National Semiconductor

COP87L88RW
8-Bit One-Time Programmable (OTP) Microcontroller with Pulse Train Generators and Capture Modules

General Description

The COP87L88RW OTP microcontrollers are large memory (32k), highly integrated COP8™ Feature core devices, with advanced features including including Pulse Train Generators, Capture Modules, and hardware multiply/divide. These multi-chip CMOS devices are suited for applications requiring a full featured controller with high I/O pincount, pulse generation and capture, and a full-duplex USART, and for pre-production devices for ROM designs. Pin and software compatible 16k ROM versions are available (COP888GW), along with a range of COP8 software and hardware development tools.

Family features include an 8-bit memory mapped architecture, 10MHz CKI with 1µs instruction cycle, hardware multiply/divide functions, two 100ns capture modules, four pulse train generators with 16 bit prescalers, two multi-function 16-bit timer/counters, idle timer, full-duplex USART, MICROWIRE/PLUS™ serial I/O, two power saving HALT/IDLE modes, MIWU, high current outputs, software selectable I/O options, 2.7v-5.5v operation, program code security, and 68 pin packages.

Devices included in this datasheet are:

Device	Memory (bytes)	RAM (bytes)	I/O Pins	Packages	Temperature
COP87L88RW	32k OTP	512	64	68 PLCC	-40 to +85˚C

Key Features

- Multiply/divide functions
- Full duplex UART
- Four pulse train generators with 16-bit prescalers
- Two 16-bit input capture modules with 8-bit prescalers
- Two 16-bit timers, each with two 16-bit registers supporting
  - — Processor independent PWM mode
  - — External event counter mode
  - — Input capture mode
- 32 kbytes on-board OTP EPROM with security feature

  Note: Mask ROMed devices with equivalent on-chip features and program memory sizes of 16k is available.
- 512 bytes on-board RAM

Additional Peripheral Features

- Idle Timer
- Multi-Input Wake-Up (MIWU) with optional interrupts (8)
- WATCHDOG™ and clock monitor logic
- MICROWIRE/PLUS serial I/O

I/O Features

- Memory mapped I/O
- Software selectable I/O options
  - — TRI-STATE® output
  - — Push-pull output
  - — Weak pull-up input
  - — High impedance input
- Schmitt trigger inputs on ports G and L
- Package: 68 PLCC with I/O pins

CPU/Instruction Set Features

- 1 µs instruction cycle time
- Fourteen multi-source vectored interrupts servicing
  - — External interrupt
  - — Idle timer T0
  - — Two timers (each with 2 interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake-Up
  - — Software trap
  - — UART (2)
  - — Default VIS
  - — Capture timers
  - — Counters (one vector for all four counters)
- 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 to 5.5V
- Temperature range: −40˚C to +85˚C

Development Support

- Emulation device for the COP888GW
- Real time emulation and full program debug offered by MetaLink's Development System

COP87L88RW 8-Bit One-Time Programmable (OTP) Microcontroller with Pulse Train Generators and Capture Modules

Block Diagram

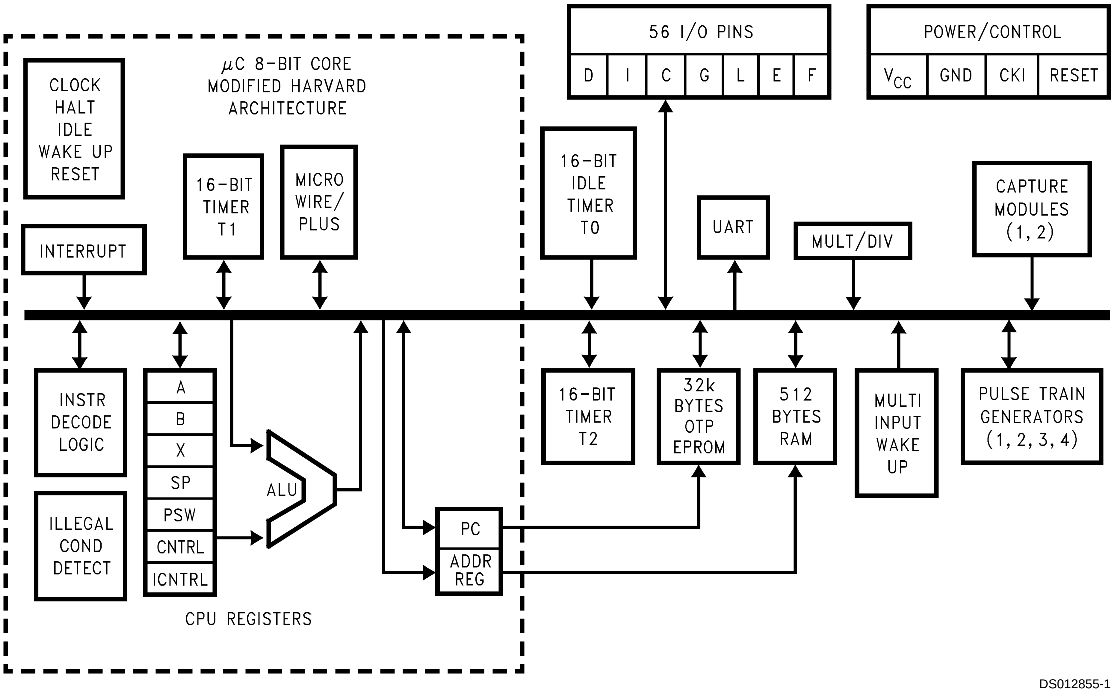
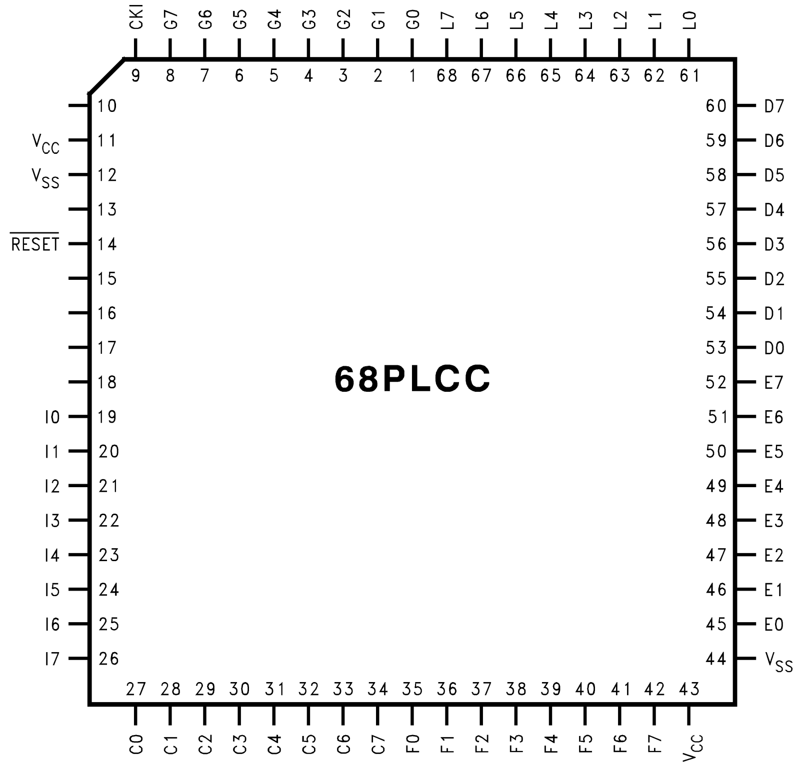


FIGURE 1. COP87L88RW Block Diagram

Connection Diagram



Note: -X Crystal Oscillator
Note: -E Halt Enable

Top View
Order Number COP87L88RWV-XE
See NS Plastic Chip Package Number V68A

FIGURE 2. Connection Diagram

Absolute Maximum Ratings (Note 1)

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 +150˚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			14	mA
HALT Current (Note 4)	V_{CC} = 5.5V, CKI = 0 MHz			12	µA
IDLE Current					
CKI = 10 MHz	V_{CC}= 5.5V			1.7	mA
Input Levels (V_{IH}, V_{IL})					
$\overline{\text{RESET}}$, CKI					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
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	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V, V_{IN} = 0V	40		−250	µA
G Port Input Hysteresis	(Note 7)		0.05 V_{CC}	0.35 V_{CC}	V
Output Current Levels					
D Outputs					
Source	V_{CC}= 4.5V, V_{OH}= 3.3V	−0.4			mA
Sink (Note 5)	V_{CC}= 4.5V, V_{OL}= 1V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC}= 4.5V, V_{OH}= 2.7V	−10		−100	µA
Source (Push-Pull Mode)	V_{CC}= 4.5V, V_{OH}= 3.3V	−0.4			mA
Sink (Push-Pull Mode)	V_{CC}= 4.5V, V_{OL}= 0.4V	1.6			mA
TRI-STATE Leakage	V_{CC} = 5.5V	−2		+2	µA
Allowable Sink/Source					
Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Notes 6, 8)	Room Temp			±200	mA
RAM Retention Voltage, V_r (Note 8)	500 ns Rise and Fall Time (min)	2			V
Input Capacitance	(Note 8)			7	pF
Load Capacitance on D2	(Note 8)			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		1.0		DC	µs
Ceramic					
Inputs					
t_{SETUP}	$V_{CC} \ge 4.5V$	200			ns
t_{HOLD}	$V_{CC} \ge 4.5V$	60			ns
Output Propagation Delay (Note 10)	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \ge 4.5V$			0.7	µs
All Others	$V_{CC} \ge 4.5V$			1	µs
MICROWIRE™ Setup Time (t_{UWS}) (Note 8)	$V_{CC} \ge 4.5V$	20			
MICROWIRE Hold Time (t_{UWH}) (Note 8)	$V_{CC} \ge 4.5V$	56			ns
MICROWIRE Output Propagation Delay (t_{UPD})	$V_{CC} \ge 4.5V$			220	
Input Pulse Width (Note 9)					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			
Timer 1, 2 Input High Time		1			
Timer 1, 2 Input Low Time		1			
Capture Timer High Time		1			CKI
Capture Timer Low Time		1			CKI
Reset Pause Width		1			µs

Note 2: Maximum rate of voltage change to be defined.

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

Note 4: The HALT mode will stop CKI from oscillatng. Test conditions: All inputs tied to V_{CC}, L, C, E, F, and G port I/O's configured as outputs and programmed low and not driving a load; D outputs programmed low and not driving a load. 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: The user must guarantee that D2 pin does not source more than 10 mA during RESET. If D2 sources more than 10 mA during reset, the device will go into programming mode.

