National Semiconductor

September 1999

COP87L88FH
8-Bit CMOS OTP Microcontrollers with 16k Memory, Comparators, USART and Hardware Multiply/Divide

General Description

The COP87L88FH OTP (One Time Programmable) microcontrollers are highly integrated COP8™ Feature core devices with 16k memory and advanced features including Analog comparators, and Hardware Multiply/Divide. These multi-chip CMOS devices are suited for applications requiring a full featured controller with comparators, a full-duplex USART, and hardware multiply/divide functions, and as pre-production devices for a masked ROM design. Lower cost pin and software compatible 12k ROM versions are available (COP888FH), as well as a range of COP8 software and hardware development tools.

Family features include an 8-bit memory mapped architecture, 10 MHz CKI (-XE = crystal oscillator; -TE = external clock) with 1µs instruction cycle, hardware multiply/divide functions, 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, WATCHDOG™ and clock monitor logic, low EMI 2.7V to 5.5V operation, and 28/40/44 pin packages.

Devices included in this data sheet are:

Device	Memory (bytes)	RAM (bytes)	I/O Pins	Packages	Temperature
COP87L84FH	16k OTP EPROM	512	24	28 DIP/SOIC	-40 to +85˚C
COP87L88FH	16k OTP EPROM	512	36/40	40 DIP, 44 PLCC	-40 to +85˚C

Key Features

- Hardware Multiply/Divide Functions
- Full duplex USART
- Three 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- 16 kbytes on-board EPROM with security features
- 512 bytes on-board RAM

Additional Peripheral Features

- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Two analog comparators
- WATCHDOG and Clock Monitor logic
- MICROWIRE/PLUS serial I/O

I/O Features

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

CPU/Instruction Set Features

- 1 µs instruction cycle time
- Fourteen multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Three Timers (Each with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — USART (2)
  - — Default VIS (default interrupt)
- Versatile and easy to use instruction set
- 8-bit Stack Pointer (SP) — stack in RAM
- Two 8-bit Register Indirect Data Memory Pointers (B and X)

Fully Static CMOS

- Two power saving modes: HALT and IDLE
- Single supply operation: 2.7V–5.5V
- Temperature ranges: –40˚C to +85˚C

Development Support

- Emulation device for COP888FH
- Real time emulation and full program debug offered by MetaLink Development System

Block Diagram

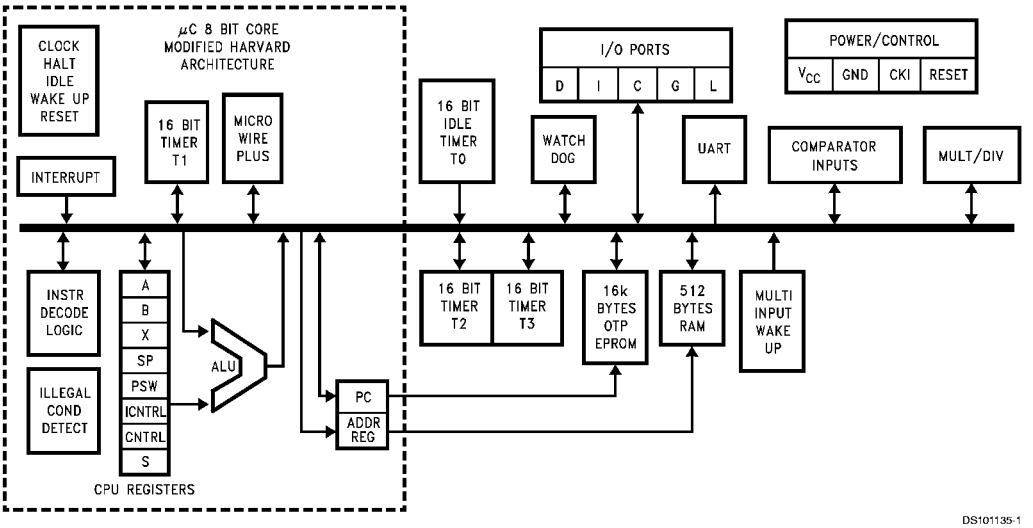
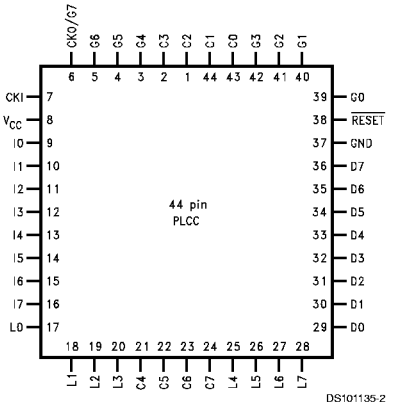


FIGURE 1. COP87L88FH Block Diagram

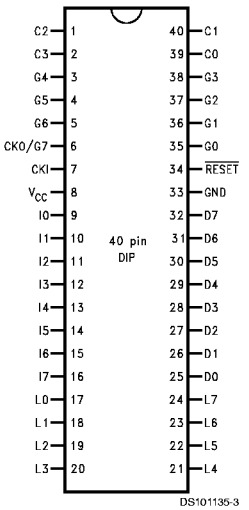
Connection Diagrams

Plastic Chip Carrier



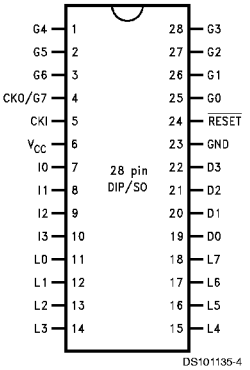
Top View
Order Number COP87L88FHV-XE/TE
See NS Plastic Chip Package Number V44A

Dual-In-Line Package



Top View Order
Number COP87L88FHN-XE/TE
See NS Molded Package Number N40A

Dual-In-Line Package



Order Number COP87L84FHM-XE/TE, or COP87L84FHN-XE/TE
See NS Molded Package Number M28B or N28B

Note: -X Crystal Oscillator
-T External Clock
-E Halt Mode Enable

FIGURE 2. Connection Diagrams

Connection Diagrams (Continued)

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

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O			19	25	29
D1	O			20	26	30
D2	O			21	27	31
D3	O			22	28	32
D4	O				29	33
D5	O				30	34
D6	O				31	35
D7	O				32	36
I0	I			7	9	9
I1	I	COMP1IN−		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN−			13	13
I5	I	COMP2IN+			14	14
I6	I	COMP2OUT			15	15
I7	I				16	16
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

Absolute Maximum Ratings (Note 1)

