

COP912C/COP912CH 8-Bit Microcontroller

General Description

The COP912C/COP912CH are members of the COP8™ 8-bit MicroController family. They are fully static Microcontrollers, fabricated using double-metal silicon gate micro-CMOS technology. These low cost MicroControllers are complete microcomputers containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the device to the specific application. The device operates over voltage ranges from 2.3V to 4.0V (COP912C) and from 4.0V to 5.5V (COP912CH). High throughput is achieved with an efficient, regular instruction set operating at a minimum of 2 μ s per instruction rate.

Key Features

- Lowest cost COP8 microcontroller
- 16-bit multi-function timer supporting
 - PWM mode
 - External event counter mode
 - Input capture mode
- 768 bytes of ROM
- 64 bytes of RAM

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 Port G
- MICROWIRE/PLUSTM™ Serial I/O
- Packages: 20 DIP/SO with 16 I/O pins

CPU/Instruction Set Features

- Instruction cycle time of 2 μ s for COP912CH and 2.5 μ s for COP912C
- Three multi-sourced interrupts servicing
 - External interrupt with selectable edge
 - Timer interrupt
 - Software interrupt
- Versatile and easy to use instruction set
- 8-bit Stack Pointer (SP)—stack in RAM
- Two 8-bit Register Indirect Memory Pointers (B, X)

Fully Static CMOS

- Low current drain (typically < 1 μ A)
- Single supply operation: 2.3V to 4.0V or 4.0V to 5.5V
- Temperature range: 0°C to +70°C

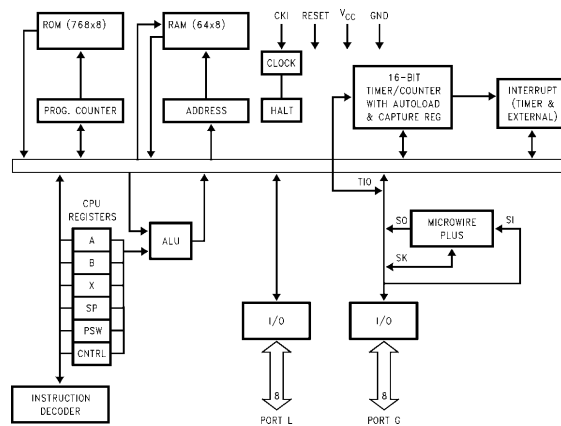
Development Support

- Emulation and OTP devices
- Real time emulation and full program debug offered by MetaLink Development System

Applications

- Electronic keys and switches
- Remote Control
- Timers
- Alarms
- Small industrial control units
- Low cost slave controllers
- Temperature meters
- Small domestic appliances
- Toys and games

Block Diagram



TRI-STATE® is a registered trademark of National Semiconductor Corporation.
 COP8™, MICROWIRE/PLUSTM™, WATCHDOG™ and MICROWIRE™ are trademarks of National Semiconductor Corporation.
 PC® is a registered trademark of International Business Machines Corp.
 iceMaster™ is a trademark of MetaLink Corporation.

Absolute Maximum Ratings

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

Supply Voltage (V_{CC}) 6.0V
Voltage at Any Pin $-0.3V$ to $V_{CC} + 0.3V$

Total Current into V_{CC} Pin (Source) 80 mA
Total Current out of GND Pin (Sink) 80 mA
Storage Temperature Range -65°C to $+150^{\circ}\text{C}$

Note: 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

COP912C/COP912CH; $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless other specified