Note 6: 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 7: Condition and parameter valid only for part in HALT mode.

Note 8: Parameter characterized but not tested.

Note 9: t_c = Instruction Cycle Time

Note 10: The output propagation delay is referenced to the end of the instruction cycle where the output change occurs.

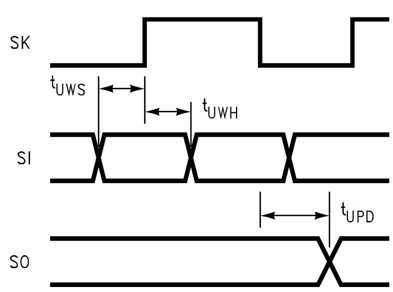


FIGURE 3. MICROWIRE/PLUS Timing

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 comes from 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 five bidirectional 8-bit I/O ports (C, E, F, 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 UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the capture timer input functions CAP1 and CAP2.

The Port L has the following alternate features:

L0 MIWU
L1 MIWU or CKX
L2 MIWU or TDX
L3 MIWU or RDX
L4 MIWU or T2A
L5 MIWU or T2B
L6 MIWU or CAP1
L7 MIWU or CAP2

Port G is an 8-bit port with 6 I/O pins (G0–G5), an input pin (G6), and a dedicated output pin (G7). Pins G0–G6 all have Schmitt Triggers on their inputs. Pin G7 serves as the dedicated output pin for the CKO clock output. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 6 I/O bits (G0–G5) can be individually configured under software control.

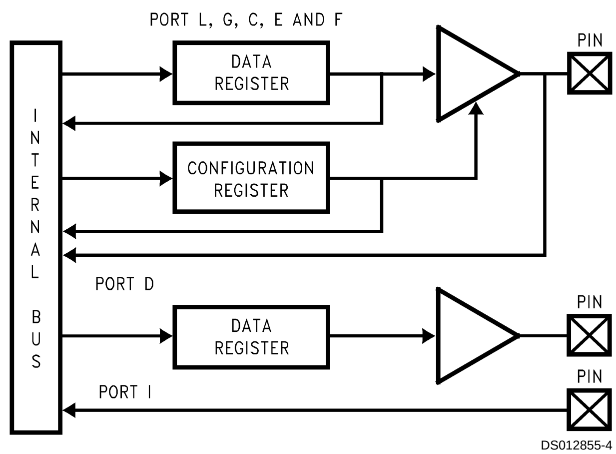


FIGURE 4. I/O Port Configurations

Since G6 is an input only pin and G7 is dedicated CKO clock output pin, 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.

	Config Reg.	Data Reg.
G7	Not Used	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0 INTR (External Interrupt Input)
G2 T1B (Timer T1 Capture Input)
G3 T1A (Timer T1 I/O)
G4 SO (MICROWIRE Serial Data Output)
G5 SK (MICROWIRE Serial Clock)
G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G7 CKO Oscillator dedicated output

Pin Descriptions (Continued)

Ports C and F are 8-bit I/O ports.

Port E is an 8-bit I/O port. It has the following alternate features:

E0 CT1 (Output for counter1, Pulse Train Generator)

E1 CT2 (Output for counter2, Pulse Train Generator)

E2 CT3 (Output for counter3, Pulse Train Generator)

E3 CT4 (Output for counter4, Pulse Train Generator)

Port I is an eight-bit Hi-Z input port.

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 $<$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 32 kbytes of OTP EPROM. 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.

Note: Mask ROMed devices with equivalent on-chip features and program memory sizes of 16k is available.

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.

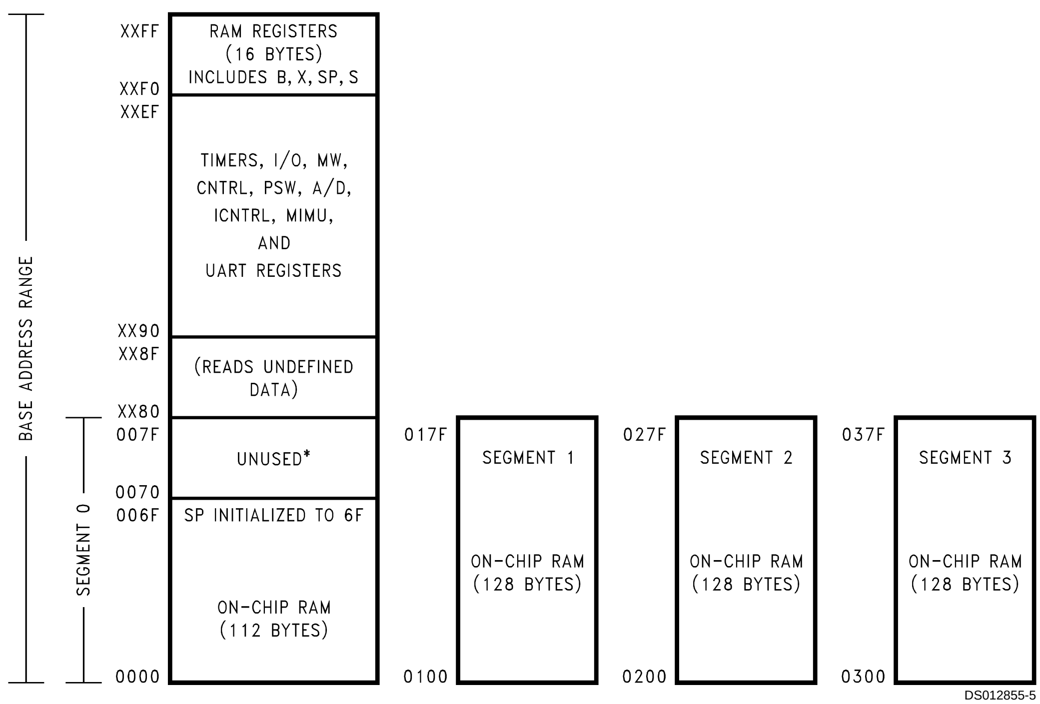
Data Memory Segment RAM Extension (Continued)

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 initialized to point at data memory location 006F as a result of reset.

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

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



*Reads as all ones.

FIGURE 5. RAM Organization

Reset

This device enters a reset state immediately upon detecting a logic low on the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin must be held low for a minimum of one instruction cycle to guarantee a valid reset. During power-up initialization, the user must insure that the $\overline{\text{RESET}}$ pin is held low until this device is within the specified V_{CC} voltage. An R/C circuit on the $\overline{\text{RESET}}$ pin with a delay 5 times (5x) greater than the power supply rise time is recommended.

When the $\overline{\text{RESET}}$ input goes low, the I/O ports are initialized immediately, with any observed delay being only propagation delay. When the $\overline{\text{RESET}}$ pin goes high, this device comes out of the reset state synchronously. This device will be running within two instruction cycles of the $\overline{\text{RESET}}$ pin going high.

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

Some registers are reset to a known state, whereas other registers and RAM are “unchanged” by reset. When the controller goes into reset state while it is performing a write operation to one of these registers or RAM that are “unchanged” by reset, the register or RAM value will become unknown (i.e. not unchanged). This is because the write operation is terminated prematurely by reset and the results become uncertain. These registers and RAM locations are unchanged by reset only if they are not written to when the controller resets.

Reset (Continued)

The following initializations occur with $\overline{\text{RESET}}$:

Port L: TRI-STATE

Port C: TRI-STATE

Port G: TRI-STATE

Port E: TRI-STATE

Port F: TRI-STATE

Port D: HIGH

PC: CLEARED

PSW, CNTRL and ICNTRL registers: CLEARED

SIOR:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

T1CNTRL: CLEARED

T2CNTRL: CLEARED

TxRA, TxRB: RANDOM

CCMR1, CCMR2: CLEARED

CM1PSC, CM1CRL, CM1CRH, CM2PSC, CM2CRL, and CM2CRH:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

CCR1 and CCR2: CLEARED

CxPRH, CxPRL, CxCTH, and CxCTL:

RANDOM after RESET at power-on

PSR, ENUR and ENUI: CLEARED

ENU: CLEARED except Bit 1 (TBMT) = 1

Accumulator, Timer 1 and Timer 2:

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

UNAFFECTED after RESET with RC clock option (power already applied)

RANDOM after RESET at power-on

MDCR: CLEARED

MDR1, MDR2, MDR3, MDR4, MDR5: RANDOM

WKEN, WKEDG: CLEARED

WKPND: RANDOM

S Register: CLEARED

SP (Stack Pointer): Loaded with 6F Hex

B and X Pointers:

UNAFFECTED after RESET with power already applied

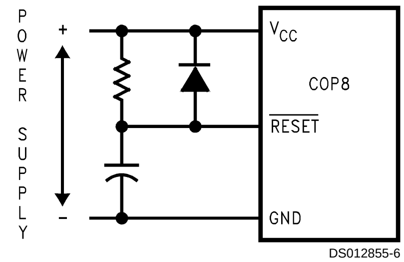
RANDOM after RESET at power-on

RAM:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-on

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.



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 diagram

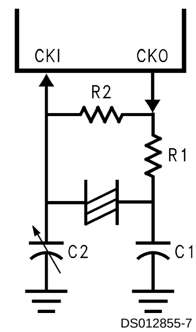


FIGURE 7. Crystal Diagram

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.

