USART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

8.4.2 SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the USART is the same as the baud rate.

When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the device generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

8.5 FRAMING FORMATS

The USART supports several serial framing formats (*Figure 23*). The format is selected using control bits in the ENU, ENUR and ENUI registers.

The first format (1, 1a, 1b, 1c) for data transmission (CHL0 = 1, CHL1 = 0) consists of Start bit, seven Data bits (excluding parity) and 7/8, one or two Stop bits. In applications using parity, the parity bit is generated and verified by hardware.

The second format (CHL0 = 0, CHL1 = 0) consists of one Start bit, eight Data bits (excluding parity) and 7/8, one or two Stop bits. Parity bit is generated and verified by hardware.

The third format for transmission (CHL0 = 0, CHL1 = 1) consists of one Start bit, nine Data bits and 7/8, one or two Stop bits. This format also supports the USART “ATTENTION” feature. When operating in this format, all eight bits of TBUF and RBUF are used for data. The ninth data bit is transmitted and received using two bits in the ENU and ENUR registers, called XBIT9 and RBIT9. RBIT9 is a read only bit. Parity is not generated or verified in this mode.

For any of the above framing formats, the last Stop bit can be programmed to be 7/8th of a bit in length. If two Stop bits are selected and the 7/8th bit is set (selected), the second Stop bit will be 7/8th of a bit in length.

The parity is enabled/disabled by PEN bit located in the ENU register. Parity is selected for 7- and 8-bit modes only. If parity is enabled (PEN = 1), the parity selection is then performed by PSEL0 and PSEL1 bits located in the ENU register.

Note that the XBIT9/PSEL0 bit located in the ENU register serves two mutually exclusive functions. This bit programs the ninth bit for transmission when the USART is operating with nine data bits per frame. There is no parity selection in this framing format. For other framing formats XBIT9 is not needed and the bit is PSEL0 used in conjunction with PSEL1 to select parity.

The frame formats for the receiver differ from the transmitter in the number of Stop bits required. The receiver only requires one Stop bit in a frame, regardless of the setting of the Stop bit selection bits in the control register. Note that an implicit assumption is made for full duplex USART operation that the framing formats are the same for the transmitter and receiver.

8.0 USART (Continued)

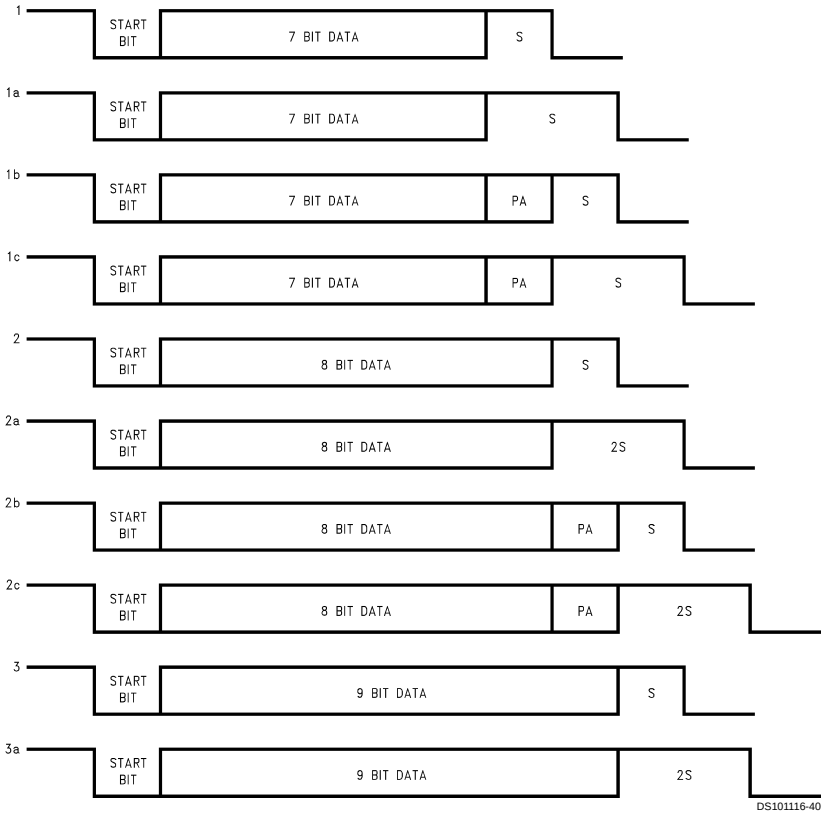


FIGURE 23. Framing Formats

8.6 USART INTERRUPTS

The USART is capable of generating interrupts. Interrupts are generated on Receive Buffer Full and Transmit Buffer Empty. Both interrupts have individual interrupt vectors. Two bytes of program memory space are reserved for each interrupt vector. The two vectors are located at addresses 0xEC to 0xEF Hex in the program memory space. The interrupts can be individually enabled or disabled using Enable Transmit Interrupt (ETI) and Enable Receive Interrupt (ERI) bits in the ENUI register.

The interrupt from the Transmitter is set pending, and remains pending, as long as both the TBMT and ETI bits are set. To remove this interrupt, software must either clear the ETI bit or write to the TBUF register (thus clearing the TBMT bit).

The interrupt from the receiver is set pending, and remains pending, as long as both the RBFL and ERI bits are set. To remove this interrupt, software must either clear the ERI bit or read from the RBUF register (thus clearing the RBFL bit).

8.7 Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the USART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1–16 (increments of 0.5) prescaler and an 11-bit binary counter. (*Figure 24*). The divide factors are specified through two read/write registers shown in *Figure 25*. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in *Table 5*, a Prescaler Factor of 0 corresponds to NO CLOCK. This condition is the USART power down mode where the USART clock is turned off for power saving purpose. The user must also turn the USART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in *Table 5*. There are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a 16x clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (*Table 4*). Other baud rates may be created by using appropriate divisors. The 16x clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

8.0 USART (Continued)

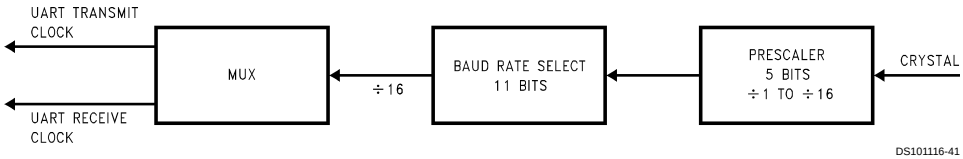


FIGURE 24. USART BAUD Clock Generation

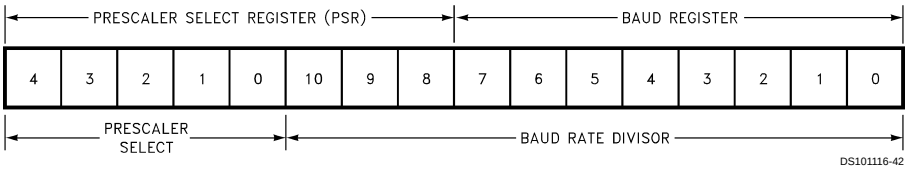


FIGURE 25. USART BAUD Clock Divisor Registers

TABLE 4. Baud Rate Divisors (1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor − 1 (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

Note: The entries in *Table 5* assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 987.5k.