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

Supply Voltage (V_{CC})	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

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

DC Electrical Characteristics

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

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.7		5.5	V
Power Supply Ripple (Note 2)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 3)					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			12.5	mA
CKI = 4 MHz	V_{CC} = 5.5V, t_c = 2.5 µs			5.5	mA
HALT Current (Note 4)	V_{CC} = 5.5V, CKI = 0 MHz		<5	10	µA
	V_{CC} = 4.0V, CKI = 0 MHz		<3	6	µA
IDLE Current					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			3.5	mA
CKI = 4 MHz	V_{CC} = 5.5V, t_c = 2.5 µs			2.5	mA
Input Levels					
$\overline{\text{RESET}}$					
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
Hi-Z Input Leakage	V_{CC} = 5.5V, V_{IN} = 0V	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V, V_{IN} = 0V	−40		−250	µA
G and L Port Input Hysteresis	(Note 6)			0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	−0.4			mA
	V_{CC} = 2.7V, V_{OH} = 1.8V	−0.2			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1V	10			mA
	V_{CC} = 2.7V, V_{OL} = 0.4V	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	−10		−100	µA
	V_{CC} = 2.7V, V_{OH} = 1.8V	−2.5		−33	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	−0.4			mA
	V_{CC} = 2.7V, V_{OH} = 1.8V	−0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.7V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage	V_{CC} = 5.5V	−2		+2	µA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Notes 5, 6)	Room Temp			±200	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V

DC Electrical Characteristics (Continued)

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

Parameter	Conditions	Min	Typ	Max	Units
Input Capacitance	(Note 6)			7	pF
Load Capacitance on D2	(Note 6)			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 or External	$2.7V \le V_{CC} \le 4.5V$	2.5		DC	μs
	$4.5V \le V_{CC} \le 5.5V$	1.0		DC	μs
R/C Oscillator	$2.7V \le V_{CC} < 4.0V$	7.5		DC	μs
	$4.5V \le V_{CC} \le 5.5V$	3.0		DC	μs
Inputs					
t_{SETUP}	$4.5V \le V_{CC} \le 5.5V$	200			ns
	$2.7V \le V_{CC} < 4.5V$	500			ns
t_{HOLD}	$4.5V \le V_{CC} \le 5.5V$	60			ns
	$2.7V \le V_{CC} < 4.5V$	150			ns
Output Propagation Delay	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
	$2.7V \le V_{CC} < 4.5V$			1.75	μs
All Others	$4.5V \le V_{CC} \le 5.5V$			1	μs
	$2.7V \le V_{CC} < 4.5V$			2.5	μs
MICROWIRE Setup Time (t_{UWS}) (Note 6)	$V_{CC} \ge 4.5V$	20			ns
MICROWIRE Hold Time (t_{UWH}) (Note 6)	$V_{CC} \ge 4.5V$	56			ns
MICROWIRE Output Propagation Delay (t_{UPD})	$V_{CC} \ge 4.5V$			220	ns
Input Pulse Width (Note 7)					
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 2: Maximum rate of voltage change must be less than 0.5V/ms.

Note 3: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 4: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Measurement of I_{DD} HALT is done with device neither sourcing or sinking current; with L, C, and G0–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 and comparators disabled. Parameter refers to HALT mode entered via setting bit 7 of the G Port data register. Part will pull up CKI during HALT in crystal clock mode.

Note 5: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than 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 less than 14V. WARNING: Voltages in excess of 14V will cause damage to the pins. This warning excludes ESD transients.

Note 6: Parameter characterized but not tested.

Note 7: t_c = Instruction cycle time.

Comparators AC and DC Characteristics

V_{CC} = 5V, $-40^{\circ}C \le T_A \le +85^{\circ}C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \le V_{IN} \le V_{CC} - 1.5V$		±10	±25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	V_{OL} = 0.4V	1.6			mA

Comparators AC and DC Characteristics (Continued)

V_{CC} = 5V, −40°C ≤ T_A ≤ +85°C

Parameter	Conditions	Min	Typ	Max	Units
High Level Output Current	V_{OH} = 4.6V	1.6			mA
DC Supply Current Per Comparator (When Enabled)				250	µA
Response Time	100 mV Overdrive, 100 pF Load			1	µs

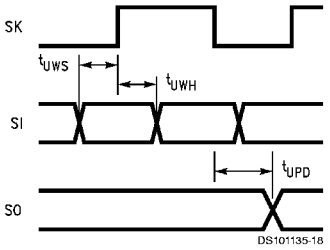


FIGURE 3. MICROWIRE/PLUS Timing

Pin Descriptions

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 an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

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

The device contains three bidirectional 8-bit I/O ports (C, G and L), 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 4* 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.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the USART external clock. L2 and L3 are used for the USART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

- L7 MIWU or T3B
- L6 MIWU or T3A
- L5 MIWU or T2B
- L4 MIWU or T2A
- L3 MIWU or RDX
- L2 MIWU or TDX
- L1 MIWU or CKX
- L0 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WATCHDOG output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

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 clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip 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.

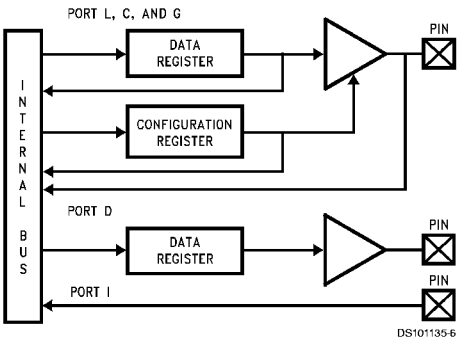


FIGURE 4. I/O Port Configurations

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

Port G has the following alternate features:

- 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)
- G0 INTR (External Interrupt Input)

Port G has the following dedicated functions:

- G7 CKO Oscillator dedicated output or general purpose input
- G1 WDOUT WATCHDOG and/or Clock Monitor dedicated output

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 for these unterminated pins will return unpredicatable values.

PORT I is an eight-bit Hi-Z input port. The 28-pin device does not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

Pin Descriptions (Continued)

The Port I has the following alternate features.

- I6 COMP2OUT (Comparator 2 Output)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I4 COMP2−IN (Comparator 2 Negative Input)
- I3 COMP1OUT (Comparator 1 Output)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I1 COMP1−IN (Comparator 1 Negative Input)

Port D is an 8-bit output port that is preset high when $\overline{\text{RESET}}$ 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 0.8 V_{CC} to prevent the chip from entering special modes. Also keep the external loading on D2 to less than 1000 pF.

Functional Description

The architecture of the device is modified Harvard architecture. With the Harvard architecture, the control store 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 Harvard architecture, permits transfer of data from ROM to RAM.

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.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

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

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

PROGRAM MEMORY

The program memory consists of 12288 bytes 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 devices vector to program memory location 0FF Hex.

The device can be configured to inhibit external reads of the program memory. This is done by programming the Security Byte.

SECURITY FEATURE

The program memory array has an associate Security Byte that is located outside of the program address range. This byte can be addressed only from programming mode by a programmer tool.

Security is an optional feature and can only be asserted after the memory array has been programmed and verified. A secured part will read all 00(hex) by a programmer. The part will fail Blank Check and will fail Verify operations. A Read operation will fill the programmer's memory with 00(hex). The Security Byte itself is always readable with value of 00(hex) if unsecure and FF(hex) if secure.

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, SP pointers and S register.

The data memory consists of 512 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and S are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage.