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

Control Registers

CNTRL Register (Address X'00EE)

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

SL1 & SL0	Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL	Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively

Control Registers (Continued)

T1C0 Timer T1 Start/Stop control in timer modes 1 and 2 T1 Underflow Interrupt Pending Flag in timer mode 3

T1C1 Timer T1 mode control bit

T1C2 Timer T1 mode control bit

T1C3 Timer T1 mode control bit

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

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE Global interrupt enable (enables interrupts)

EXEN Enable external interrupt

BUSY MICROWIRE/PLUS busy shifting flag

EXPND External interrupt pending

T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge

T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)

C Carry Flag

HC Half Carry Flag

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

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

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge

µWEN Enable MICROWIRE/PLUS interrupt

µWPND MICROWIRE/PLUS interrupt pending

T0EN Timer T0 Interrupt Enable (Bit 12 toggle)

T0PND Timer T0 Interrupt pending

LPEN L Port Interrupt Enable (Multi-Input Wake up/ Interrupt)

Bit 7 could be used as a flag

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

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB Timer T2 Interrupt Enable for T2B Input capture edge

T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge

T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge

T2PNDA Timer T2 Interrupt Pending Flag (Auto reload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)

T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1 Timer T2 mode control bit

T2C2 Timer T2 mode control bit

T2C3 Timer T2 mode control bit

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

Timers

The device contains a very versatile set of timers (T0, T1, T2). 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)
- 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 AND TIMER T2

The device has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2 are identical, all comments are equally applicable to either of the two 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 tc. Upon every underflow the timer is alternately reloaded with

Timers (Continued)

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.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

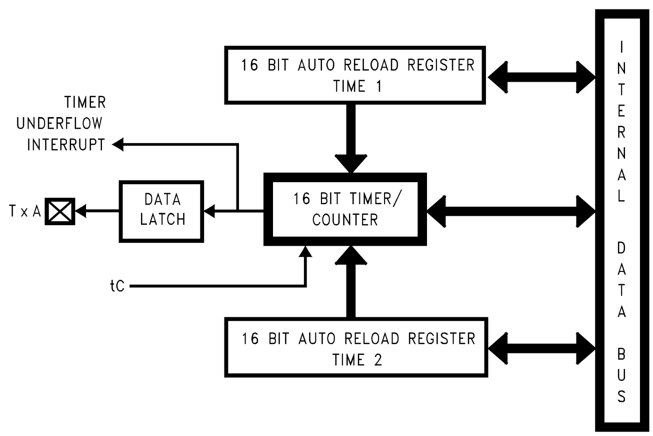


FIGURE 8. Timer in PWM Mode

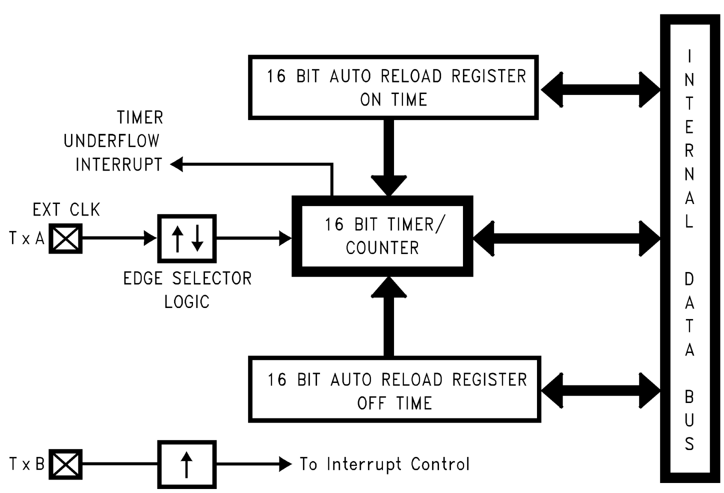


FIGURE 9. Timer in External Event Counter Mode

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)

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.

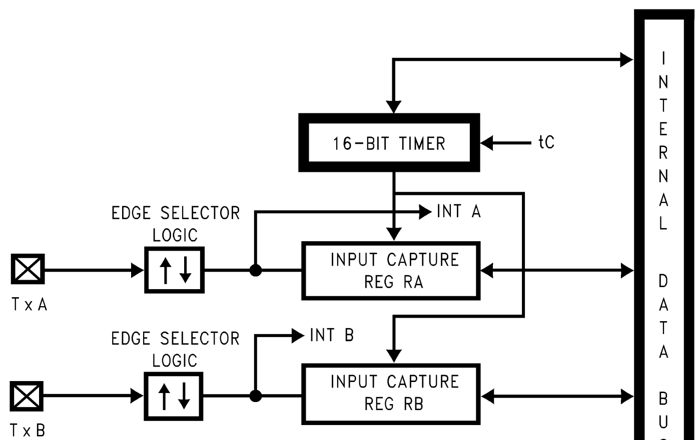
The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

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.



DS012855-10

FIGURE 10. Timer in Input Capture Mode

TIMER CONTROL FLAGS

The timers T1 and T2 have identical control structures. The control bits and their functions are summarized below.

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
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag
	1 = Timer Interrupt Enabled
	0 = Timer Interrupt Disabled
TxC3	Timer Mode Control
TxC2	Timer Mode Control
TxC1	Timer Mode Control

CAPTURE TIMER

This device contains two independent capture timers, Capture Timer 1 and Capture Timer 2. Each capture timer contains an 8-bit programmable prescaler register, a 16-bit down counter, a 16-bit input capture register, and capture edge select logic. The 16-bit down counter is clocked at a specific frequency determined by the value loaded into the prescaler register. A selected positive or negative edge transition on the capture input causes the contents of the down counter to be latched into the capture register. The values captured in the registers reflect the elapsed time between two positive or two negative transitions on the capture input. The time between a positive and negative edge (a pulse width) may be measured if the selected capture edge is switched after the first edge is captured. Each capture timer may be stopped/started under software control, and each capture timer may be configured to interrupt the microcontroller on an underflow or input capture.

Figure 11 shows the capture timer 1 block diagram.

Timers (Continued)

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

TABLE 2. Timer Mode Control

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Positive TxB Edge	TxA Positive Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Positive TxB Edge	TxA Negative Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Positive Edge TxB Positive Edge	Positive TxA Edge or Timer Underflow	Positive TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Positive Edge TxB Negative Edge	Positive TxA Edge or Timer Underflow	Negative TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Negative Edge TxB Positive Edge	Negative TxA Edge or Timer Underflow	Positive TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Negative Edge TxB Negative Edge	Negative TxA Edge or Timer Underflow	Negative TxB Edge	t_c

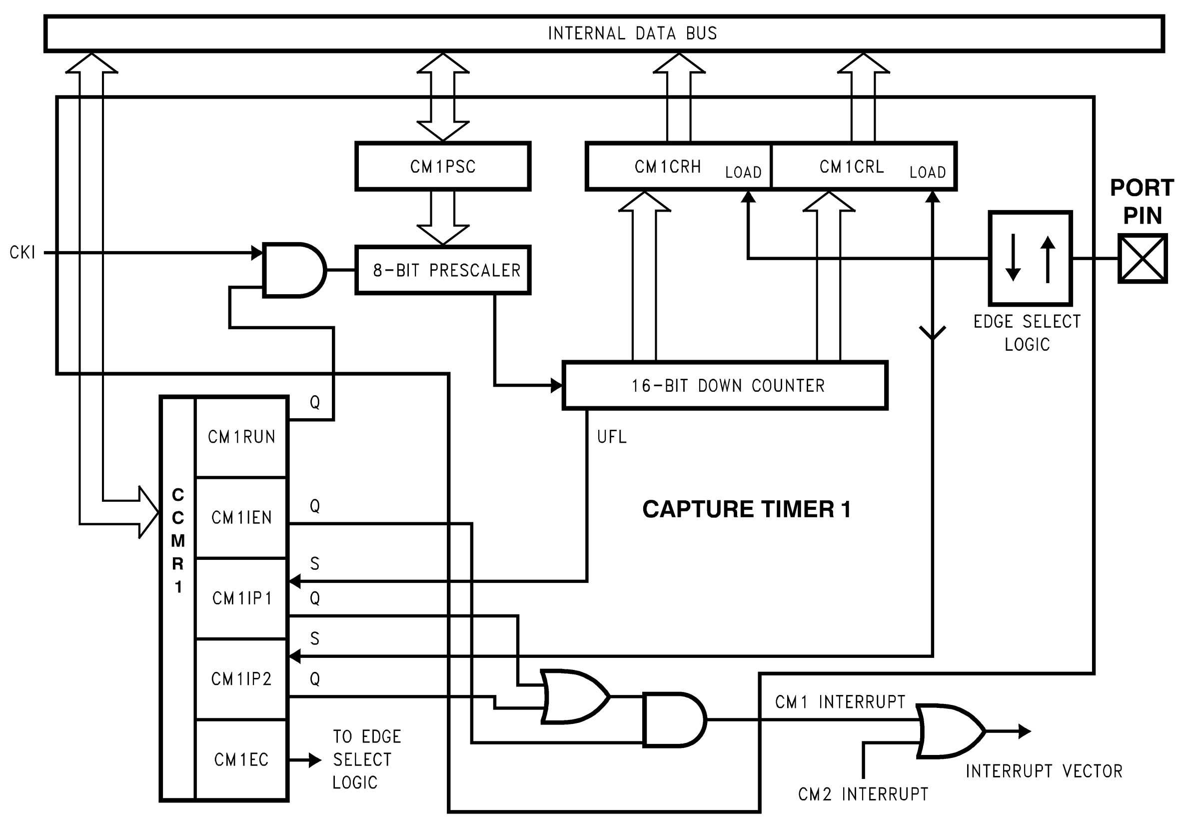


FIGURE 11. Capture Timer 1 Block Diagram

Timers (Continued)

The registers shown in the block diagram include those for Capture Timer 1 (CM1), as well as, the capture timer 1 control register. These registers are read/writable (with the exception of the capture registers, which are read-only) and may be accessed through the data memory address/data bus. The registers are designated as:

CM1PSC Capture Timer 1 Prescaler (8-bit)

CM1CRL Capture Timer 1 Capture Register (Low-byte), read-only

CM1CRH Capture Timer 1 Capture Register (High-byte), read-only

CM2PSC Capture Timer 2 Prescaler (8-bit)

CM2CRL Capture Timer 2 Capture Register (Low-byte), read-only

CM2CRH Capture Timer 2 Capture Register (High-byte), read-only

CCMR1 Control Register for Capture Timer 1

CCMR2 Control Register for Capture Timer 2

CONTROL REGISTER BITS

The control bits for Capture Timer 1 (CM1) and Capture Timer 2 (CM2) are contained in CCMR1 and CCMR2.

The CCMR1 Register Bits are:

CM1RUN CM1 start/stop control bit (1 = start; 0 = stop)

CM1IEN CM1 interrupt enable control bit (1 = enable IRQ)

CM1IP1 CM1 interrupt pending bit 1 (1 = CM1 underflowed)

CM1IP2 CM1 interrupt pending bit 2 (1 = CM1 captured)

CM1EC Select the active edge for capture on CM1 (0 = rising, 1 = falling)

CM1TM CM1 test mode control bit (1 = special test path in test mode. This bit is reserved during normal operation, and must never be set to one.)

CM1 TM	un-used	un-used	CM1 EC	CM1 IP2	CM1 IP1	CM1 IEN	CM1 RUN
Bit 7							Bit 0

All interrupt pending bits must be reset by software.

The CCMR2 Register Bits are:

CM2RUN CM2 start/stop control bit (1 start; 0=stop)

CM2IEN CM2 interrupt enable control bit (1= enable IRQ)

CM2IP1 CM2 interrupt pending bit 1 (1=CM2 underflowed)

CM2IP2 CM2 interrupt pending bit 2 (1=CM2 captured)

CM2EC Select the active edge for capture on CM2 (0 = rising, 1 = falling)

CM2TM CM2 test mode control bit (1 = special test path in test mode. This bit is reserved during normal operation, and must never be set to one.)