TABLE 5. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

8.0 USART (Continued)

As an example, considering Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608/1.8432 = 2.5$$

The 2.5 entry is available in *Table 5*. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (*Table 4*) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in *Table 4* is 5.

$$N - 1 = 5 \text{ (N} - 1 \text{ is the value from } \textit{Table 4}\text{)}$$

$$N = 6 \text{ (N is the Baud Rate Divisor)}$$

$$\text{Baud Rate} = 1.8432 \text{ MHz}/(16 \times 6) = 19200$$

The divide by 16 is performed because in the asynchronous mode, the input frequency to the USART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc/(16 \times N \times P)$$

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is the Baud Rate Divisor (*Table 4*).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (*Table 5*)

Note: In the Synchronous Mode, the divisor 16 is replaced by two.

Example:

Asynchronous Mode:

Crystal Frequency = 5 MHz

Desired baud rate = 9600

Using the above equation N x P can be calculated first.

$$N \times P = (5 \times 10^6)/(16 \times 9600) = 32.552$$

Now 32.552 is divided by each Prescaler Factor (*Table 5*) to obtain a value closest to an integer. This factor happens to be 6.5 (P = 6.5).

$$N = 32.552/6.5 = 5.008 \text{ (N} = 5\text{)}$$

The programmed value (from *Table 4*) should be 4 (N − 1).

Using the above values calculated for N and P:

$$BR = (5 \times 10^6)/(16 \times 5 \times 6.5) = 9615.384$$

$$\% \text{ error} = (9615.385 - 9600)/9600 \times 100 = 0.16\%$$

8.8 Effect of HALT/IDLE

The USART logic is reinitialized when either the HALT or IDLE modes are entered. This reinitialization sets the TBMT flag and resets all read only bits in the USART control and status registers. Read/Write bits remain unchanged. The Transmit Buffer (TBUF) is not affected, but the Transmit Shift register (TSFT) bits are set to one. The receiver registers RBUF and RSFT are not affected.

The device will exit from the HALT/IDLE modes when the Start bit of a character is detected at the RDX (L3) pin. This feature is obtained by using the Multi-Input Wakeup scheme provided on the device.

Before entering the HALT or IDLE modes the user program must select the Wakeup source to be on the RDX pin. This selection is done by setting bit 3 of WKEN (Wakeup Enable) register. The Wakeup trigger condition is then selected to be high to low transition. This is done via the WKEDG register (Bit 3 is one.)

If the device is halted and crystal oscillator is used, the Wakeup signal will not start the chip running immediately because of the finite start up time requirement of the crystal oscillator. The idle timer (T0) generates a fixed (256 t_c) delay to ensure that the oscillator has indeed stabilized before allowing the device to execute code. The user has to consider this delay when data transfer is expected immediately after exiting the HALT mode.

8.9 Diagnostic

Bits CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the USART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Register is "looped back" into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the USART.

Note that the framing format for this mode is the nine bit format; one Start bit, nine data bits, and 7/8, one or two Stop bits. Parity is not generated or verified in this mode.

8.10 Attention Mode

The USART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the device with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the USART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the USART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if USART Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the USART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

9.0 Comparators

The device contains two differential comparators, each with a pair of inputs (positive and negative) and an output. Ports F1–F3 and F4–F6 are used for the comparators. The following is the Port F assignment:

F6 Comparator2 output
F5 Comparator2 positive input
F4 Comparator2 negative input
F3 Comparator1 output
F2 Comparator1 positive input
F1 Comparator1 negative input

A Comparator Select Register (CMPSL) is used to enable the comparators, read the outputs of the comparators internally, and enable the outputs of the comparators to the pins. Two control bits (enable and output enable) and one result bit are associated with each comparator. The comparator result bits (CMP1RD and CMP2RD) are read only bits which will read as zero if the associated comparator is not enabled. The Comparator Select Register is cleared with reset, resulting in the comparators being disabled. The comparators should also be disabled before entering either the HALT or IDLE modes in order to save power. The configuration of the CMPSL register is as follows:

CMPSL REGISTER (ADDRESS X'00B7)

Reserved	CMP20E	CMP2RD	CMP2EN	CMP10E	CMP1RD	CMP1EN	Reserved
Bit 7							Bit 0

The CMPSL register contains the following bits:

Reserved These bits are reserved and must be zero

CMP20E Selects pin I6 as comparator 2 output provided that CMP2EN is set to enable the comparator

CMP2RD Comparator 2 result (this is a read only bit, which will read as 0 if the comparator is not enabled)

CMP2EN Enable comparator 2

CMP10E Selects pin I3 as comparator 1 output provided that CMPIEN is set to enable the comparator

CMP1RD Comparator 1 result (this is a read only bit, which will read as 0 if the comparator is not enabled)

CMP1EN Enable comparator 1

Note that the two unused bits of CMPSL may be used as software flags.

Note: For compatibility with existing code and with existing Mask ROMMed devices the bits of the CMPSL register will take precedence over the associated Port F configuration and data output bits.

10.0 Interrupts

10.1 INTRODUCTION

Each device supports thirteen vectored interrupts. Interrupt sources include Timer 0, Timer 1, Timer 2, Timer 3, Port L Wakeup, 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.

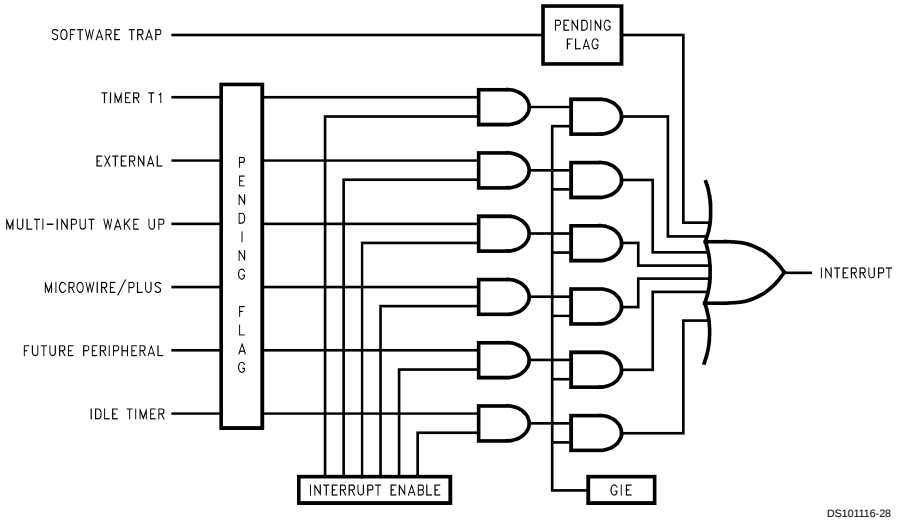


FIGURE 26. Interrupt Block Diagram

10.0 Interrupts (Continued)

10.2 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 asychronous 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 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 corresponding 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 sets 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.

10.3 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 6 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 ex-

10.0 Interrupts (Continued)

ample, 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, (50µs 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.

TABLE 6. Interrupt Vector Table

Arbitration Ranking	Source	Description	Vector Address (Note 15) (Hi-Low Byte)
(1) Highest	Software	INTR Instruction	0yFE–0yFF
(2)	Reserved		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)	Reserved		0yF0–0yF1
(9)	USART	Receive	0yEE–0yEF
(10)	USART	Transmit	0yEC–0yED
(11)	Timer T2	T2A/Underflow	0yEA–0yEB
(12)	Timer T2	T2B	0yE8–0yE9
(13)	Timer T3	T2A/Underflow	0yE6–0yE7
(14)	Timer T3	T3B	0yE4–0yE5
(15)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(16) Lowest	Default VIS	Reserved	0yE0–0yE1

Note 15: 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.