| Parameter | Conditions | Min | Typ | Max | Units |
|---------------------------------------|---|--------------|---------------|---------------|---------------|
| Operating Voltage | | | | | |
| 912C | | 2.3 | | 4.0 | V |
| 912CH | | 4.0 | | 5.5 | V |
| Power Supply Ripple 1 (Note 1) | Peak to Peak | | | $0.1 V_{CC}$ | V |
| Supply Current (Note 2) | | | | | |
| CKI = 4 MHz | $V_{CC} = 5.5V$, $t_c = 2.5 \mu\text{s}$ | | | 6.0 | mA |
| CKI = 4 MHz | $V_{CC} = 4.0V$, $t_c = 2.5 \mu\text{s}$ | | | 2.5 | mA |
| HALT Current | $V_{CC} = 5.5V$, CKI = 0 MHz | | < 1 | 8 | μA |
| INPUT LEVELS (V_{IH} , V_{IL}) | | | | | |
| Reset, CKI: | | | | | |
| Logic High | | $0.9 V_{CC}$ | | | V |
| Logic Low | | | | $0.1 V_{CC}$ | V |
| All Other Inputs | | | | | |
| Logic High | | $0.7 V_{CC}$ | | | V |
| Logic Low | | | | $0.2 V_{CC}$ | V |
| Hi-Z Input Leakage/TRI-STATE Leakage | $V_{CC} = 5.5V$ | -2 | | +2 | μA |
| Input Pullup Current | $V_{CC} = 5.5V$ | | | 250 | μA |
| G-Port Hysteresis | | | $0.05 V_{CC}$ | $0.35 V_{CC}$ | V |
| Output Current Levels | | | | | |
| Source (Push-Pull Mode) | $V_{CC} = 4.0V$, $V_{OH} = 3.8V$ | 0.4 | | | mA |
| | $V_{CC} = 2.3V$, $V_{OH} = 1.8V$ | 0.2 | | | mA |
| Sink (Push-Pull Mode) | $V_{CC} = 4.0V$, $V_{OL} = 1.0V$ | 4.0 | | | mA |
| | $V_{CC} = 2.3V$, $V_{OL} = 0.4V$ | 0.7 | | | mA |
| Allowable Sink/Source Current Per Pin | | | | 3 | mA |
| Input Capacitance (Note 3) | | | | 7 | pF |
| Load Capacitance on D2 (Note 3) | | | | 1000 | pF |

Note 1: Rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: Characterized, not tested.

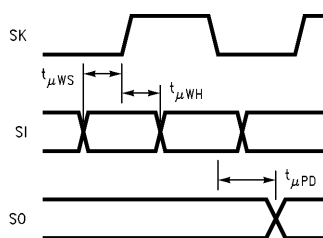
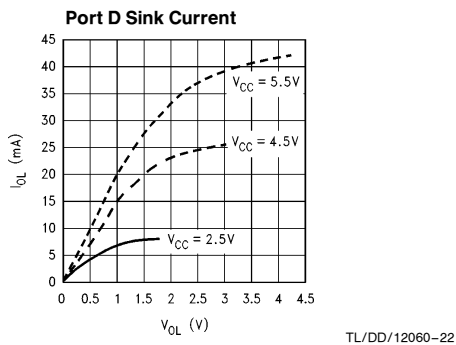
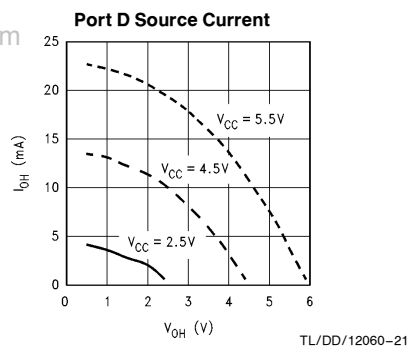
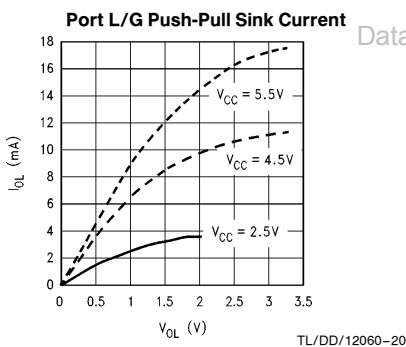
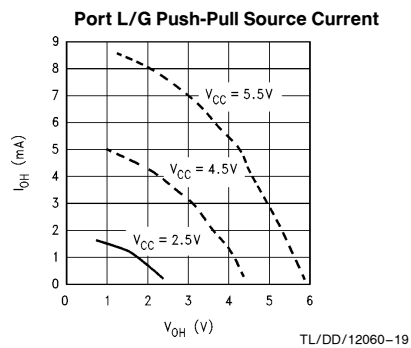
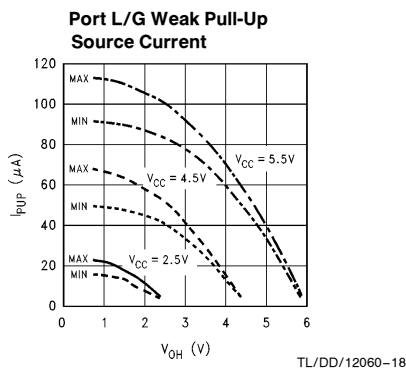
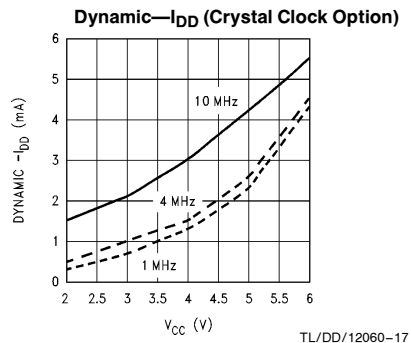
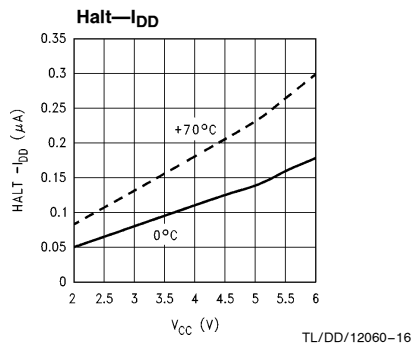