CM2 TM	un-used	un-used	CM2 EC	CM2 IP2	CM2 IP1	CM2 IEN	CM2 RUN
Bit 7							Bit 0

All interrupt pending bits must be reset by software.

FUNCTIONAL DESCRIPTION

The capture timer is used to determine the time between events, where an event is simply a selected edge transition on the capture input. The resolution of the time measurement is dependent on the frequency at which the down counter is clocked. The value loaded into the prescaler controls this frequency.

The prescaler is clocked by CKI, while the down counter is clocked on every underflow of the prescaler. This means the prescaler simply divides the CKI clock before it is fed into the down counter. The prescaler register must be loaded with a value corresponding to the CKI divisor needed to produce the desired down counter clock. The appropriate prescaler value can be determined using the following equation:

Down Counter Clock Frequency = CKI/(CMxPSC + 1)

The capture input signal is set up by configuring the port pin associated with the capture timer as an input. The edge select bit for the capture input is then set or reset according to the desired transition. If the pin is configured as an input, the appropriate external transition will cause a capture. If the pin is configured as an output, toggling the data register bit will cause a capture. If interrupts are used, the capture timer interrupt pending bits are cleared and the capture timer interrupt enable bit is set. Both interrupt sources, down counter underflow and input capture edge, are enabled/disabled with the same CMxIEN bit. The GIE bit must also be set to enable interrupts. The interrupt signals from the two capture timers are gated to a single 16-bit interrupt vector located at addresses 0xE6 and 0xE7.

The capture timer is started by writing a “1” to the capture timer start/stop bit. Setting this bit also enables the port pin to be the capture input to the capture timer. The internal prescaler is loaded with the contents of the prescaler register, and begins counting down. Setting the start/stop bit also loads the down counter with 0FFFF Hex. The prescaler is clocked by CKI. An underflow of the prescaler decrements the 16-bit down counter, and reloads the value from the prescaler register into the prescaler. Each additional underflow of the prescaler decrements the down counter, and reloads the prescaler from the prescaler register.

If a selected edge transition on the input capture pin occurs, the contents of the down counter are immediately latched into the capture register, the down counter is re-initialized to 0FFFF Hex, and the capture input pending flag is set. The prescaler counter is not loaded. (In order for an input transition to be guaranteed recognized, the signal on the capture input pin must have a low pulse width and a high pulse width of at least one CKI period.) If interrupts are enabled, the capture timer generates an interrupt. The prescaler and down counter continue to operate until a reset condition occurs or the capture timer start/stop bit is reset. The user must process capture interrupts faster than the capture input frequency, otherwise input captures may be lost or erroneous values may be read.

If the down counter underflows (changes state from 0000 to FFFF) before a capture input is detected, the underflow interrupt pending flag is set. If interrupts are enabled, the capture timer generates an interrupt.

The capture timer may be stopped at any time under software control by resetting the capture timer start/stop bit. A capture may occur before the start/stop bit is physically cleared, due to the fully asynchronous nature of the input capture signal. The user must ensure that the software handles this situation correctly. If the user wishes to process this capture and interrupts are being used, the capture timer interrupts should not be disabled prior to stopping the timer. If interrupts are not being used, the user should poll the capture timer pending bits after stopping the timer. If the user wishes to ignore this capture and interrupts are being used, the capture timer interrupt service routine should check that

Timers (Continued)

the timer is still running prior to processing capture interrupts. If the user is polling the pending flags, these flags should be cleared after the timer is stopped. The contents of the prescaler and down counter remain unchanged while the capture timer is stopped. The capture edge detect logic is disabled, and no capture takes place even if an external capture signal occurs. The capture timer may be restarted under software control by writing a "1" to the start/stop bit. This causes the prescaler and down counter to be re-initialized. The prescaler is loaded from the prescaler register, and the down counter is loaded with 0FFFF Hex.

RESET STATE

A reset signal applied to the counter block during normal operation has the following effects:

- Clear CCMR1 register
- Clear CCMR2 register
- CM1PSC, CMICRL, CM1CRH, CM2PSC, CM2CRL and CM2CRH are unaffected. (At power-on, the contents of these registers are undefined.)

The bi-directional port pins are initialized during reset as HI-Z inputs. Setting the start/stop bits connects the pins to the capture timers.

INITIALIZATION

The user should perform the following initialization prior to starting the capture timer:

1. Reset the CMxRUN bit
2. Configure the corresponding Port bits as inputs
3. Set the edge control bits CMxEC
4. Reset CMxIP1 (CMxIP1 = 0)
5. Reset CMxIP2 (CMxIP2 = 0)
6. Load the 8-bit prescaler register CMxPSC with the desired value (from 0 to 255)
7. Set CMxIEN (if interrupts are to be used)
8. Set the Global Interrupt Enable (GIE) bit (if interrupts are to be used)
9. Set CMxRUN bit to start the capture timer

WARNING

In order to avoid erroneous interrupts, the capture timer interrupts must be disabled prior to setting/resetting the capture edge control bits (CMxEC). In addition, after selecting the interrupt edge, the pending flags must be reset before the capture interrupts are enabled or re-enabled. If the initialization sequence outlined above is followed each time the user alters the edge control bits, the user is guaranteed to avoid erroneous interrupts.

Pulse Train Generators

This device contains four independent pulse train generators. Each individual generator is controlled by a corresponding 16-bit counter. Each counter has a 16-bit prescaler and a 16-bit count register. Each counter may be configured to output a selected number of 50% duty cycle pulses. The contents of the prescaler determine the width of the output pulses, and the value of the count register determines the number of pulses. Each counter may be stopped/started under software control, and each counter may be configured to interrupt the microcontroller on an underflow.

Figure 12 shows the pulse train generator 1 block diagram.

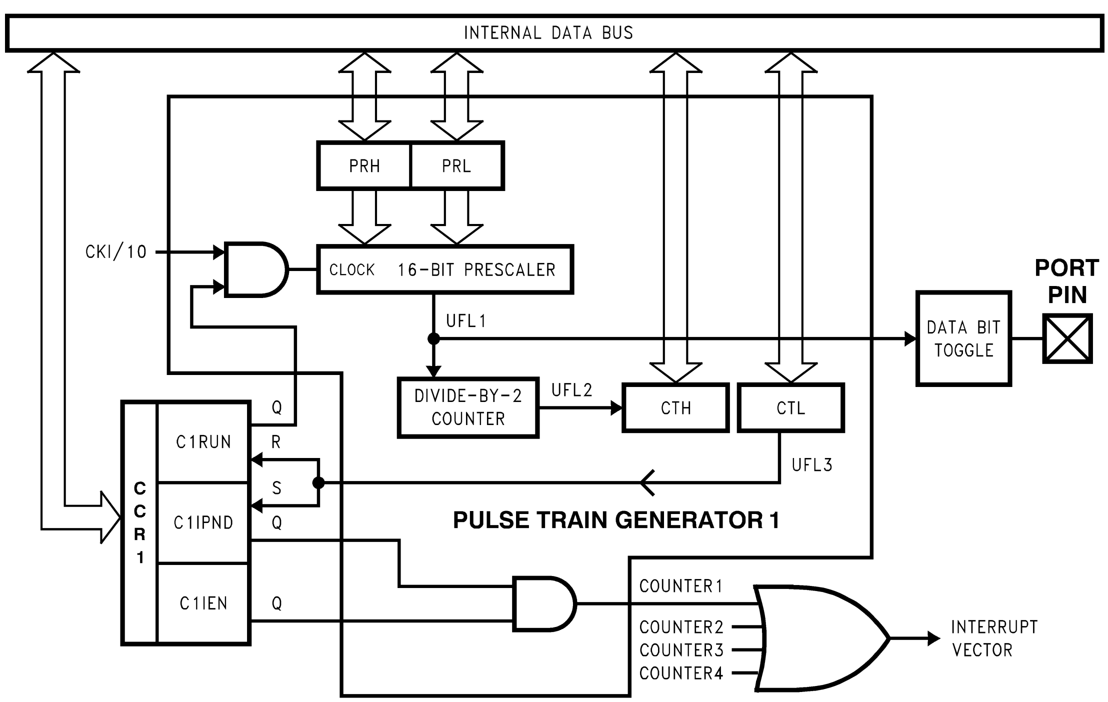


FIGURE 12. Pulse Train Generator 1 Block Diagram

Pulse Train Generators (Continued)

The four 8-bit registers shown in each individual counter in the block diagram constitute a 16-bit prescaler and a 16-bit count register. These registers are all read/writable and may be accessed through the data memory address/data bus. The registers are designated as:

CxPRL Low-byte of the Prescaler

CxPRH High-byte of the Prescaler

CxCTL Low-byte of the Count Register

CxCTH High-byte of the Count Register

CONTROL REGISTER BITS

The control bits for Counter 1 and Counter 2 are contained in the CCR1 register. The CCR1 Register bits are:

C1RUN COUNTER1 start/stop control bit (1 = start; 0 = stop)

C1IEN COUNTER1 interrupt enable control bit (1 = enable IRQ)

C1IPND COUNTER1 interrupt pending bit (1 = counter 1 underflowed)

C1TM COUNTER1 test mode control bit (1=special test path in test mode. This bit is reserved during normal operation, and must never be set to one.)

C2RUN COUNTER2 start/stop control bit (1 = start; 0 = stop)

C2IEN COUNTER2 interrupt enable control bit (1= enable IRQ)

C2IPND COUNTER2 interrupt pending bit (1 = counter 2 underflowed)

C2TM COUNTER2 test mode control bit (1=special test path. This bit is reserved during normal operation, and must never be set to one.)

All interrupt pending bits must be reset by software.

C2TM	C2 IPND	C2 IEN	C2 RUN	C1TM	C1 IPND	C1 IEN	C1 RUN
Bit 7							Bit 0

The control bits for Counter 3 and Counter 4 are contained in the CCR2 register. The CCR2 Register bits are:

C3RUN COUNTER3 start stop control bit (1 =start; 0 = stop)

C3IEN COUNTER3 interrupt enable control bit (1 = enable IRQ)

C3IPND COUNTER3 interrupt pending Bit (1=counter 3 underflowed)

C3TM COUNTER3 test mode control bit (1=special test path. This bit is reserved during normal operation, and must never be set to one.)

C4RUN COUNTER4 start/stop control bit (1 = start; 0 = stop)

C4IEN COUNTER4 interrupt enable control bit (1 = enable IRQ)

C4IPND COUNTER4 interrupt pending bit (1 =counter 4 underflowed

C4TM COUNTER4 test mode control bit (1 =special test path. This bit is reserved during normal operation, and must never be set to one.)