10.0 Interrupts (Continued)

10.3.1 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 the only active interrupt is software trap, than 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.

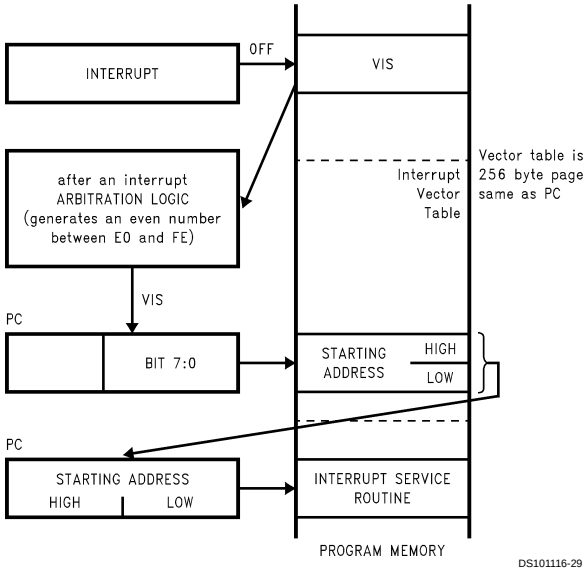


FIGURE 27. VIS Operation

10.0 Interrupts (Continued)

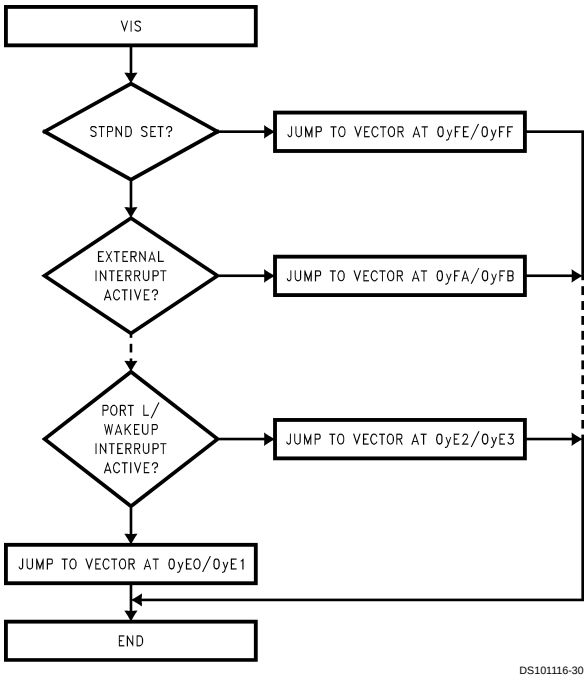


FIGURE 28. VIS Flowchart

10.0 Interrupts (Continued)

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           EXEN, PSW      ; Enable the external interrupt
           SBIT           GIE, PSW       ; Set the GIE bit
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
           .
           .
INT_EXIT:
           RETI
           .
           .
SERVICE:   RBIT    EXPND, PSW           ; Interrupt Service Routine
                                         ; Reset ext interrupt pend. bit
           .
           .
           .
           JP   INT_EXIT                 ; Return, set the GIE bit
```

10.0 Interrupts (Continued)

10.4 NON-MASKABLE INTERRUPT

10.4.1 Pending Flag

There is a pending flag bit associated with the non-maskable 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.

10.4.2 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 available program memory space, the nonexistent or unused memory location returns zeroes which is interpreted as the INTR instruction.

If the stack is popped beyond the allowed limit (address 06F Hex), a 7FFF will be loaded into the PC, if this last location in program memory is unprogrammed or unavailable, a Software Trap will be triggered.

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 reinitialize the stack pointer and perform a recovery procedure that restarts 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 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.

10.5 PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches 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 L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is 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. (See HALT MODE for clock option wakeup information.)

10.6 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 restart 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 routiness if the time between interrupts is short. In this case the RETI instruction would only be executed when the default VIS routine is reached.

11.0 WATCHDOG/Clock Monitor

Each device contains a user selectable WATCHDOG and clock monitor. The following section is applicable only if WATCHDOG feature has been selected in the ECON register. 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 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 7* shows the WDSVR register.

TABLE 7. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

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 8 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 8. WATCHDOG Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Clock Monitor	Service Window (Lower-Upper Limits)
0	0	x	2048–8k t_C Cycles
0	1	x	2048–16k t_C Cycles
1	0	x	2048–32k t_C Cycles
1	1	x	2048–64k t_C Cycles
x	x	0	Clock Monitor Disabled
x	x	1	Clock Monitor Enabled

11.1 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.

11.2 WATCHDOG/CLOCK MONITOR OPERATION

The WATCHDOG is enabled by bit 2 of the ECON register. When this ECON 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 WATCHDOG 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 9* 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 WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low and must be externally connected to the RESET pin or to some other external logic which handles WATCHDOG event. The WDOUT 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 WDOUT (G1) pin low for an additional 16 t_C–32 t_C cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the device will stop forcing the WDOUT output low. The WATCHDOG service window will restart when the WDOUT pin goes high.

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

$1/t_C > 10$ kHz — No clock rejection.

$1/t_C < 10$ Hz — Guaranteed clock rejection.

11.0 WATCHDOG/Clock Monitor (Continued)

TABLE 9. 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

11.3 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.
- 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.

11.4 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. It is recommended that the user either leave this location unprogrammed or place an INTR instruction (all 0's) in this location to 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.