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

Note: RAM contents are undefined upon power-up.

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.

Data Memory Segment RAM Extension (Continued)

Figure 5 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.

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 intitialized 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.

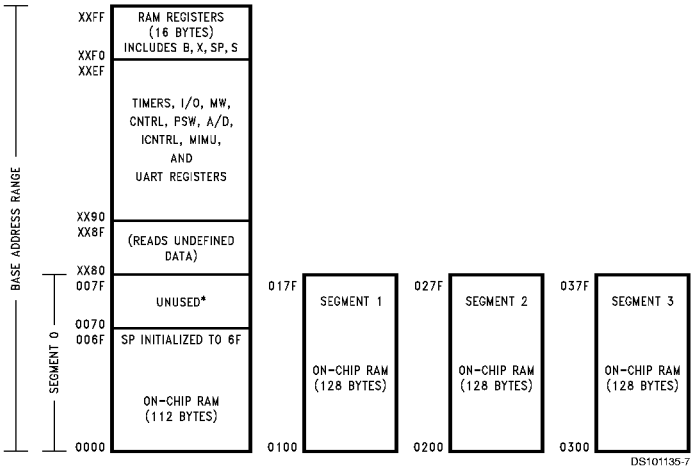
Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 128 bytes of RAM are memory mapped at address locations 0100 to 017F hex.

Reset

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WATCHDOG and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, T2CNTRL and T3CNTRL control registers are cleared. The USART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUI are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN, WKEDG and WKPND are cleared. (Wakeup register WKPND is unknown.) The stack pointer, SP, is initialized to 6F Hex.

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 enter the TRI-STATE mode.

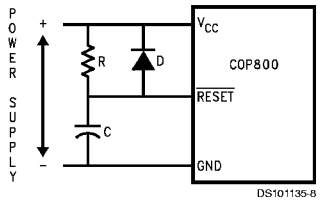
The external RC network shown in *Figure 6* should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



*Reads as all ones.

FIGURE 5. RAM Organization

Reset (Continued)



RC > 5 x Power Supply Rise Time

FIGURE 6. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock (t_c).

Figure 7 shows the Crystal and R/C oscillator diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table 1 shows the component values required for various standard crystal values.

R/C OSCILLATOR

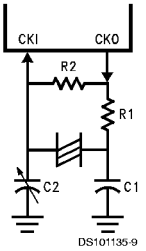
By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table 2 shows the variation in the oscillator frequencies as functions of the component (R and C) values.

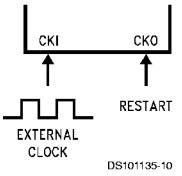
EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

Crystal Oscillator



External Oscillator



R/C Oscillator

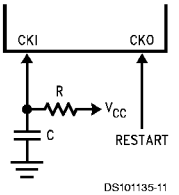


FIGURE 7. Crystal R/C, and External Oscillator Diagrams

TABLE 1. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 5V
0	1	200	100–150	0.455	V_{CC} = 5V

TABLE 2. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (µs)	Conditions
3.3	82	2.2 to 2.7	3.7 to 4.6	V_{CC} = 5V
5.6	100	1.1 to 1.3	7.4 to 9.0	V_{CC} = 5V
6.8	100	0.9 to 1.1	8.8 to 10.8	V_{CC} = 5V

Note: 3k ≤ R ≤ 200k
50 pF ≤ C ≤ 200 pF

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

Timers

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

TIMER T0 (IDLE TIMER)

The device supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The 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

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency (t_c = 1 µs). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1, TIMER T2 AND TIMER T3

The device has a set of three powerful timer/counter blocks, T1, T2 and T3. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the three timer blocks, T1, T2 and T3 are identical, all comments are equally applicable to any of the three timer blocks.

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 powerful and flexible timer block allows the device to easily perform all timer functions with minimal software overhead. 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.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the device to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

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.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 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.

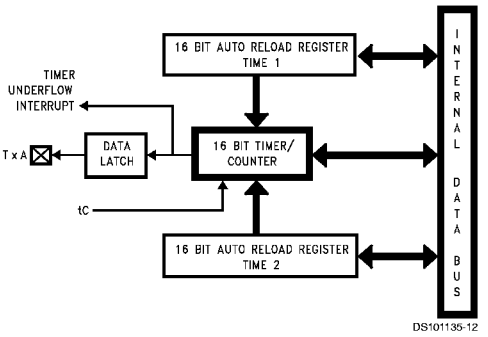


FIGURE 8. Timer in PWM Mode

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.

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 9 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.

Timers (Continued)

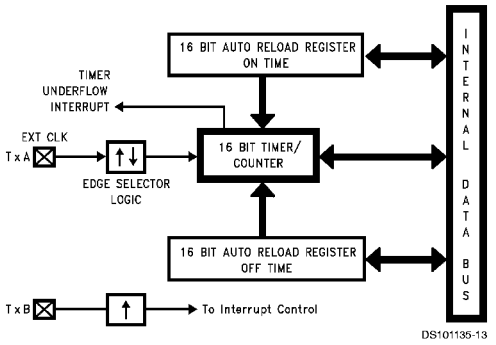


FIGURE 9. Timer in External Event Counter Mode

Mode 3. Input Capture Mode

The 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 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 10 shows a block diagram of the timer in Input Capture mode.

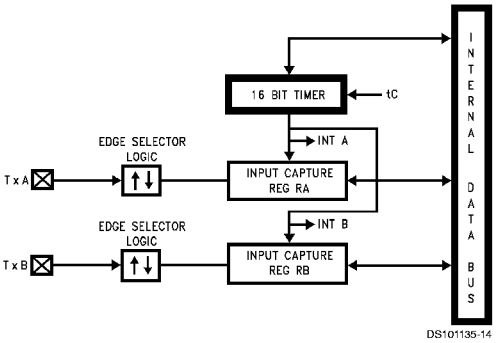


FIGURE 10. Timer in Input Capture Mode

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

Timers (Continued)

Mode	TxC3	TxC2	TxC1	Description	Interrupt A Source	Interrupt B Source	Timer Counts On
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

Power Save Modes

The 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.

HALT MODE

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

The device supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. 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 be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the $\overline{\text{RESET}}$ pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the 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 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

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

The device has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the device will enter and exit the HALT mode as described above. With the HALT disable mask 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").

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.

The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

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 thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, t_c = 1 µs) of the IDLE Timer toggles.

This toggle condition of the thirteenth 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 thirteenth 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.