C4TM	C4 IPND	C4 IEN	C4 RUN	C3TM	C3 IPND	C3 IEN	C3 RUN
Bit 7							Bit 0

All interrupt pending bits must be reset by software.

FUNCTIONAL DESCRIPTION

The pulse train generator may be used to produce a series of output pulses of a given width. The high/low time of a pulse is determined by the contents of the prescaler. The number of pulses in a series is determined by the contents of the count register.

The prescaler is loaded with a value corresponding to the desired width of the output pulse (t_w). The high time and low time of the output signal are each equal to t_w, therefore the output signal produced has a 50% duty cycle and a period equal to 2 * t_w. The appropriate prescaler value can be determined using the following equation:

$$t_w = [(PRH * 256) + PRL + 1] * t_c$$

Since PRH and PRL are both 8-bit registers, this equation allows a maximum t_w of 65536 t_c and a minimum t_w of one t_c. The internal prescaler is automatically loaded from PRH and PRL when the counter start/stop bit is set.

The count register is loaded with a value corresponding to the desired number of output pulses. The appropriate count value is calculated with the following equation:

$$\text{Number of Pulses} = CTH * 256 + CTL + 1$$

The port pin associated with the counter OUT signal is configured in software as an output, and preset to the desired start logic level. lf interrupts are to be used, the counter interrupt pending bit is cleared and the interrupt enable bit is set. The GIE bit must also be set to enable interrupts. The interrupt signals from the four counters are gated to a single interrupt vector located at addresses 0xF0–0xF1.

The counter is started by writing a “1” to the counter start/stop bit. This resets the divide-by-2 counter which produces the clock signal for the counter register from the prescaler underflow (See *Figure 12*). It also reloads the internal prescaler and starts the prescaler counting down on the next rising edge of t_c. The prescaler is clocked on the rising edge of t_c to ensure synchronization. Each subsequent rising edge of t_c causes the prescaler to be decremented. When the prescaler underflows, UFL1 is generated (see *Figure 13*). This signal causes the port pin to toggle. In addition, the internal prescaler is reloaded with the value from the PRH and PRL registers. Each additional underflow of the prescaler causes the port pin to toggle and reloads the internal prescaler.

Every second underflow of the prescaler generates the signal UFL2. (UFL2 occurs at half the frequency of UFL1, or once per output pulse.) This signal, UFL2, decrements the count register. Therefore, the count registers are decremented once per output pulse.

The underflow of the counter register produces the signal UFL3. This signal stops the counter by resetting the counter start/stop bit, and sets the counter interrupt pending flag. If the counter interrupt is enabled, an interrupt occurs.

The counter may be stopped at any time under software control by resetting the counter start/stop bit. The contents of the count register and the output on the associated port pin are frozen. The counter may be restarted under software control by setting the start/stop bit. The internal prescaler is automatically reloaded from PRH and PRL when the counter start/stop bit is set, therefore a full width pulse will be generated before the output is toggled. The user may also choose to alter the logic level on the port pin before restarting. This is done by initializing the associated port pin data register bit. A counter underflow may occur before the start/stop bit is physically cleared by software. The user must ensure that the software handles this situation correctly. If the user wishes to process this underflow and interrupts are being used, the counter interrupts should not be disabled prior to

Pulse Train Generators (Continued)

stopping the timer. If interrupts are not being used, the user should poll the counter pending bits after stopping the timer. If the user wishes to ignore this underflow and interrupts are being used, the counter interrupt should be disabled prior to stopping the timer. If the user is polling the pending flags, these flags should be cleared after the timer is stopped.

If the default level of the output pin is high (associated port data register bit is set to "1") and the counter is stopped during a low level, the low level becomes the default level. The software must reinitialize the port pin to a high level before restarting if necessary. The programmer may also have to adjust the counter value.

RESET STATE

A reset signal applied to the pulse train generator block during normal operation has the following effects:

- Counting stops immediately
- Interrupt enable bit is reset to zero
- Counter start/stop bit is reset to zero
- Interrupt pending bit is reset to zero
- Test mode controI bit is reset to zero
- PRL, PRH, CTL and CTH are unaffected (At power-on reset, the contents of the prescaler and count register are undefined.)
- Divide-by-2 counter is reset
- The bi-directional port pins are initialized during reset as HI-Z inputs. The appropriate bits must be initialized as outputs, in order to route the Counter OUT signals to the port pins.

INITIALIZATION

The user should perform the following initialization prior to starting the counter:

1. Load PRL register
2. Load PRH register
3. Load CTL register
4. Load CTH register
5. Reset CxIPND bit
6. Set CxIEN (if interrupt is to be used)
7. Configure the associated port bit as an output (if OUT is to be used)
8. Set the Global Interrupt Enable (GIE) bit (if interrupt is to be used)
9. Set CxRUN bit to start counter

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.

TABLE 3. 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

CONTROL REGISTER BITS

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

MULT Start Multiplication Operation (1 = start)

DIV Start Division Operation (1 = start)

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

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

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.

Multiply/Divide (Continued)

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 into 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 addresses 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.

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. 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 lhe machine.

The device supports two 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 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 deIay to ensure that the oscilIator 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.

The devices have 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.

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

This toggle condition of the 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.

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

Figure 13 shows the Multi-Input Wake Up logic.

Multi-Input Wakeup (Continued)

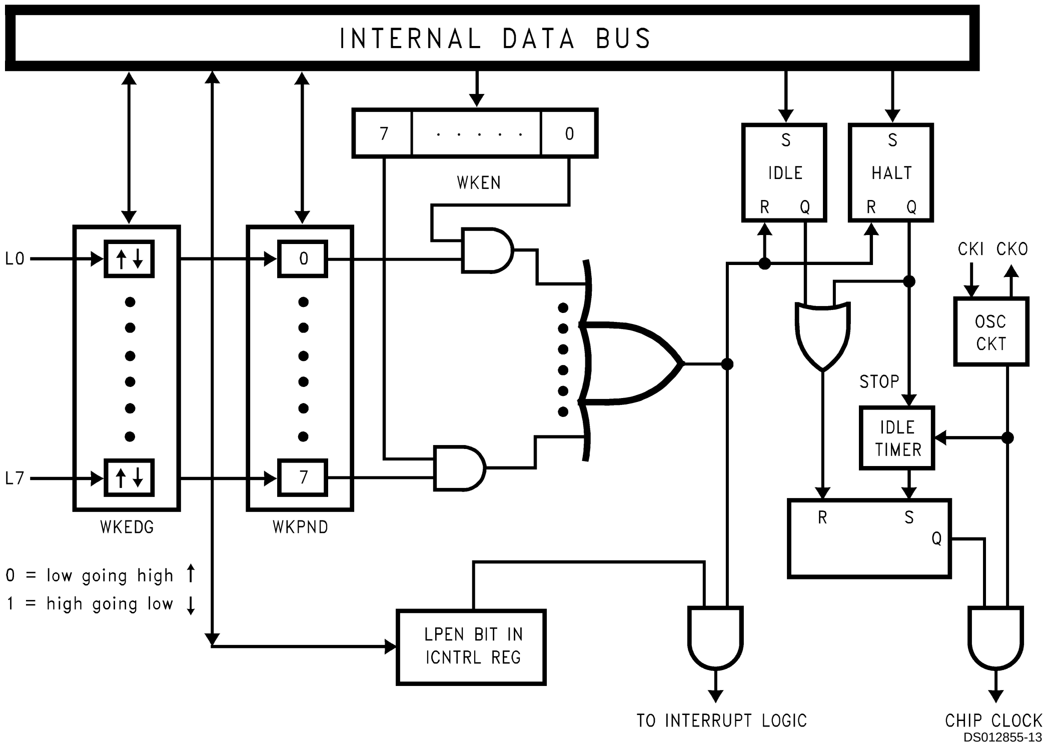


FIGURE 13. Multi-Input Wake Up Logic

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

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the register WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a Wake Up 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 reenabled.