12.0 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 10* details the different clock rates that may be selected.

TABLE 10. MICROWIRE/PLUS Master Mode Clock Select

SL1	SL0	SK Period
0	0	2 x t_C
0	1	4 x t_C
1	x	8 x t_C

Where t_C is the instruction cycle clock

12.1 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 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.

12.1.1 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 11* summarizes the bit settings required for Master mode of operation.

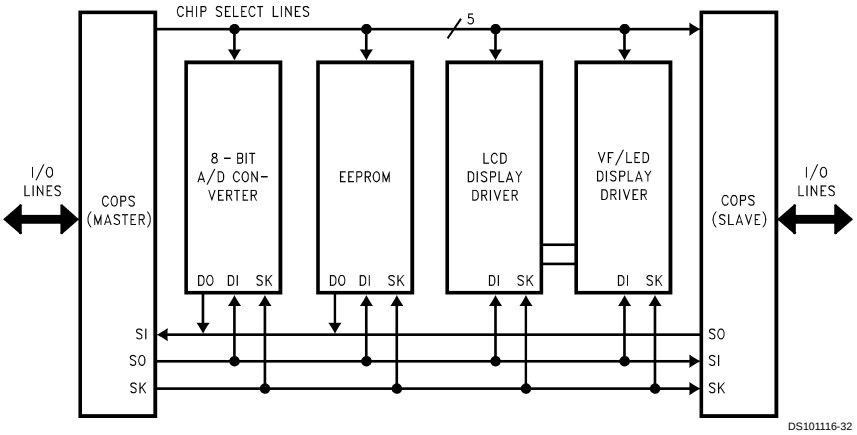


FIGURE 29. MICROWIRE/PLUS Application

12.0 MICROWIRE/PLUS (Continued)

12.1.2 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 11* summarizes the settings required to enter the Slave mode of operation.

TABLE 11. MICROWIRE/PLUS Mode Settings

This table assumes that the control flag MSEL is set.

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.

12.1.3 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. 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.

TABLE 12. 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

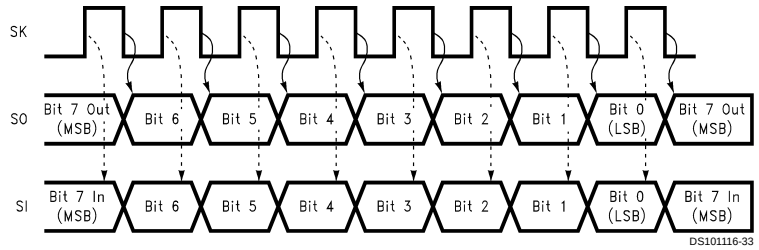


FIGURE 30. MICROWIRE/PLUS SPI Mode Interface Timing, Normal SK Mode, SK Idle Phase being Low

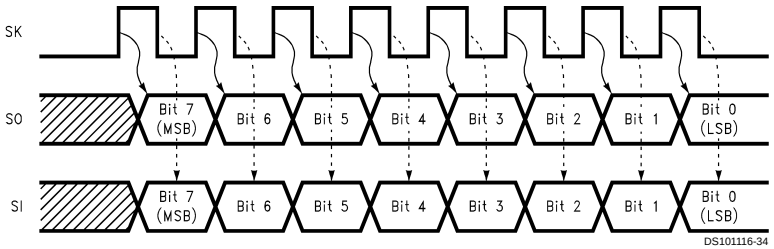


FIGURE 31. MICROWIRE/PLUS SPI Mode Interface Timing, Alternate SK Mode, SK Idle Phase being Low

12.0 MICROWIRE/PLUS (Continued)

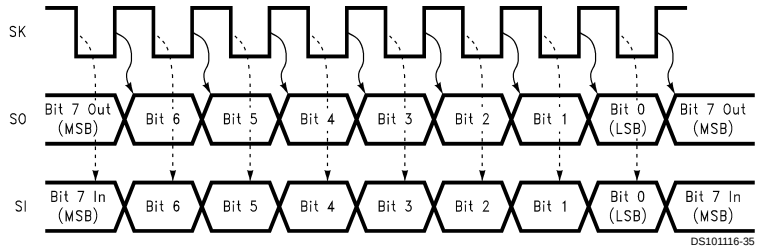


FIGURE 32. MICROWIRE/PLUS SPI Mode Interface Timing, Alternate SK Mode, SK Idle Phase being High

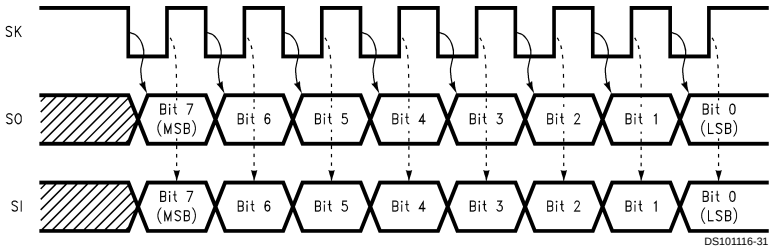


FIGURE 33. MICROWIRE/PLUS SPI Mode Interface Timing, Normal SK Mode, SK Idle Phase being High

13.0 Memory Map

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

Address S/ADD REG	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xx93	Unused RAM Address Space (Reads Undefined Data)
xx94	Port F data register, PORTFD
xx95	Port F configuration register, PORTFC
xx96	Port F input pins (read only), PORTFP
xx97 to xxAF	Unused address space (Reads Undefined Data)
xxB0	Timer T3 Lower Byte
xxB1	Timer T3 Upper Byte
xxB2	Timer T3 Autoload Register T3RA Lower Byte
xxB3	Timer T3 Autoload Register T3RA Upper Byte
xxB4	Timer T3 Autoload Register T3RB Lower Byte
xxB5	Timer T3 Autoload Register T3RB Upper Byte
xxB6	Timer T3 Control Register
xxB7	Comparator Select Register (Reg:CMPSL)
xxB8	UART Transmit Buffer (Reg:TBUF)
xxB9	UART Receive Buffer (Reg:RBUF)
xxBA	UART Control and Status Register (Reg:ENU)
xxBB	UART Receive Control and Status Register (Reg:ENUR)
xxBC	UART Interrupt and Clock Source Register (Reg:ENUI)
xxBD	UART Baud Register (Reg:BAUD)
xxBE	UART Prescale Select Register (Reg:PSR)
xxBF	Reserved for UART
xxC0	Timer T2 Lower Byte
xxC1	Timer T2 Upper Byte
xxC2	Timer T2 Autoload Register T2RA Lower Byte
xxC3	Timer T2 Autoload Register T2RA Upper Byte
xxC4	Timer T2 Autoload Register T2RB Lower Byte
xxC5	Timer T2 Autoload Register T2RB Upper Byte
xxC6	Timer T2 Control Register
xxC7	WATCHDOG Service Register (Reg:WDSVR)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB to xxCF	Reserved
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only) (Actually reads Port F input pins)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to xxDF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE/PLUS Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100–017F	On-Chip 128 RAM Bytes
0200–027F	On-Chip 128 RAM Bytes (Reads as undefined data on COP8FGE)
0300–037F	On-Chip 128 RAM Bytes (Reads as undefined data on COP8FGE)

Note: Reading memory locations 0070H–007FH (Segment 0) will return all ones. Reading unused memory locations 0080H–0093H (Segment 0) will return undefined data. Reading memory locations from other Segments (i.e., Segment 4, Segment 5, … etc.) will return undefined data.

14.0 Instruction Set

14.1 INTRODUCTION

This section defines the instruction set of the COPSAx7 Family members. It contains information about the instruction set features, addressing modes and types.

14.2 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.

14.3 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: 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.

14.3.1 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 Memory	Contents Before	Contents 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 Memory	Contents Before	Contents 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 Memory	Contents Before	Contents 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.

14.0 Instruction Set (Continued)

Example: Load Accumulator Immediate
LD A,#05