FIGURE 1. MICROWIRE/PLUS Timing

TL/DD/12060-2

Typical Performance Characteristics



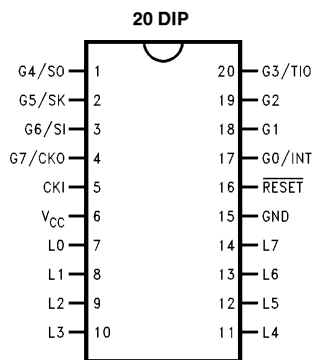
AC Electrical Characteristics COP912C/COP912CH; $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ unless otherwise specified

| Parameter | Conditions | Min | Typ | Max | Units |
|---|--|---------|-----|------|---------------|
| INSTRUCTION CYCLE TIME (t_c) Crystal/Resonator | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | 2 | | DC | μs |
| | $2.3\text{V} \leq V_{CC} < 4.0\text{V}$ | 2.5 | | DC | μs |
| | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | 3 | | DC | μs |
| | $2.3\text{V} \leq V_{CC} < 4.0\text{V}$ | 7.5 | | DC | μs |
| Inputs t_{Setup} t_{Hold} | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | 200 | | | ns |
| | $2.3\text{V} \leq V_{CC} < 4.0\text{V}$ | 500 | | | ns |
| | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | 60 | | | ns |
| | $2.3\text{V} \leq V_{CC} < 4.0\text{V}$ | 150 | | | ns |
| Output Propagation Delay t_{PD1} , t_{PD0} SO, SK All Others | $R_L = 2.2\text{ k}\Omega$, $C_L = 100\text{ pF}$ | | | | |
| | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | | | 0.7 | μs |
| | $2.3\text{V} \leq V_{CC} < 4.0\text{V}$ | | | 1.75 | μs |
| | $4.0\text{V} \leq V_{CC} \leq 5.5\text{V}$ | | | 1 | μs |
| Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time | | | | | |
| | | 1 t_c | | | |
| | | 1 t_c | | | |
| | | 1 t_c | | | |
| MICROWIRE Setup Time ($t_{\mu\text{WS}}$) MICROWIRE Hold Time ($t_{\mu\text{WH}}$) MICROWIRE Output Propagation Delay ($t_{\mu\text{PD}}$) | | 20 | | | ns |
| | | 56 | | | ns |
| | | | | 220 | ns |
| | | | | | |
| Reset Pulse Width | | 1.0 | | | μs |

COP912C/COP912CH Pinout

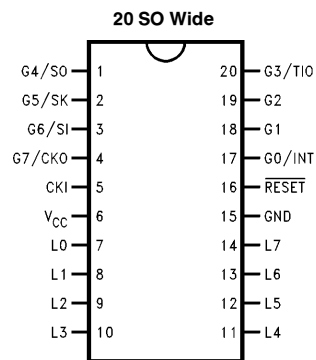
DataSheet4U.com

Top View



TL/DD/12060-3

Order Number COP912C-XXX/N, COP912CH-XXX/N



TL/DD/12060-4

Order Number COP912C-XXX/WM,
COP912CH-XXX/WM

FIGURE 2. COP912C/COP912CH Pinout

Pin Description

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

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT L is an 8-bit I/O port.

There are two registers associated to configure the L port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

| Port L Config. | Port L Data | PORT L Setup |
|----------------|-------------|-------------------------|
| 0 | 0 | Hi-Z Input (TRI-STATE) |
| 0 | 1 | Input with Weak Pull-Up |
| 1 | 0 | Push-Pull Zero Output |
| 1 | 1 | Push-Pull One Output |

Three data memory address locations are allocated for this port, one each for data register [00D0], configuration register [00D1] and the input pins [00D2].