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
SBIT   5,   WKEDG
RBIT   5,   WKPND
SB1T   5,   WKEN
```

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

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

The occurrence of the selected trigger condition for Multi-Input Wake Up 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 wake up conditions, the device will not enter the HALT mode if any Wake Up bit is both enabled and pending. Consequently, the user must clear 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.

Multi-Input Wakeup (Continued)

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 wake up information.)

UART

The device contains a full-duplex software programmable UART. The UART (*Figure 14*) consists of a transmit shift register, a receive shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART'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 UART mode of operation: asynchronous or synchronous.

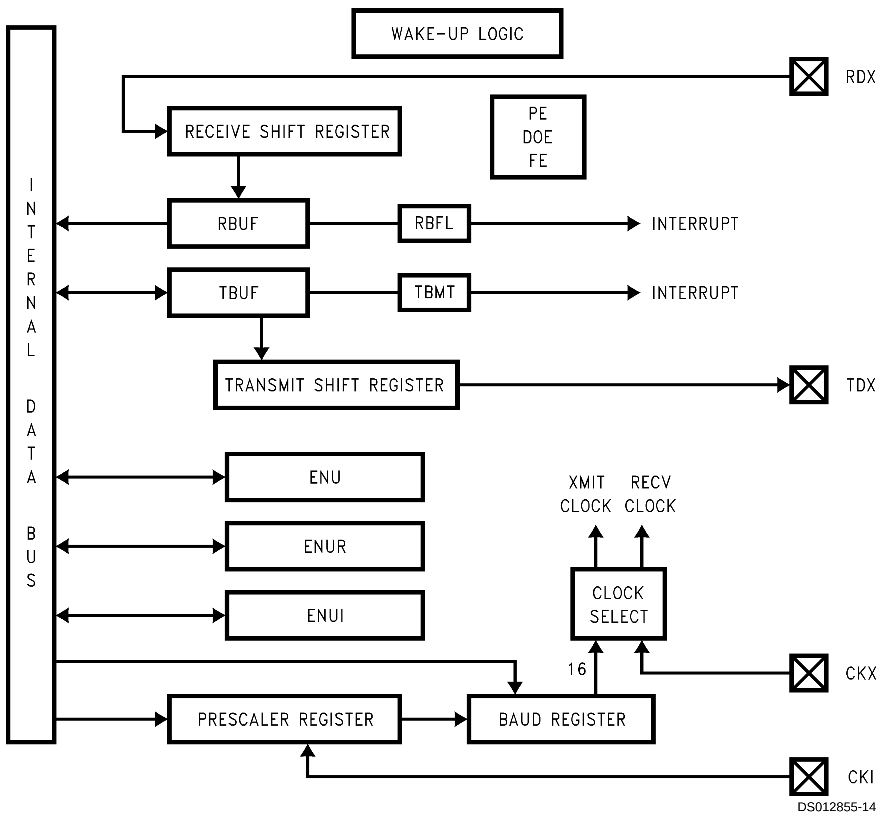


FIGURE 14. UART Block Diagram

UART (Continued)

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

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

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

ENUR-UART Receive Control and Status Register (Address at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
0RD	0RD	0RD	0RW*	0R	0RW	0R	0R
Bit 7							Bit 0

ENUI-UART Interrupt and Clock Source Register (Address at 0BC)

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

* Bit is not used.

0 Bit is cleared on reset.

1 Bit is set to one on reset.

R Bit is read-only; it cannot be written by software.

RW Bit is read/write.

D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

DESCRIPTION OF UART REGISTER BITS

ENU—UART CONTROL AND STATUS REGISTER

TBMT: This bit is set when the UART 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.

RBFL: This bit is set when the UART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF.

ERR: This bit is a global UART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur.

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

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

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

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

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

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

PSEL1, PSEL0: Parity select bits.

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

PSEL1 = 0, PSEL1 = 1 Odd Parity (if Parity enabled)

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

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

PEN: This bit enables/disables Parity (7- and 8-bit modes only).

PEN = 0 Parity disabled.

PEN = 1 Parity enabled.

ENUR—UART RECEIVE CONTROL AND STATUS REGISTER

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high.

XMTG: This bit is set to indicate that the UART is transmitting. It gets reset at the end of the last frame (end of last Stop bit).

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

RBIT9: Contains the ninth data bit received when the UART is operating with nine data bits per frame.

SPARE: Reserved for future use.

PE: Flags a Parity Error.

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.

FE: Flags a Framing Error.

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.

DOE: Flags a Data Overrun Error.

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.

ENUI—UART INTERRUPT AND CLOCK SOURCE REGISTER

ETI: This bit enables/disables interrupt from the transmitter section.

ETI = 0 Interrupt from the transmitter is disabled.

ETI = 1 Interrupt from the transmitter is enabled.

ERI: This bit enables/disables interrupt from the receiver section.

ERI = 0 Interrupt from the receiver is disabled.

ERI = 1 Interrupt from the receiver is enabled.

XTCLK: This bit selects the clock source for the transmitter section.

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.

XRCLK: This bit selects the clock source for the receiver section.

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.

SSEL: UART mode select.

SSEL = 0 Asynchronous Mode.

SSEL = 1 Synchronous Mode.

ETDX: TDX (UART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit.

UART (Continued)

To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length.

STP2: This bit programs the number of Stop bits to be transmitted.

STP2 = 0 One Stop bit transmitted.

STP2 = 1 Two Stop bits transmitted.

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 UART 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.

UART Operation

The UART 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 UART 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 UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART 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 UART 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 UART 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 UART supports several serial framing formats (*Figure 15*). 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 UART “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.

UART Operation (Continued)

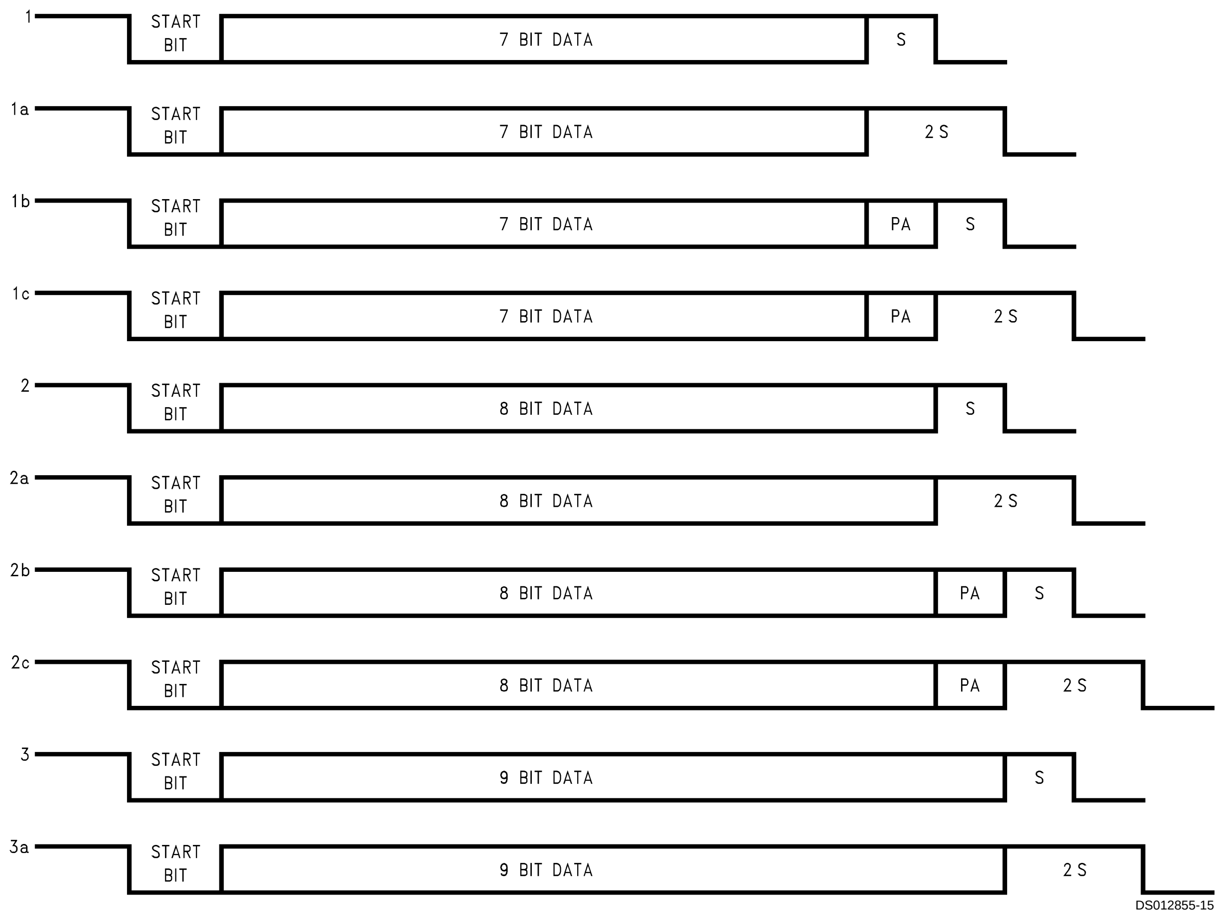


FIGURE 15. Framing Formats

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 UART 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 UART operation that the framing formats are the same for the transmitter and receiver.

UART INTERRUPTS

The UART 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 UART 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 16*).The divide factors are specified through two read/write registers shown in *Figure 17*. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

Baud Clock Generation (Continued)

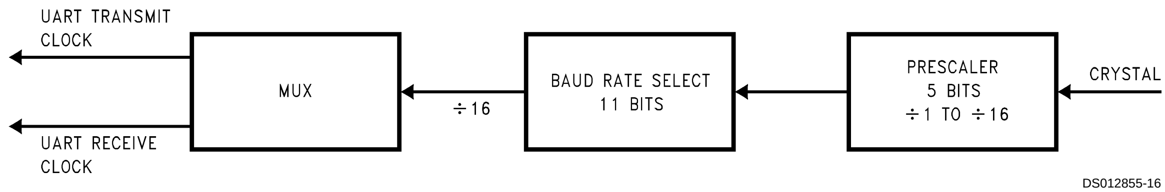


FIGURE 16. UART BAUD Clock Generation

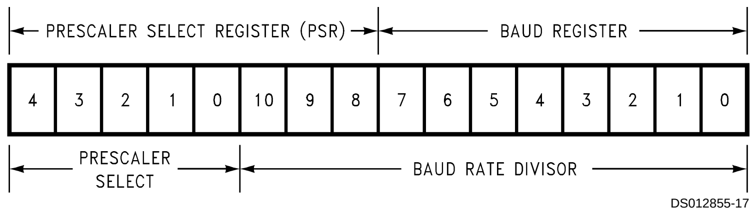


FIGURE 17. UART BAUD Clock Divisor Registers

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

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

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

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

Note 11: The entries in *Table 4* assume a prescaler output of 1.8432 MHz. In asynchronous mode the baud rate could be as high as 625k.

TABLE 5. Prescaler Factors

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

Baud Clock Generation (Continued)

TABLE 5. Prescaler Factors (Continued)

Prescaler Select	Prescaler Factor
11110	15.5
11111	16

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

4.608/1.8432 = 2.5

The 2.5 entry is available in *Table 5*. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (*Table 5*) 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

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART 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 x N x P)

Where:

BR is the Baud Rate

Fc is the CKI frequency

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

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

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

Example:

Asynchronous Mode:

Crystal Frequency = 5 MHz

Desired baud rate = 9600

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

N x P = (5 x 106)/(16 x 9600) = 32.552

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

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

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

Using the above values calculated for N and P:

BR = (5 x 106)/(16 x 5 x 6.5) = 9615.384

% error = (9615.385 − 9600)/9600 = 0.16

Effect of HALT/IDLE

The UART 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 UART 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 Wake Up 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 CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the UART. 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 UART.

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 UART 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 UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART 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 UART 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.