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

14.3.2 Tranfer-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

14.0 Instruction Set (Continued)

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 0126 Hex	32 Hex	32 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.

14.4 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.

14.4.1 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 (SUB)
- Subtract with Carry (SUBC)
- Increment (INC)
- Decrement (DEC)
- Decimal Correct (DCOR)
- Clear Accumulator (CLR)
- Set Carry (SC)
- Reset Carry (RC)

14.4.2 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)
- Jump Absolute Long (JMPL)
- Jump Indirect (JID)
- Jump to Subroutine (JSR)
- Jump to Subroutine Long (JSRL)
- Return from Subroutine (RET)
- Return from Subroutine and Skip (RETSK)
- Return from Interrupt (RETI)
- Software Trap Interrupt (INTR)
- Vector Interrupt Select (VIS)

14.4.3 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)

14.4.4 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 exclusiveORing the Accumulator with FF Hex.

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

14.4.5 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)

14.4.6 Stack Control Instructions

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

14.4.7 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)

14.4.8 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)

14.0 Instruction Set (Continued)

14.4.9 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.

14.5 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
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC

Registers	
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

14.6 INSTRUCTION SET SUMMARY

ADD	A,Meml	ADD	A←A + Meml
ADC	A,Meml	ADD with Carry	A←A + Meml + C, C←Carry, HC←Half Carry
SUBC	A,Meml	Subtract with Carry	A←A − $\overline{\text{Meml}}$ + C, C←Carry, HC←Half Carry
AND	A,Meml	Logical AND	A←A and $\overline{\text{Meml}}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A←A or Meml
XOR	A,Meml	Logical EXclusive OR	A←A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A ≠ Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg←Reg − 1, Skip if 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↔Mem
X	A,[X]	EXchange A with Memory [X]	A↔[X]
LD	A,Meml	LoaD A with Memory	A←Meml
LD	A,[X]	LoaD A with Memory [X]	A←[X]
LD	B,Imm	LoaD B with Immed.	B←Imm
LD	Mem,Imm	LoaD Memory Immed.	Mem←Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg←Imm
X	A, [B ±]	EXchange A with Memory [B]	A↔[B], (B←B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A↔[X], (X←X ±1)

14.0 Instruction Set (Continued)

LD	A, [B±]	LoaD A with Memory [B]	A←[B], (B←B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A←[X], (X←X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B]←Imm, (B←B±1)
CLR	A	CLeaR A	A←0
INC	A	INCrement A	A←A + 1
DEC	A	DECrement A	A←A − 1
LAID		Load A InDirect from ROM	A←ROM (PU,A)
DCOR	A	Decimal CORrect A	A←BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C→A7→...→A0→C
RLC	A	Rotate A Left thru C	C←A7←...←A0←C, HC←A0
SWAP	A	SWAP nibbles of A	A7...A4↔A3...A0
SC		Set C	C←1, HC←1
RC		Reset C	C←0, HC←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	SP←SP + 1, A←[SP]
PUSH	A	PUSH A onto the stack	[SP]←A, SP←SP − 1
VIS		Vector to Interrupt Service Routine	PU←[VU], PL←[VL]
JMPL	Addr.	Jump absolute Long	PC←ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9...0←i (i = 12 bits)
JP	Disp.	Jump relative short	PC←PC + r (r is −31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP]←PL, [SP−1]←PU,SP−2, PC←ii
JSR	Addr.	Jump SubRoutine	[SP]←PL, [SP−1]←PU,SP−2, PC9...0←i
JID		Jump InDirect	PL←ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL←[SP], PU←[SP−1]
RETSK		RETurn and SKip	SP + 2, PL←[SP],PU←[SP−1], skip next instruction
RETI		RETurn from Interrupt	SP + 2, PL ←[SP],PU←[SP−1],GIE←1
INTR		Generate an Interrupt	[SP]←PL, [SP−1]←PU, SP−2, PC←0FF
NOP		No OPeration	PC←PC + 1

14.0 Instruction Set (Continued)

14.7 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
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A, (Note 16)	1/1	1/3	2/3		1/2	1/3	
LD A, (Note 16)	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 16: = > Memory location addressed by B or X or directly.

14.8 OPCODE TABLE

Upper Nibble																Lower 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]	*	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]	*	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]	*	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+]	LD [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−]	LD [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