Power Save Modes (Continued)

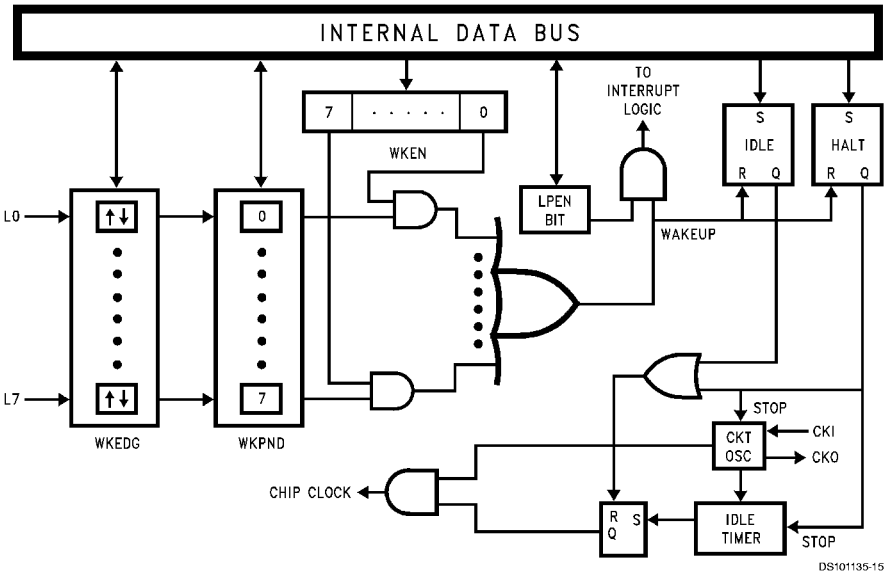
Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the device will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

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.

Multi-Input Wakeup

The Multi-Input Wakeup feature is ued 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 11 shows the Multi-Input Wakeup logic.



0 = low going high ↑
1 = high going low ↓

FIGURE 11. Multi-Input Wake Up 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 Reg: WKEN. The Reg: 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 Reg: 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 pseudo 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 MIWN
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 inherited pseudo 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

Multi-Input Wakeup (Continued)

both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

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.)

Note: There is always the possibility of an interrupt occurring during an instruction which is attempting to reset the GIE bit or any other interrupt enable bit. If this occurs when a single cycle instruction is being used to reset the interrupt enable bit, the interrupt enable bit will be reset but an interrupt may still occur. This is because interrupt processing is started at the same time as the interrupt bit is being reset. To avoid this scenario, the user should always use a two, three, or four cycle instruction to reset interrupt enable bits.

USART

The device contains a full-duplex software programmable USART. The USART (*Figure 12*) consists of a transmit shift register, a receiver 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 framming, 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.

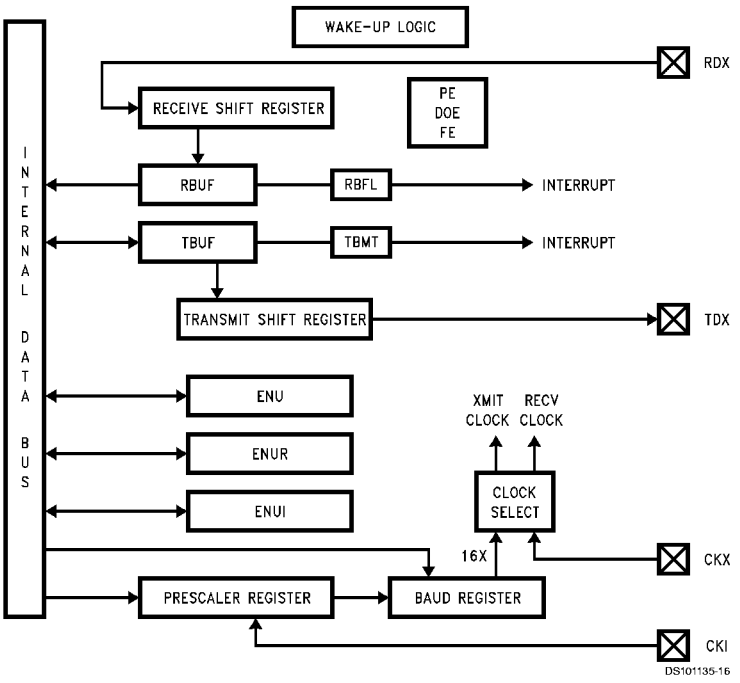


FIGURE 12. USART Block Diagram

USART (Continued)

USART CONTROL AND STATUS REGISTERS

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

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 8)	RBIT9	ATTN	XMTG	RCVG
Bit 7							Bit 0

Note 8: 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.

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.

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.

USART Operation

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

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.

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.

FRAMING FORMATS

The USART supports several serial framing formats (*Figure 13*). 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.

USART Operation (Continued)

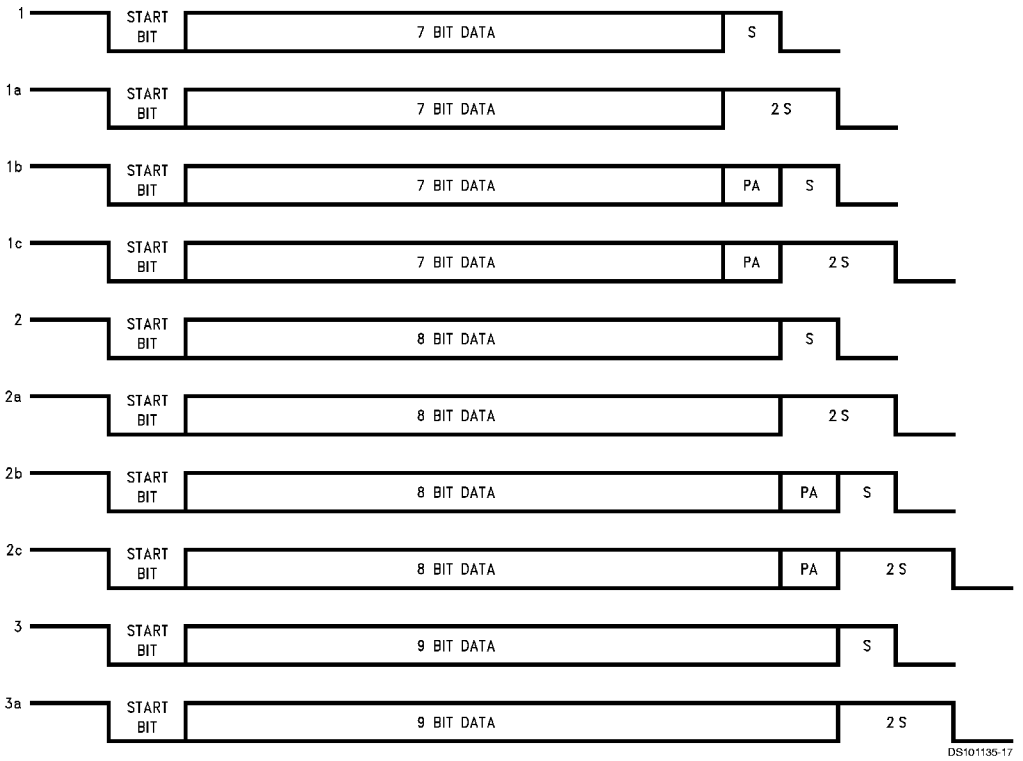


FIGURE 13. Framing Formats

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).