Interrupts

The device supports a vectored interrupt scheme. It supports a total of fourteen interrupt sources. *Table 6* lists all the pos-

Interrupts (Continued)

sible device interrupt sources, their arbitration rankings and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and one or more Pending bits. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_c cycles to execute.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

TABLE 6. Interrupt Vector Table

ARBITRATION RANKING	SOURCE DESCRIPTION		VECTOR* ADDRESS (Hi-Low Byte)
(1) Highest	Software		0yFE–0yFF
(2)	Reserved		0yFC–0yFD
(3)	External	G0	0yFA-0yFB
(4)	Timer T0	Underflow	0yF8–0yF9
(5)	Timer T1	T1A/Underflow	0yF6–0yF7
(6)	Timer T1	T1B	0yF4-0yF5
(7)	MICROWIRE/PLUS	Busy Low	0yF2–0yF3
(8)	Counters		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)	Capture Timer 1 and 2		0yE6–0yE7
(14)	Unused		0yE4–0yE5
(15)	Port L/Wakeup		0yE2–0yE3
(16) Lowest	Default VIS	Reserved	0yE0–0yE1

Note 12: *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 and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block (y ≠ 0).

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

Interrupts (Continued)

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1.

WARNING

A Default VIS interrupt handler routine must be present. As a minimum, this handler should confirm that the GIE bit is cleared (this indicates that the interrupt sequence has been taken), take care of any required housekeeping, restore ocntext and return. Some sort of Warm Restart procedure should be implemented. These events can occur without any error on the part of the system designer or programmer.

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.

Figure 18 shows the Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST 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. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

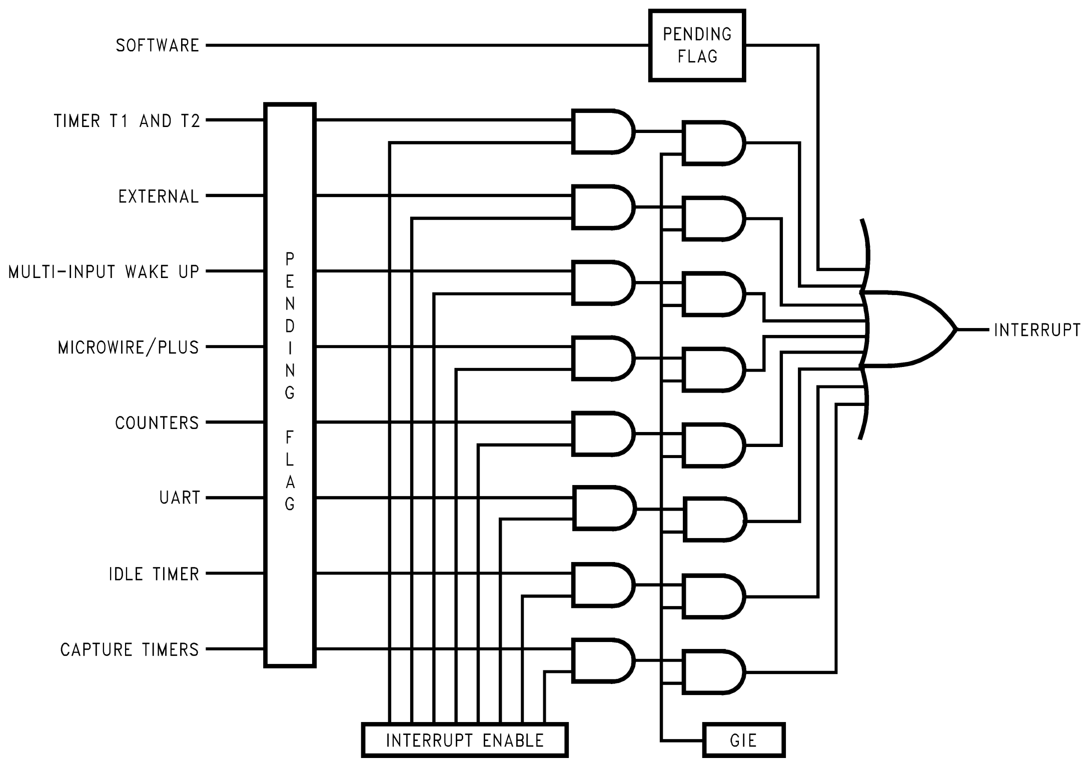


FIGURE 18. Interrupt Block Diagram

Detection of Illegal Conditions

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

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

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POR 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 3... etc.) is read as all 1's, which in turn will cause the program to return to address

Detection of Illegal Conditions (Continued)

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, E2PROMs 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.

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 7* details the different clock rates that may be selected.

TABLE 7. MICROWIRE/PLUS Master Mode Clock Select

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

Where t_c is the instruction cycle clock

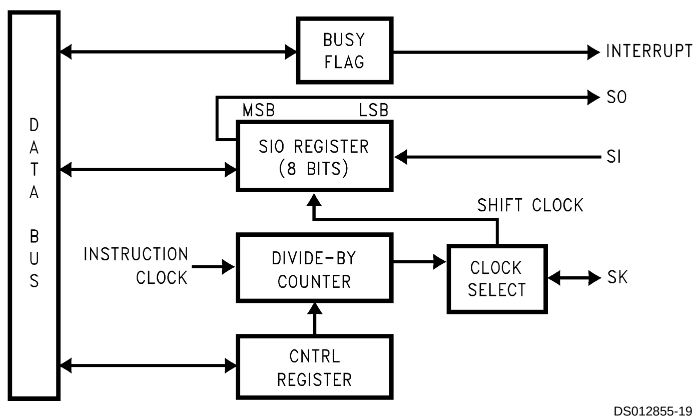


FIGURE 19. MICROWIRE/PLUS Block Diagram

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 8* 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 8* summarizes the settings required to enter the Slave mode of operation.

This table assumes that the control flag MSEL is set.

MICROWIRE/PLUS (Continued)

TABLE 8. MICROWIRE Mode Settings

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

The user must set the BUSY flag immediately upon entering the Slave mode. This 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.

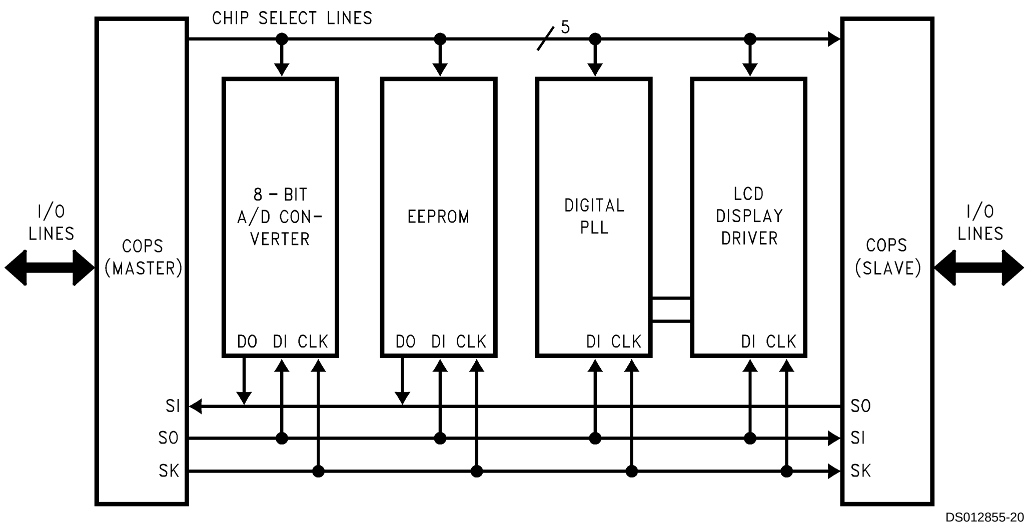


FIGURE 20. MICROWIRE/PLUS Application

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	112 On-Chip RAM Bytes
0070 to 007F	Unused RAM Address Space (reads as all 1's)
xx80 to xx8F	Unused RAM Address Space (reads undefined data)
xx90	Port E Data Register
xx91	Port E Configuration Register
xx92	Port E Input Pins (read only)
xx93	Reserved
xx94	Port F Data Register
xx95	Port F Configuration Register
xx96	Port F Input Pins (read only)

Memory Map (Continued)

ADDRESS S/ADD REG	CONTENTS
xx97	Reserved
xx98	Dividend or Result Byte (MDR1)
xx99	Dividend/Multiplier or Result Byte (MDR2)
xx9A	Dividend/Result Byte or Undefined (MDR3)
xx9B	Divisor/Multiplicand or Result Byte (MDR4)
xx9C	Divisor or Multiplicand Byte(MDR5)
xx9D	Multiply/Divide Control Register (MDCR)
xx9E	Counter Control 1 Register (CCR1)
xx9F	Counter Control 2 Register (CCR2)
xxA0	Counter 1 Prescaler Lower Byte (C1PRL)
xxA1	Counter 1 Prescaler Upper Byte (C1PRH)
xxA2	Counter 1 Count Register Lower Byte (C1CTL)
xxA3	Counter 1 Count Register Upper Byte (C1CTH)
xxA4	Counter 2 Prescaler Lower Byte (C2PRL)
xxA5	Counter 2 Prescaler Upper Byte (C2PRH)
xxA6	Counter 2 Count Register Lower Byte (C2CTL)
xxA7	Counter 2 Count Register Upper Byte (C2CTH)
xxA8	Counter 3 Prescaler Lower Byte (C3PRL)
xxA9	Counter 3 Prescaler Upper Byte (C3PRH)
xxAA	Counter 3 Count Register Lower Byte (C3CTL)
xxAB	Counter 3 Count Register Upper Byte (C3CTH)
xxAC	Counter 4 Prescaler Lower Byte (C4PRL)
xxAD	Counter 4 Prescaler Upper Byte (C4PRH)
xxAE	Counter 4 Count Register Lower Byte (C4CTL)
xxAF	Counter 4 Count Register Upper Byte (C4CTH)
xxB0	Capture Timer 1 Prescaler Register (CM1 PSC)
xxB1	Capture Timer 1 Lower Byte (CM1CRL) Read-Only
xxB2	Capture Timer 1 Upper Byte (CM1CRH) Read-Only
xxB3	Capture Timer 2 Prescaler Register (CM2PSC)
xxB4	Capture Timer 2 Lower Byte (CM2CRL) Read-Only
xxB5	Capture Timer 2 Upper Byte (CM2CRH) Read-Only
xxB6	Capture Timer 1 Control Register (CCMR1)
xxB7	Capture Timer 2 Control Register (CCMR2)
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 Prescaler 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

Memory Map (Continued)

ADDRESS S/ADD REG	CONTENTS
xxC7	Reserved
xxC8	MIWU Edge Select Register (WKEDG)
xxC9	MIWU Enable Register (WKEN)
xxCA	MIWU Pending Register (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 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 0200 to 027F 0300 to 037F	On Chip RAM Bytes (384 Bytes)

Note 13: Reading memory locations 0070H-007FH (Segment 0) will return all ones. Reading unused memory locations between 0080H-00F0 Hex (Segment 0) will return undefined data. Reading memory locations from other segments (i.e., segment 4, segment 5, etc.) will return all ones.

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.

Addressing Modes (Continued)

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 I 2 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 in the 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]
MemI	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

ADD	A,MemI	ADD	A←A + MemI
ADC	A,MemI	ADD with Carry	A←A + MemI + C, C←Carry, HC←Half Carry
SUBC	A,MemI	Subtract with Carry	A←A − MemI + C, C←Carry, HC←Half Carry
AND	A,MemI	Logical AND	A←A and $\overline{\text{MemI}}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,MemI	Logical OR	A←A or MemI

Instruction Set (Continued)

XOR	A,MemI	Logical EXclusive OR	A←A xor MemI
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,MemI	IF EQual	Compare A and MemI, Do next if A = MemI
IFNE	A,MemI	IF Not Equal	Compare A and MemI, Do next if A ≠ MemI
IFGT	A,MemI	IF Greater Than	Compare A and MemI, Do next if A > MemI
IFBNE	#	IF B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg←Reg − 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit #, A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A↔Mem
X	A,[X]	EXchange A with Memory [X]	A↔[X]
LD	A,MemI	LoaD A with Memory	A←MemI
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	DECrement A	A←A − 1
LAID		Load A InDirect from ROM	A←ROM (PU, A)
DCOR	A	Decimal CORrect A	A←BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C→A7→…→A0→C
RLC	A	Rotate A Left thru C	C←A7←…← A0←C
SWAP	A	SWAP nibbles of A	A7…A4↔A3…A0
SC		Set C	C←1, HC←1
RC		Reset C	C←0, HC←0
IFC		IF C	If C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP←SP + 1, A←[SP]
PUSH	A	PUSH A onto the stack	[SP]←A, SP←SP − 1
VIS		Vector to Interrupt Service Routine	PU←[VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC←ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9…0←i (i = 12 bits)
JP	Disp.	Jump relative short	PC←PC + r (r is −31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP]←PL, [SP − 1]←PU, SP − 2, PC←ii