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

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

15.0 Mask Options

See Section 5.5 ECON (CONFIGURATION) REGISTER.

16.0 Development Support

16.1 OVERVIEW

National is engaged with an international community of independent 3rd party vendors who provide hardware and software development tool support. Through National's interaction and guidance, these tools cooperate to form a choice of tools that fits each developer's needs.

This section provides a summary of the tool and development kits currently available. Up-to-date information, selection guides, free tools, demos, updates, and purchase information can be obtained at our web site at: www.national.com/cop8.

16.2 SUMMARY OF TOOLS

COP8 Evaluation Tools

- **COP8–NSEVAL:** Free Software Evaluation package for Windows. A fully integrated evaluation environment for COP8, including versions of WCOP8 IDE (Integrated Development Environment), COP8-NSASM, COP8-MLSIM, COP8C, DriveWay™ COP8, Manuals, and other COP8 information.
- **COP8–MLSIM:** Free Instruction Level Simulator tool for Windows. For testing and debugging software instructions only (No I/O or interrupt support).
- **COP8–EPU:** Very Low cost COP8 Evaluation & Programming Unit. Windows based evaluation and hardware-simulation tool, with COP8 device programmer and erasable samples. Includes COP8-NSDEV, Driveway COP8 Demo, MetaLink Debugger, I/O cables and power supply.
- **COP8–EVAL-ICUxx:** Very Low cost evaluation and design test board for COP8ACC and COP8SGx Families, from ICU. Real-time environment with add-on A/D, D/A, and EEPROM. Includes software routines and reference designs.
- **Manuals, Applications Notes, Literature:** Available free from our web site at: www.national.com/cop8.

COP8 Integrated Software/Hardware Design Development Kits

- **COP8-EPU:** Very Low cost Evaluation & Programming Unit. Windows based development and hardware-simulation tool for COPSx/xG families, with COP8 device programmer and samples. Includes COP8-NSDEV, Driveway COP8 Demo, MetaLink Debugger, cables and power supply.
- **COP8-DM:** Moderate cost Debug Module from MetaLink. A Windows based, real-time in-circuit emulation tool with COP8 device programmer. Includes COP8-NSDEV, DriveWay COP8 Demo, MetaLink Debugger, power supply, emulation cables and adapters.

COP8 Development Languages and Environments

- **COP8-NSASM:** Free COP8 Assembler v5 for Win32. Macro assembler, linker, and librarian for COP8 software development. Supports all COP8 devices. (DOS/Win16 v4.10.2 available with limited support). (Compatible with WCOP8 IDE, COP8C, and DriveWay COP8).
- **COP8-NSDEV:** Very low cost Software Development Package for Windows. An integrated development environment for COP8, including WCOP8 IDE, COP8-NSASM, COP8-MLSIM.
- **COP8C:** Moderately priced C Cross-Compiler and Code Development System from Byte Craft (no code limit). Includes BCLIDE (Byte Craft Limited Integrated Development Environment) for Win32, editor, optimizing C Cross-Compiler, macro cross assembler, BC-Linker, and MetaLink tools support. (DOS/SUN versions available; Compiler is installable under WCOP8 IDE; Compatible with DriveWay COP8).
- **EWCOP8-KS:** Very Low cost ANSI C-Compiler and Embedded Workbench from IAR (Kickstart version: COP8Sx/Fx only with 2k code limit; No FP). A fully integrated Win32 IDE, ANSI C-Compiler, macro assembler, editor, linker, Liberian, C-Spy simulator/debugger, PLUS MetaLink EPU/DM emulator support.
- **EWCOP8-AS:** Moderately priced COP8 Assembler and Embedded Workbench from IAR (no code limit). A fully integrated Win32 IDE, macro assembler, editor, linker, librarian, and C-Spy high-level simulator/debugger with I/O and interrupts support. (Upgradeable with optional C-Compiler and/or MetaLink Debugger/Emulator support).
- **EWCOP8-BL:** Moderately priced ANSI C-Compiler and Embedded Workbench from IAR (Baseline version: All COP8 devices; 4k code limit; no FP). A fully integrated Win32 IDE, ANSI C-Compiler, macro assembler, editor, linker, librarian, and C-Spy high-level simulator/debugger. (Upgradeable; CWCOP8-M MetaLink tools interface support optional).
- **EWCOP8:** Full featured ANSI C-Compiler and Embedded Workbench for Windows from IAR (no code limit). A fully integrated Win32 IDE, ANSI C-Compiler, macro assembler, editor, linker, librarian, and C-Spy high-level simulator/debugger. (CWCOP8-M MetaLink tools interface support optional).