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 14*) The divide factors are specified through two read/write registers shown in *Figure 15*. 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 3*, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK 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 3*. 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.

Baud Clock Generation (Continued)

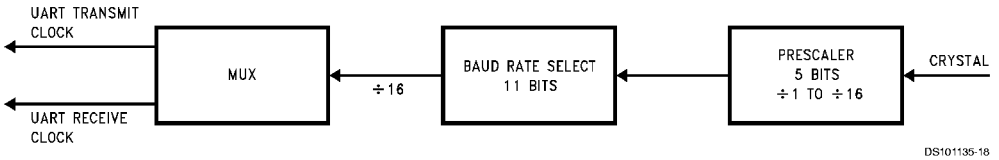


FIGURE 14. USART BAUD Clock Generation

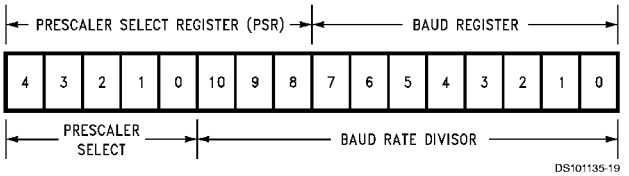


FIGURE 15. USART BAUD Clock Divisor Registers

TABLE 3. 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

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

The entries in *Table 4* assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

As an example, considering the 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 3*. 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 (N – 1 is the value from *Table 4*)

N = 6 (N is the Baud Rate Divisor)

Baud Rate = 1.8432 MHz/(16 x 6) = 19200

Baud Clock Generation (Continued)

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 3*)

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 4*) to obtain a value closest to an integer. This factor happens to be 6.5 (P = 6.5).

$$N = 32.552/6.5 = 5.008 \ (N = 5)$$

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

Using the above values calculated for N and P:

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

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

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.

Diagnostic

Bits CHL0 and CHL1 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.

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.

Comparators

The device contains two differential comparators, each with a pair of inputs (positive and negative) and an output. Ports I1–I3 and I4–I6 are used for the comparators. The following is the Port I assignment:

- I6 Comparator2 output
- I5 Comparator2 positive input
- I4 Comparator2 negative input
- I3 Comparator1 output
- I2 Comparator1 positive input
- I1 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

Comparators (Continued)

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)

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

The CMPSL register contains the following bits:

Rsvd These bit 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.

Comparator outputs have the same spec as Ports L and G except that the rise and fall times are symmetrical.

Multiply/Divide

This device contains a multiply/divide block. This block supports a 1 byte x 2 bytes (3 bytes result) multiply or a 3 bytes/2 bytes (2 bytes result) divide operation. The multiply or divide operation is executed by setting control bits located in the multiply/divide control register. The multiply or divide operands must be placed into the appropriate memory mapped locations before the operation is initiated.

CONTROL REGISTER BITS

Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	DIV OVF	DIV	MULT
Bit 7							Bit 0

The Multiply/Divide control register (MDCR) is located at address xx9D. It has the following bit assignments:

Rsvd These bits are reserved and must be zero

DIVOVF Division Overflow (if the result of a division is greater than 16 bits or the user attempted to divide by zero; 1 = error)

DIV Start Division Operation (1 = start)

MULT Start Multiplication Operation (1 = start)

After the appropriate MDR registers are loaded, the MULT and DIV start bits are set by the user to start a multiply or divide operation. The division operation has priority, if both bits are set simultaneously. The MULT and DIV bits are BOTH automatically cleared by hardware at the end of a divide or multiply operation. Each division operation causes the DIVOVF flag to be set/reset as appropriate. The DIVOVF flag is cleared following a multiplication operation. DIVOVF is a read-only bit. The MULT and DIV bits are read/writable. Bits 3–7 in MDCR should not be used, as the MULT and DIV operations will change their values.

MULTIPLY/DIVIDE OPERATION

For the multiply operation, the multiplicand is placed at addresses xx9B and xx9C. The multiplier is placed at address xx99. For the divide operation, the dividend is placed at addresses xx98 to xx9A and the divisor is placed at addresses xx9B to xx9C. In both operations, all operands are interpreted as unsigned values. The divide or multiply operation is started by setting the appropriate MDCR bit. If both the MULT and DIV bits are set, the microcontroller performs a divide operation. (The user is not required to read or clear the DIVOVF error bit prior to beginning a new multiply/divide operation. This bit is ignored during subsequent operations. However, the next divide operation will overwrite the error flag as appropriate, and the next multiply operation will clear it.)

The multiply operation requires 1 instruction cycle to complete. The divide operation requires 2 instruction cycles to complete. A divide by zero or a division which produces an overflow requires only 1 instruction cycle to execute. The MDR1 through MDR5 registers and the MDCR register can not be read from or written to during a multiply or divide operation. Any attempt to write in to these registers will be ignored. Any attempt to read these registers will return undefined data.

The result of a multiply is placed in addresses xx99–xx9B. The result of a divide is placed in a ddresses xx98–xx99. If a division by zero is attempted or if the resulting quotient of a divide operation is more than 16 bits long, then the DIVOVF bit is set in the multiply/divide control register. The dividend and the divisor are left unchanged. The divide operation always causes the DIVOVF flag to be set or reset as appropriate. The DIVOVF flag is cleared following a multiply operation.

RESET STATE

A reset signal applied to the device during normal operation has the following affects:

MDCR is cleared, and any operation in progress is stopped.

MDR1 through MDR5 are undefined.

Multiply/Divide (Continued)

TABLE 5. Multiply/Divide Registers

Register Name (Address)	Multiplication Assignment		Division Assignment	
	Before Operation	After Operation	Before Operation	After Operation
MDR1 (xx98)	Unused	Unchanged	Low Byte of Dividend	Low Byte of Result
MDR2 (xx99)	Multiplier	Low Byte of Result	Middle Byte of Dividend	High Byte of Result
MDR3 (xx9A)		Middle Byte of Result	High Byte of Dividend	Undefined
MDR4 (xx9B)	Low Byte of Multiplicand	High Byte of Result	Low Byte of Divisor	Low Byte of Divisor
MDR5 (xx9C)	High Byte of Multiplicand	Unchanged	High Byte of Divisor	High Byte of Divisor

Interrupts

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 16 shows the Interrupt Block diagram.