JSR	Addr	Jump SubRoutine	[SP]←PL, [SP − 1] ← PU, SP − 2, PC9…0←i
JID		Jump InDirect	PL←ROM (PU, A)
RET		RETurn from subroutine	SP + 2, PL←[SP], PU ← [SP − 1]
RETSK		RETurn and SKip	SP + 2, PL←[SP], PU←[SP − 1], skip next instruction
RETI		RETurn from Interrupt	SP + 2, PL←[SP], PU ← [SP − 1], GIE←1
INTR		Generate an Interrupt	[SP]←PL, [SP − 1] ← PU, SP − 2, PC←0FF
NOP		No OPeration	PC←PC + 1

Instruction Execution Time

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

Most single byte instructions take one cycle time to execute.

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

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

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

Arithmetic and Logic Instructions

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

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
RLCA	1/1

SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

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

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. and 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.

Instruction Execution Time (Continued)

Opcode List

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

Where, #i is the immediate data Md is a directly addressed memory location * is an unused opcode

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

COP8 Tools Overview

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

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

SUMMARY OF TOOLS

COP8 Evaluation Software and Reference Designs

- **COP8–NSEVAL:** Software Evaluation package for Windows. A fully integrated evaluation environment for COP8. Includes WCOP8 IDE evaluation version (Integrated Development Environment), COP8-NSASM (Full COP8 Assembler), COP8-MLSIM (COP8 Instruction Level Simulator), COP8C Compiler Demo, DriveWay™ COP8 Device-Driver-Builder Demo, Manuals, Applications Software, and other COP8 technical information.
- **COP8–REF-xx:** Reference Designs for COP8 Families. Realtime hardware environment with a variety of functions for demonstrating the various capabilities and features of specific COP8 device families. Run Win 95 demo reference software and exercise specific device capabilities.

  Includes PCB with pre-programmed COP8, 9v battery for stand-alone operation, assembly listing, full applications source code, BOM, and schematics.

  (Add COP8-NSEVAL and an OTP programmer to implement your own software ideas in Assembly Code.)

COP8 Starter Kits and Hardware Target Solutions

- **COP8-EVAL-xxx:** A variety of Multifunction Evaluation, Design Test, and Target Boards for COP8 Families. Realtime target design environments with a selection of peripherals and features including multi I/O, LCD display, keyboard, A/D, D/A, EEPROM, USART, LEDs, and bread-board area. Quickly design, test, and implement a custom target system (some target boards are standalone, and ready for mounting into a standard enclosure), or just evaluate and test your code. Includes COP8-NSDEV with IDE and Assembler, software routines, reference designs, and source code (no p/s).

COP8 Software Development Languages and Integrated Environments

- **COP8-NSDEV:** National's COP8 Software Development package for Windows on CD. A fully Integrated Development Environment for COP8. Includes a fully licensed WCOP8 IDE, COP8-NSASM. Plus Manuals, Applications Software, and other COP8 technical information.
- **COP8C:** ByteCraft - C Cross-Compiler and Code Development System. Includes BCLIDE (Integrated Development Environment) for Win32, editor, optimizing C Cross-Compiler, macro cross assembler, BC-Linker, and MetaLinktools support. (DOS/SUN versions available; Compiler is linkable under WCOP8 IDE; Compatible with DriveWay COP8)
- **EWCOP8, EWCOP8-M, EWCOP8-BL:** IAR - ANSI C-Compiler and Embedded Workbench. (M version includes MetaLink debugger support) (BL version: 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.

COP8 Development Productivity Tools

- **DriveWay-COP8:** Aisys Corporation - COP8 Peripherals Code Generation tool. 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:** COP8 assembly code examples, device drivers, and utilities to speed up code development. (Included with COP8-NSDEV and COP8-NSEVAL.)
- **WCOP8 IDE:** KKD - COP8 IDE (Integrated Development Environment). 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 a single project window framework. (Included in COP8-NSDEV and COP8-NSEVAL.)

COP8 Hardware Debug Tools

- **COP8xx-DM:** Metalink COP8 Debug Module for non-flash COP8 Families. Windows based development and real-time in-circuit emulation tool, with 100 frame trace, 32k s/w breaks, Enhanced User Interface, MetaLinkDebugger, and COP8 OTP Programmer with sockets. Includes COP8-NSDEV, power supply, DIP and/or SMD emulation cables and adapters.
- **IM-COP8:** MetaLink iceMASTER® for non-flash COP8 devices. Windows based, full featured real-time in-circuit emulator, with 4k trace, 32k s/w breaks, and MetaLink-Windows Debugger. Includes COP8-NSDEV and power supply. Package-specific probes and surface mount adaptors are ordered separately. (Add COP8-PM and adapters for OTP programming.)

COP8 Development and OTP Programming Tools

- **COP8-PM:** COP8 Development Programming Module. Windows programming tool for COP8 OTP Families. Includes 40 DIP programming socket, control software, RS232 cable, and power supply. (SMD and 87Lxx programming adapters are extra.)
- **Development:** Metalink's Debug Module includes development device programming capability for COP8 devices. Many other third-party programmers are approved for development and engineering use.
- **Production:** Third-party programmers and automatic handling equipment cover needs from engineering prototype and pilot production, to full production environments.
- **Factory Programming:** Factory programming available for high-volume requirements.

COP8 Tools Overview (Continued)

TOOLS ORDERING NUMBERS FOR THE COP87L88RW DEVICES

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

Vendor	Tools	Order Number	Cost	Notes
National	COP8-NSEVAL	COP8-NSEVAL	Free	Web site download
	COP8-NSASM	COP8-NSASM	Free	Included in DM. Web site download
	COP8-MLSIM	COP8-MLSIM	Free	Included in DM. Web site download
	COP8-NSDEV	COP8-NSDEV	VL	Included in DM. Order CD from web site
	COP8-EM	Not available for this device		
	Development Devices	COP87L88RW		32k OTP only
MetaLink	COP8-DM	DM4-COP8-888GW plus PS-10, plus DM-COP8/68P6	M	Included p/s (PS-10), 68 target cable. Add OTP adapter
	DM Adapters	None needed		
	OTP Programming Adapters	MHW-COP8-PGMA-DS44-68P	L	For programming GW 68 PLCC on the DM
	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-888GW68PW-AD-10	M	10 MHz 68 PLCC probe card; 2.5V to 6.0V
	IM Probe Target Adapters	None needed		
Hilton	COP8-EVAL-HI	Not available for this device		
ICU	COP8-EVAL-ICU	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	Does not support this device		
OTP Programmers		Go to: www.national.com/cop8	L - H	A wide variety world-wide; use EDI programming adapter for COP887L88RW
Cost: Free; VL =< $100; L = $100 - $300; M = $300 - $1k; H = $1k - $3k; VH = $3k - $5k				

COP8 Tools Overview (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

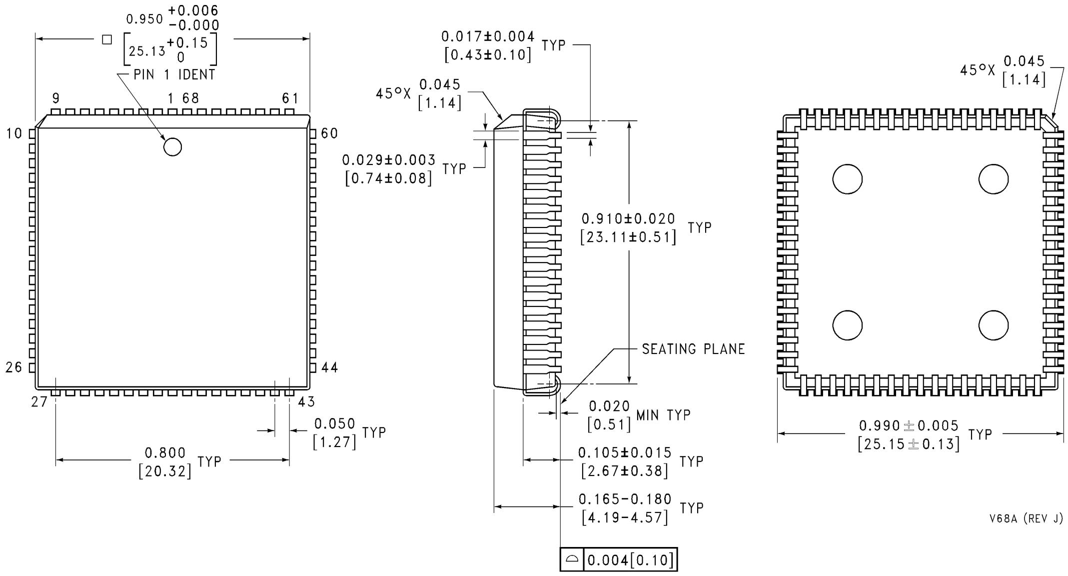
The following companies have approved COP8 programmers in a variety of configurations. Contact your local office or distributor. You can link to their web sites and get the latest listing of approved programmers from National's COP8 OTP Support page at: www.national.com/cop8.

Advantech; Dataman; EE Tools; Minato; 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



Plastic Leaded Chip Carrier (V)
Order Number COP87L88RWV-XE
NS Plastic Chip Package Number V68A

LIFE SUPPORT POLICY

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

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

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

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

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

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