- **EWCOP8-M:** Full featured ANSI C-Compiler and Embedded Workbench for Windows from IAR (no code limit). A fully integrated Win32 IDE, ANSI C-Compiler, macro assembler, editor, linker, librarian, C-Spy high-level simulator/debugger, PLUS MetaLink debugger/hardware interface (CWCOP8-M).

COP8 Productivity Enhancement Tools

- **WCOP8 IDE:** Very Low cost IDE (Integrated Development Environment) from KKD. Supports COP8C, COP8-NSASM, COP8-MLSIM, DriveWay COP8, and MetaLink debugger under a common Windows Project Management environment. Code development, debug, and emulation tools can be launched from the project window framework.
- **DriveWay-COP8:** Low cost COP8 Peripherals Code Generation tool from Aisys Corporation. Automatically generates tested and documented C or Assembly source code modules containing I/O drivers and interrupt handlers for each on-chip peripheral. Application specific code can be inserted for customization using the integrated editor. (Compatible with COP8-NSASM, COP8C, and WCOP8 IDE.)
- **COP8-UTILS:** Free set of COP8 assembly code examples, device drivers, and utilities to speed up code development.

16.0 Development Support (Continued)

- **COP8-MLSIM:** Free Instruction Level Simulator tool for Windows. For testing and debugging software instructions only (No I/O or interrupt support).

COP8 Real-Time Emulation Tools

- **COP8-DM:** MetaLink Debug Module. A moderately priced real-time in-circuit emulation tool, with COP8 device programmer. Includes MetaLink Debugger, power supply, emulation cables and adapters.
- **IM-COP8:** MetaLink iceMASTER®. A full featured, real-time in-circuit emulator for COP8 devices. Includes COP8-NSDEV, Driveway COP8 Demo, MetaLink Windows Debugger, and power supply. Package-specific probes and surface mount adaptors are ordered separately.

COP8 Device Programmer Support

- MetaLink's EPU and Debug Module include development device programming capability for COP8 devices.
- Third-party programmers and automatic handling equipment cover needs from engineering prototype and pilot production, to full production environments.
- Factory programming available for high-volume requirements.

16.3 TOOLS ORDERING NUMBERS FOR THE COP8FGx FAMILY DEVICES

The COP8FGx devices are faster speed versions of the COP8SGx devices, and the existing SGx tools can be used without updating or modification (just use the SGx menus). The COP8SG-DM and IM-COP8/400 ICE can be used for emulation with the limitation of 10 MHz emulation speed maximum. For full speed COP8FGx emulation, use the 15 MHz COP8FG-DM.

Note: The following order numbers apply to the COP8 devices in this datasheet only.

Vendor	Tools	Order Number	Cost	Notes
National	COP8-NSEVAL	COP8-NSEVAL	Free	Web site download
	COP8-NSASM	COP8-NSASM	Free	Included in EPU and DM. Web site download
	COP8-MLSIM	COP8-MLSIM	Free	Included in EPU and DM. Web site download
	COP8-NSDEV	COP8-NSDEV	VL	Included in EPU and DM. Order CD from website
	COP8-EPU	COP8SG-EPU (-1 or -2)	VL	-1 = 110V, -2 = 220V; Included p/s, 40 pin DIP target cable, manuals, software, 16/20/28/40 DIP OTP programming socket; add DM target adapter or OTP adapter (if needed)
	COP8-DM	COP8SG-DM (10 MHz)	M	Included p/s, 28/40/44 pin DIP/SO/PLCC target cables, manuals, software, 16/20/28/40 DIP/SO and 44 PLCC programming socket; add OTP adapter or target adapter (if needed)
	DM Target Adapters	DM-COP8/20D-SO	VL	20 pin DIP to SO converter
		DM-COP8/20D-16D	VL	20 pin DIP to 16 pin DIP converter
		DM-COP8/20D	VL	20 pin DIP target cable
		DM-COP8/28D-28CSP	L	28 pin DIP to 28 pin CSP converter
		DM-COP8/44P-44Q	L	44 pin PLCC to 44 QFP converter
	Development Devices	COP8FGx7	VL	8k or 32k Eraseable or OTP devices
	OTP Programming Adapters	COP8SA-PGMA	L	For programming 16/20/28 SOIC and 44 PLCC on the EPU
		COP8-PGMA-44QFP	L	For programming 44 QFP on any programmer
		COP8-PGMA-28CSP	L	For programming 28 CSP on any programmer
		COP8-PGMA-28SO	VL	For programming 16/20/28 SOIC on any programmer
	IM-COP8	Call MetaLink		

16.0 Development Support (Continued)

MetaLink	COP8-EPU	EPU-COP8SG	VL	1 = 110V, 2 = 220V; included p/s, 40 pin DIP target cable, manuals, software, 16/20/28/40 DIP OTP programming socket; add DM target adapter or OTP adapter (if needed)
	COP8-DM	DM5-COP8-FGx (15 MHz) or DM4-COP8-SGx (10 MHz), plus PS-10, plus DM-COP8/xxx (ie. 28D)	M	Included p/s (PS-10), target cable of choice (DIP or PLCC; i.e. DM-COP8/28D), 16/20/28/40 DIP/SO and 44 PLCC programming sockets. Add OTP adapter (if needed) and target adapter (if needed)