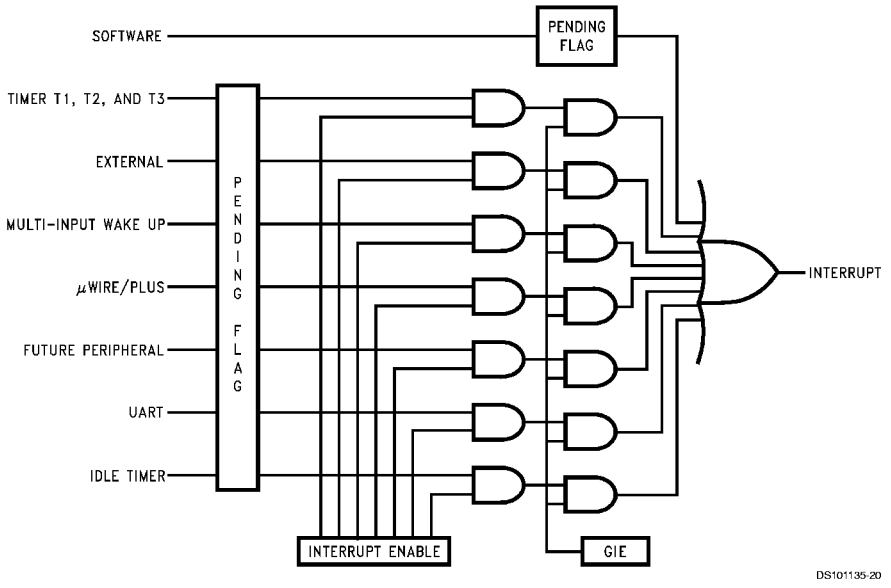


FIGURE 16. Interrupt Block Diagram

Interrupts (Continued)

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.

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-

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 Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
(2)	Reserved		0yFC–0yFD
(3)	External	Pin G0 Edge	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 Goes Low	0yF2–0yF3
(8)	Reserved		0yF0–0yF1
(9)	UART	Receive	0yEE–0yEF
(10)	UART	Transmit	0yEC–0yED
(11)	Timer T2	T2A/Underflow	0yEA–0yEB
(12)	Timer T2	T2B	0yE8–0yE9
(13)	Timer T3	T3A/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 9: 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.

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.

Interrupts (Continued)

Figure 17 illustrates the different steps performed by the VIS instruction. *Figure 18* 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.