	DM Target Adapters	MHW-CNVxx (xx = 33, 34 etc.)	L	DM target converters for 16DIP/20SO/28SO/44QFP/28CSP; (i.e. MHW-CNV38 for 20 pin DIP to SO package converter)
	OTP Programming Adapters	MHW-COP8-PGMA-DS	L	For programming 16/20/28 SOIC and 44 PLCC on the EPU
		MHW-COP8-PGMA-44QFP	L	For programming 44 QFP on any programmer
		MHW-COP8-PGMA-28CSP	L	For programming 28 CSP on any programmer
	IM-COP8	IM-COP8-AD-464 (-220) (10 MHz maximum)	H	Base unit 10 MHz; -220 = 220V; add probe card (required) and target adapter (if needed); included software and manuals
	IM Probe Card	PC-COP8SG44PW-AD-10	M	10 MHz 44 PLCC probe card; 2.5V to 6.0V
		PC-COP8SG40DW-AD-10	M	10 MHz 40 DIP probe card; 2.5V to 6.0V
	IM Probe Target Adapters	MHW-SOICxx (xx = 16, 20, 28)	L	16 or 20 or 28 pin SOIC adapter for probe card
		MHW-CSPxx (xx = 20, 28)	L	20 or 28 pin CSP adapter for probe card
		MHW-CONV33	L	44 pin QFP adapter for 44 PLCC probe card
ICU or National	COP8-EVAL-ICUxx	ICU-303	L	No power supply
		COP8-EVAL-ICUSG	L	No power supply
KKD	WCOP8-IDE	WCOP8-IDE	VL	Included in EPU and DM
IAR	EWCOP8-xx	See summary above	L - H	Included all software and manuals
Byte Craft	COP8C	COP8C	M	Included all software and manuals
Aisys	DriveWay COP8	DriveWay COP8	L	Included all software and manuals
OTP Programmers		Go to: www.national.com/cop8	L - H	A wide variety world-wide
Cost: Free; VL =< $100; L = $100 - $300; M = $300 - $1k; H = $1k - $3k; VH = $3k - $5k				

16.0 Development Support (Continued)

16.4 WHERE TO GET TOOLS

Tools are ordered directly from the following vendors. Please go to the vendor's web site for current listings of distributors.

Vendor	Home Office	Electronic Sites	Other Main Offices
Aisys	U.S.A.: Santa Clara, CA 1-408-327-8820 fax: 1-408-327-8830	www.aisysinc.com info@aisysinc.com	Distributors
Byte Craft	U.S.A. 1-519-888-6911 fax: 1-519-746-6751	www.bytecraft.com info@bytecraft.com	Distributors
IAR	Sweden: Uppsala +46 18 16 78 00 fax: +46 18 16 78 38	www.iar.se info@iar.se info@iar.com info@iarsys.co.uk info@iar.de	U.S.A.: San Francisco 1-415-765-5500 fax: 1-415-765-5503 U.K.: London +44 171 924 33 34 fax: +44 171 924 53 41 Germany: Munich +49 89 470 6022 fax: +49 89 470 956
ICU	Sweden: Polygonvaegen +46 8 630 11 20 fax: +46 8 630 11 70	www.icu.se support@icu.se support@icu.ch	Switzeland: Hoehe +41 34 497 28 20 fax: +41 34 497 28 21
KKD	Denmark:	www.kkd.dk	
MetaLink	U.S.A.: Chandler, AZ 1-800-638-2423 fax: 1-602-926-1198	www.metaice.com sales@metaice.com support@metaice.com bbs: 1-602-962-0013 www.metalink.de	Germany: Kirchseeon 80-91-5696-0 fax: 80-91-2386 islanger@metalink.de Distributors Worldwide
National	U.S.A.: Santa Clara, CA 1-800-272-9959 fax: 1-800-737-7018	www.national.com/cop8 support@nsc.com europe.support@nsc.com	Europe: +49 (0) 180 530 8585 fax: +49 (0) 180 530 8586 Distributors Worldwide

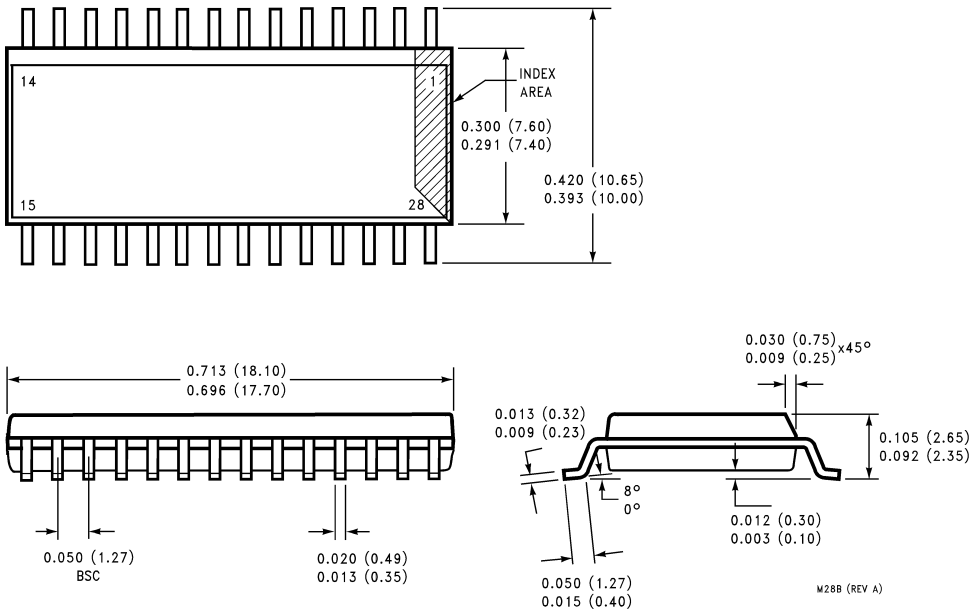
The following companies have approved COP8 programmers in a variety of configurations. Contact your local office or distributor. You can link to their web sites and get the latest listing of approved programmers from National's COP8 OTP Support page at: www.national.com/cop8.

Advantech; Advin; BP Microsystems; Data I/O; Hi-Lo Systems; ICE Technology; Lloyd Research; Logical Devices; MQP; Needhams; Phyton; SMS; Stag Programmers; System General; Tribal Microsystems; Xeltek.

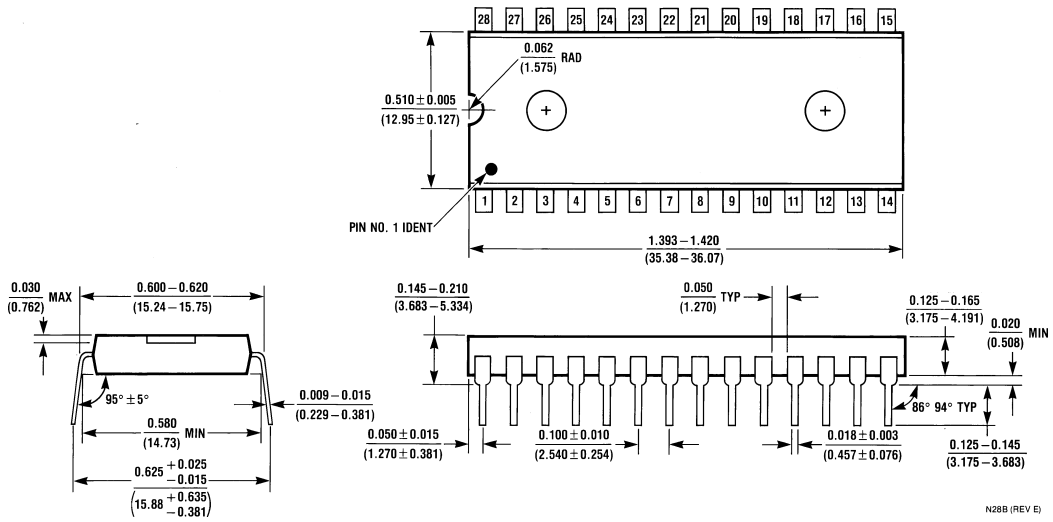
16.5 CUSTOMER SUPPORT

Complete product information and technical support is available from National's customer response centers, and from our on-line COP8 customer support sites.

Physical Dimensions inches (millimeters) unless otherwise noted

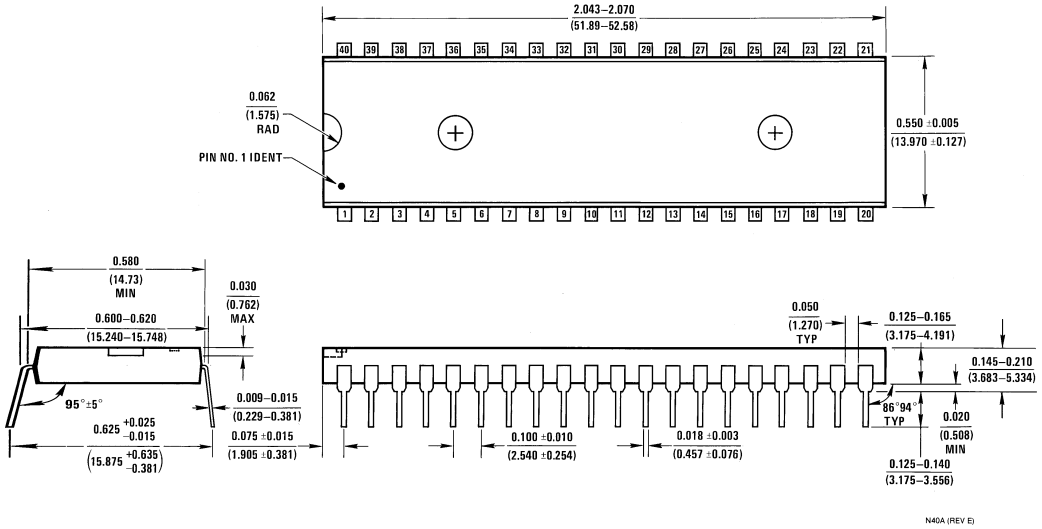


Molded SO Wide Body Package (WM)
Order Number COP8FGx528Mx,
NS Package Number M28B



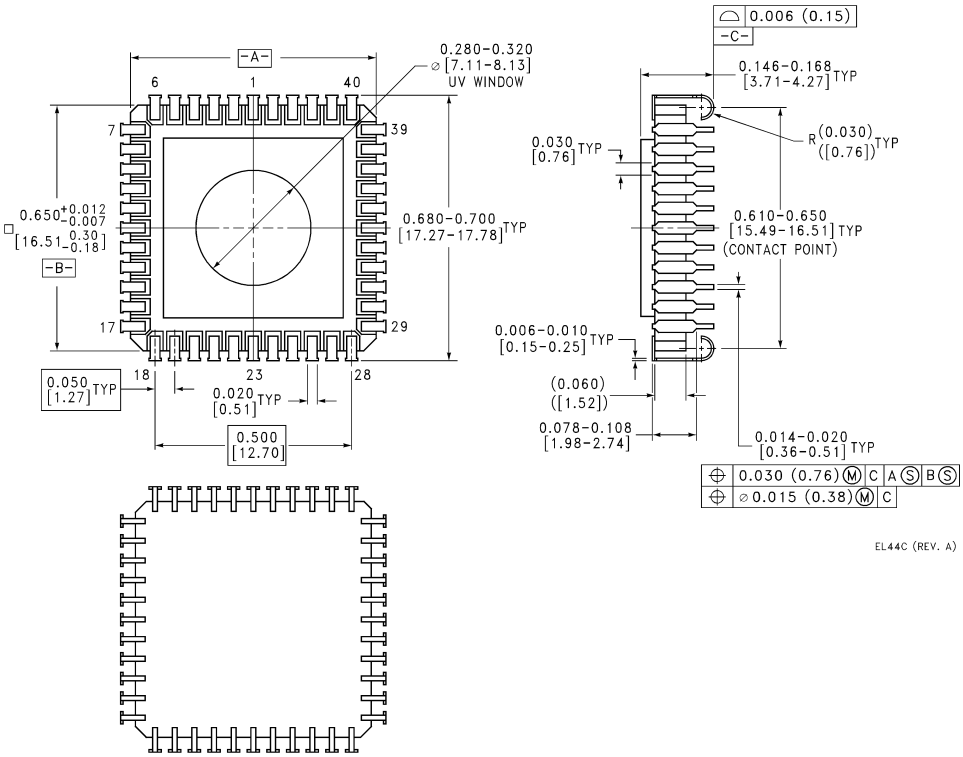
Molded Dual-In-Line Package (N)
Order Number COP8SGx728Nx
NS Package Number N28A

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



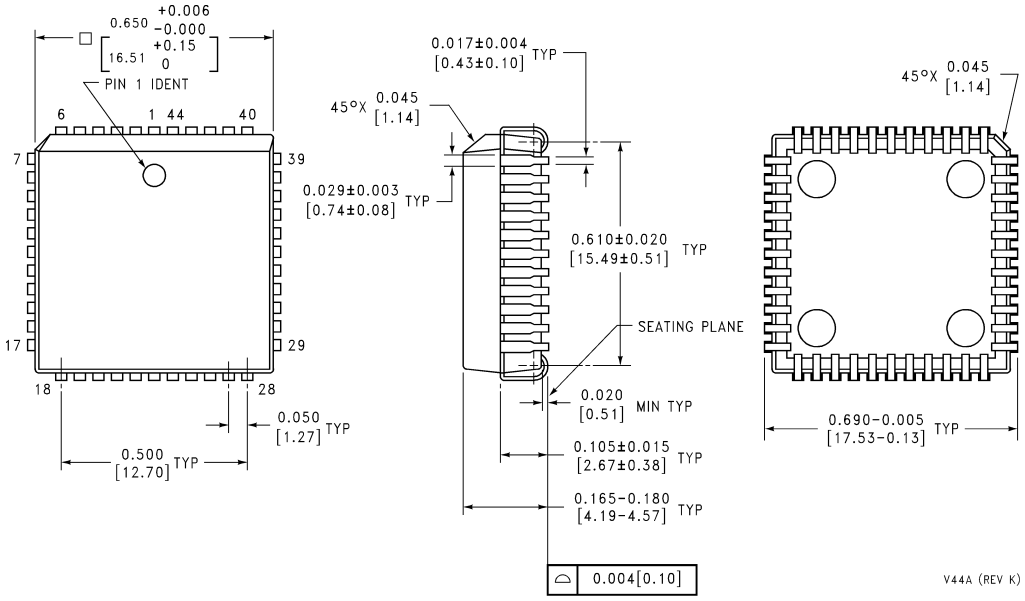
Molded Dual-In-Line Package (N)
Order Number COP8FGx540Nx
NS Package Number N40A

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

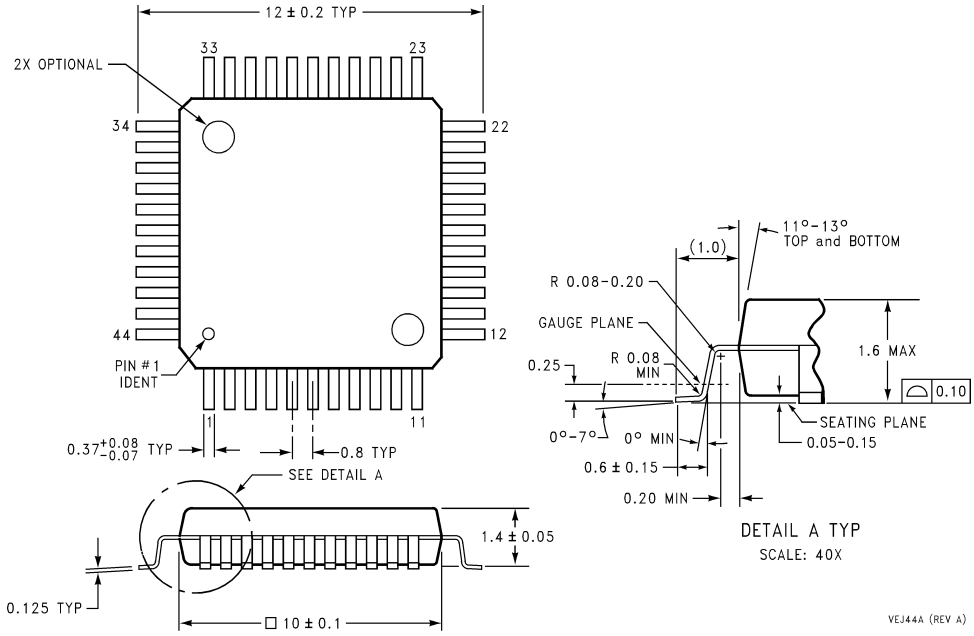


44-Lead EPROM Leaded Chip Carrier (EL)
Order Number COP8SGR744J3
NS Package Number EL44C

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



Molded Dual-In-Line Package (N)
Order Number COP8FGx544Vx
NS Package Number V44A



Plastic Quad Flat Package (VEJ)
Order Number COP8FGx544VEJx
NS Package Number VEJ44A

Notes

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.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507