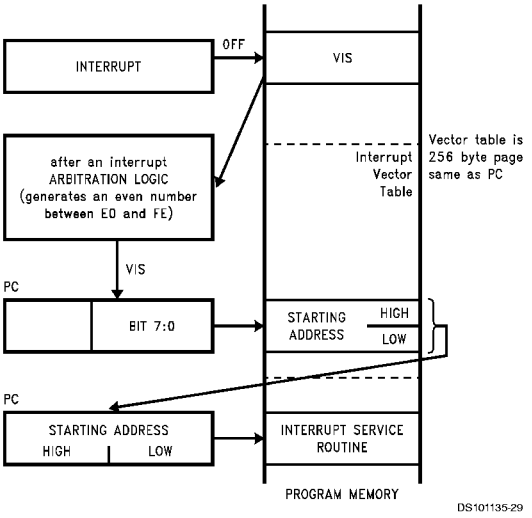


FIGURE 17. VIS Operation

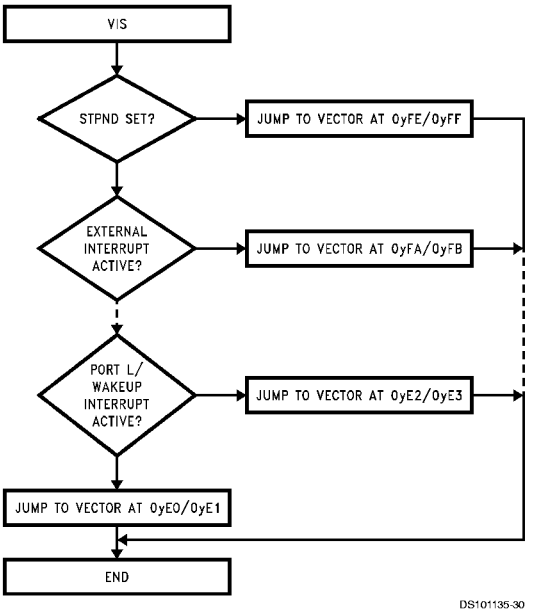


FIGURE 18. VIS Flowchart

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
```

Interrupts (Continued)

NON-MASKABLE INTERRUPT

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.

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.

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.)

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.

WATCHDOG

The device contains a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. 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.

The lower limit of the service window is fixed at 2048 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 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

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

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.

WATCHDOG Operation

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 user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. 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. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT 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 enter high impedance state.

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 enter the high impedance TRI-STATE mode 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.

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 he 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.

WATCHDOG Operation (Continued)

- 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 mask 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 mask option selected, or with the single-pin R/C oscillator mask 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 2048 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 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 thirteenth 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 2048 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 2048 instruction cycles without causing a WATCHDOG error.

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 zeros. The opcode for software interrupt is zero. 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), 140 to 17F (Segment 1), and all other segments (i.e., Segments 2 ... etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

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

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

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the device to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^2PROMs etc.) and with other microcontrollers which support the MICROWIRE 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 19* shows a block diagram of the MICROWIRE/PLUS logic.

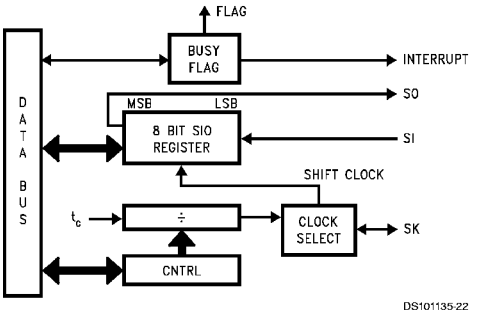


FIGURE 19. MICROWIRE/PLUS Block Diagram

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 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

TABLE 10. MICROWIRE/PLUS Master Mode Clock Select

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

Where t_c is the instruction cycle clock

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 20* shows how two devices, microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS 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 low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the device. 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. Table VII summarizes the bit settings required for Master mode of operation.

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 VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

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 is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

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 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

MICROWIRE/PLUS (Continued)

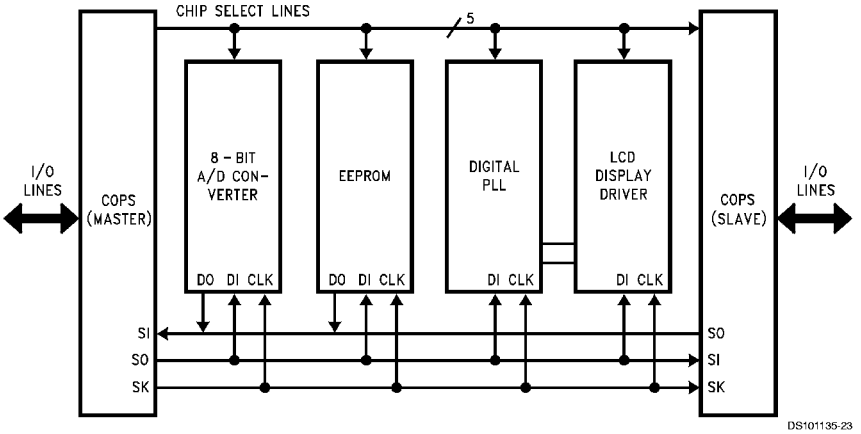


FIGURE 20. MICROWIRE/PLUS Application

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 xx97	Unused RAM Address Space (Reads Undefined Data)
xx98	Dividend or Result Byte (MDR1)
xx99	Dividend/Multiplier or Result Byte (MDR2)
xx9A	Dividend/Result Byte or Undefined (MDR3)
xx9B	Dividend/Multiplicand or Result Byte (MDR4)
xx9C	Divisor or Multiplicand Byte (MDR5)
xx9D	Multiply/Divide Control Register (MDCR)
xx9E to xxAF	Reserved
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 (CMPSL)
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (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	Reserved
xxCC	Reserved
xxCD 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)
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 xxFB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100 to 017F	On-Chip 128 RAM Bytes
0200 to 027F	On-Chip 128 RAM Bytes
0300 to 037F	On-Chip 128 RAM Bytes

Note: Reading memory locations 0070H–007FH (Segment 0) will return all ones. Reading unused memory locations 0080H–00AFH (Segment 0) will return undefined data. Reading memory locations from other unused Segments (i.e., Segment 4, Segment 5, ... etc.) will return undefined data.

Addressing Modes

There are ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the Load B Immediate instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location up to 32k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

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.

Instruction Set

Register and Symbol Definition

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
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

Instruction Set (Continued)

INSTRUCTION SET

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 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 in 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)
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	DECrementA	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
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)

Instruction Set (Continued)

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

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
IFNE	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
DCOR	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,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	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				

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

Opcode Table

Bits 7–4																Bits 3–0
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]	*	LD B,0F	IFBNE 0	JSR 0000–00FF	JMP 0000–00FF	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 0100–01FF	JMP 0100–01FF	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 0200–02FF	JMP 0200–02FF	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 0300–03FF	JMP 0300–03FF	JP+20	JP+4	3
JP–11	JP–27	LD 0F4,#i	DRSZ 0F4	*	LAID	ADD A,#i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B,0B	IFBNE 4	JSR 0400–04FF	JMP 0400–04FF	JP+21	JP+5	4
JP–10	JP–26	LD 0F5,#i	DRSZ 0F5	*	JID	AND A,#i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B,0A	IFBNE 5	JSR 0500–05FF	JMP 0500–05FF	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,9	IFBNE 6	JSR 0600–06FF	JMP 0600–06FF	JP+23	JP+7	6
JP–8	JP–24	LD 0F7,#i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	IFBIT 7,[B]	*	LD B,8	IFBNE 7	JSR 0700–07FF	JMP 0700–07FF	JP+24	JP+8	7
JP–7	JP–23	LD 0F8,#i	DRSZ 0F8	NOP	*	LD A,#i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B,7	IFBNE 8	JSR 0800–08FF	JMP 0800–08FF	JP+25	JP+9	8
JP–6	JP–22	LD 0F9,#i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B,6	IFBNE 9	JSR 0900–09FF	JMP 0900–09FF	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,5	IFBNE 0A	JSR 0A00–0AFF	JMP 0A00–0AFF	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,4	IFBNE 0B	JSR 0B00–0BFF	JMP 0B00–0BFF	JP+28	JP+12	B
JP–3	JP–19	LD 0FC,#i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B,3	IFBNE 0C	JSR 0C00–0CFF	JMP 0C00–0CFF	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,2	IFBNE 0D	JSR 0D00–0DFF	JMP 0D00–0DFF	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,1	IFBNE 0E	JSR 0E00–0EFF	JMP 0E00–0EFF	JP+31	JP+15	E
JP–0	JP–16	LD 0FF,#i	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B,0	IFBNE 0F	JSR 0F00–0FFF	JMP 0F00–0FFF	JP+32	JP+16	F

where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode (see following table)

Development Tools Support

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 solutions 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.

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

Development Tools Support

(Continued)

COP8 Real-Time Emulation Tools

- **COP8-DM:** MetaLink Debug Module. A moderately priced real-time in-circuit emulation tool, with COP8 device programmer. Includes COP8-NSDEV, DriveWay COP8 Demo, MetaLink Debugger, power supply, emulation cables and adapters.
- **IM-COP8:** MetaLink iceMASTER®. A full featured, real-time in-circuit emulator for COP8 devices. Includes 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.

TOOLS ORDERING NUMBERS FOR THE COP87L88FH FAMILY DEVICES

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	Not available for this device		
	COP8-DM	Contact MetaLink		
	Development Devices	COP87L84FH COP87L88FH	VL	16k OTP devices. No windowed devices
	IM-COP8	Contact MetaLink		
MetaLink	COP8-EPU	Not available for this device		
	COP8-DM	DM4-COP8-888FH (10 MHz), plus PS-10, plus DM-COP8/xxx (ie. 40D)	M	Included p/s (PS-10), target cable of choice (DIP or PLCC; i.e. DM-COP8/40D), 16/20/28/40 DIP/SO and 44 PLCC programming sockets.
	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-888FH44PW-AD-10	M	10 MHz 40 DIP probe card; 2.5V to 6.0V
		PC-888FH40DW-AD-10	M	10 MHz 44 PLCC probe card; 2.5V to 6.0V
ICU	COP8-EVAL	Not available for this device		
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		Contact vendors	L - H	For approved programmer listings and vendor information, go to our OTP support page at: www.national.com/cop8
Cost: Free; VL =< $100; L = $100 - $300; M = $300 - $1k; H = $1k - $3k; VH = $3k - $5k				

Development Tools Support (Continued)

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

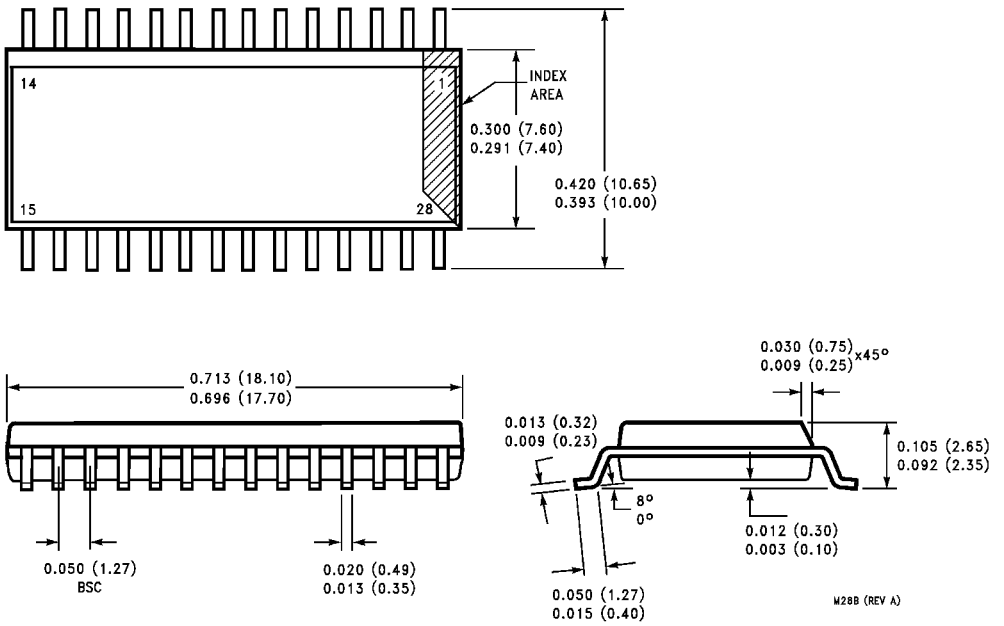
The following companies have approved COP8 programmers in a variety of configurations. Contact your local office or distributor. You can link to their web sites and get the latest listing of approved programmers from National's COP8 OTP Support page at: www.national.com/cop8.

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.

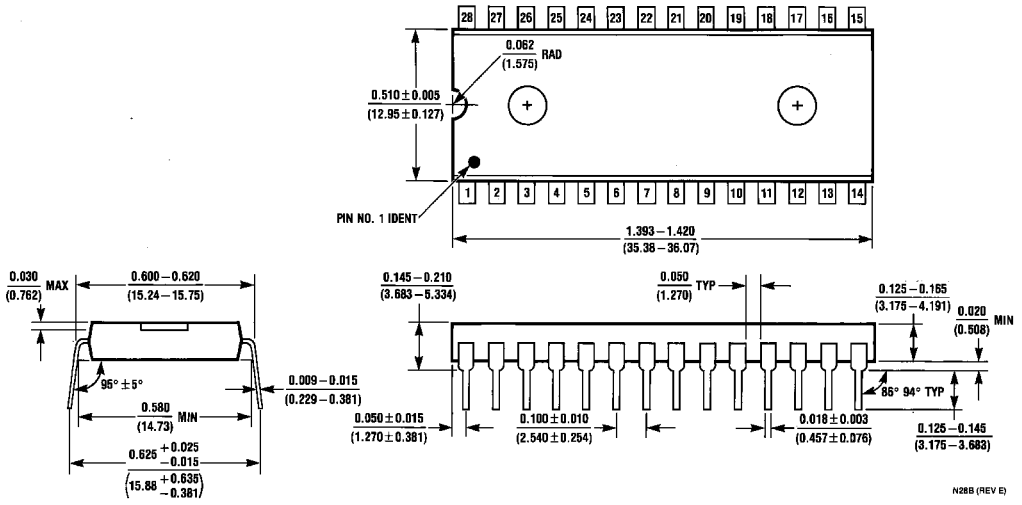
Customer Support

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

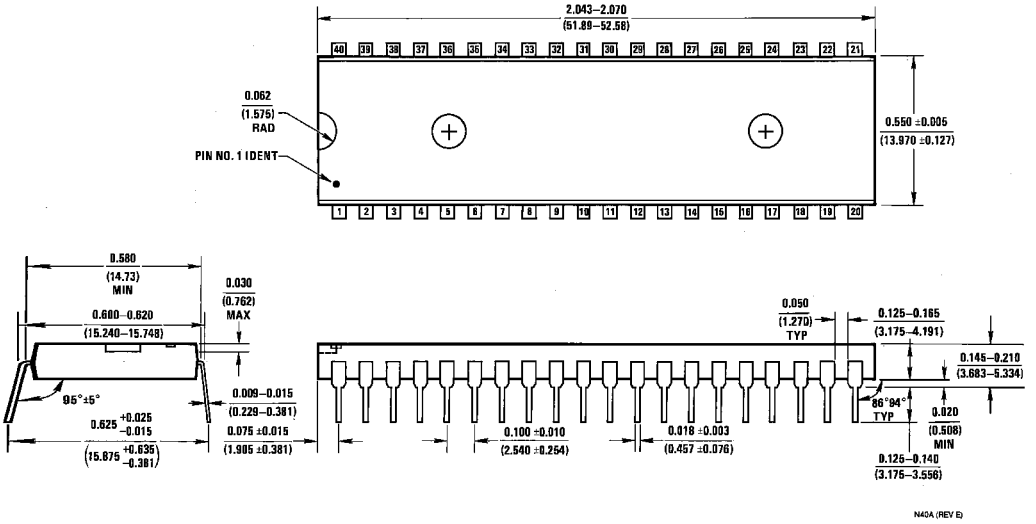


Molded SO Wide Body Package (M)
Order Number COP684FH-XXX/M,
COP884FH-XXX/M or COP984FH-XXX/M
NS Package Number M28B

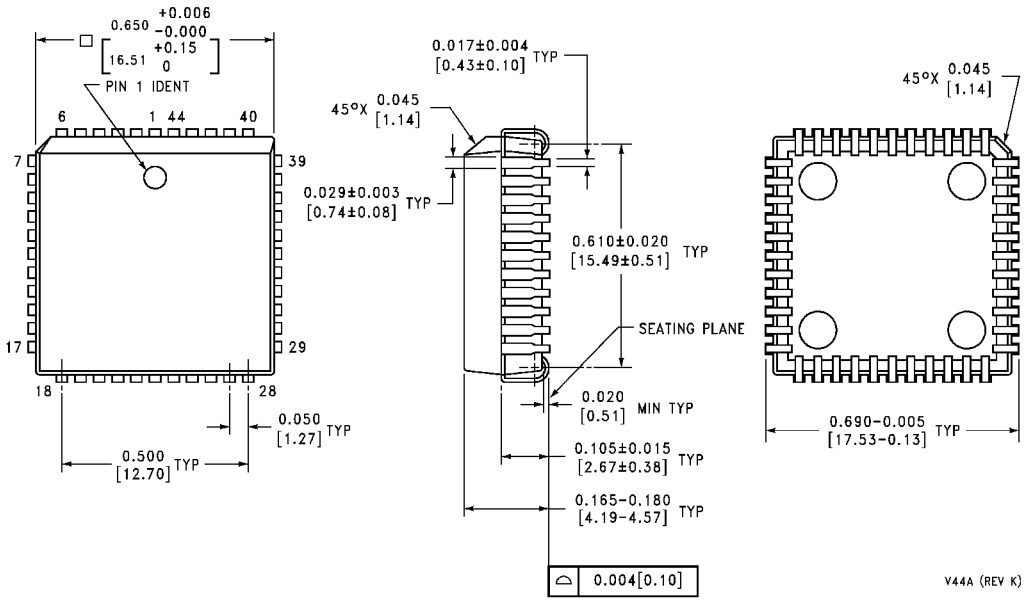


Molded Dual-In-Line Package (N)
Order Number COP684FH-XXX/N,
COP884FH-XXX/N or COP984FH-XXX/N
NS Package Number N28B

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



Molded Dual-In-Line Package (N)
Order Number COP688FH-XXX/N,
COP888FH-XXX/N or COP988FH-XXX/N
NS Package Number N40A



Plastic Leaded Chip Carrier (V)
Order Number COP688FH-XXX/V,
COP888FH-XXX/V or COP988FH-XXX/V
NS Package Number V44A