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7).

All eight G-pins have Schmitt Triggers on the inputs.

There are two registers associated to configure the G port: a data register and a configuration register. Therefore each G port bit can be individually configured under software control as shown below:

| Port G Config. | Port G Data | PORT G Setup |
|----------------|-------------|-------------------------|
| 0 | 0 | Hi-Z Input (TRI-STATE) |
| 0 | 1 | Input with Weak Pull-Up |
| 1 | 0 | Push-Pull Zero Output |
| 1 | 1 | Push-Pull One Output |

Three data memory address locations are allocated for this port, one for data register [00D4], one for configuration register [00D5] and one for the input pins [00D6]. Since G6 and G7 are Hi-Z input only pins, any attempt by the user to configure them as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeroes. Note that the chip will be placed in the Halt mode by writing a "1" to the G7 data bit.

Six pins of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input/general purpose input (if clock option is R/C- or external clock)

Pins G1 and G2 currently do not have any alternate functions.

The selection of alternate Port G functions are done through registers PSW [00EF] to enable external interrupt and CNTRL [00EE] to select TIO and MICROWIRE operations.

Functional Description

The internal architecture is shown in the block diagram. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operations in one cycle time. There are five CPU registers:

A is the 8-bit Accumulator register

PC is the 15-bit Program Counter register

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

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

B is the 8-bit address register and can be auto incremented or decremented

X is the 8-bit alternate address register and can be auto incremented or decremented.

SP is the 8-bit stack pointer which points to the subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns. The SP must be preset by software upon initialization.

MEMORY

The memory is separated into two memory spaces: program and data.

PROGRAM MEMORY

Program memory consists of 768 x 8 ROM. These bytes of ROM may be instructions or constant data. The memory is addressed by the 15-bit program counter (PC). There are no "pages" of ROM, the PC counts all 15 bits. ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

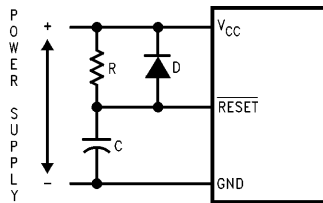
The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly through B, X and SP registers. The device has 64 bytes of RAM. Sixteen bytes of RAM are mapped as "registers", these can be loaded immediately, decremented and tested. Three specific registers: X, B, and SP are mapped into this space, the other registers are available for general usage.

Any bit of data memory can be directly set, reset or tested. 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.

RESET

The RESET input pin when pulled low initializes the microcontroller. Upon initialization, the ports L and G are placed in the TRI-STATE mode. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for ports L and G are cleared. The external RC network shown in *Figure 3* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.

Functional Description (Continued)



TL/DD/12060-5

$$RC > 5 \times \text{POWER SUPPLY RISE TIME}$$

FIGURE 3. Recommended Reset Circuit

OSCILLATOR CIRCUITS

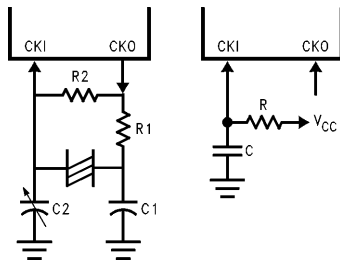
The device can be driven by a clock input which can be between DC and 5 MHz.

CRYSTAL OSCILLATOR

By selecting CKO as a clock output, CKI and CKO can be connected to create a crystal controlled oscillator. Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator, CKI can make an R/C oscillator. CKO is available as a general purpose input and/or HALT control. Table II shows variation in the oscillator frequencies as functions of the component (R and C) value.



TL/DD/12060-6

FIGURE 4. Clock Oscillator Configurations

TABLE I. Crystal Oscillator Configuration

| R1 (k Ω) | R2 (m Ω) | C1 (pF) | C2 (pF) | CKI Freq. (MHz) |
|------------------|------------------|---------|---------|-----------------|
| 0 | 1 | 30 | 30-36 | 5 |
| 0 | 1 | 30 | 30-36 | 4 |
| 5.6 | 1 | 200 | 100-150 | 0.455 |

TABLE II. RC Oscillator Configuration (Part-to-Part Variation, $T_A = 25^\circ\text{C}$)

| R (k Ω) | C (pF) | CKI Freq. (MHz) | Intr. Cycle (μs) |
|-----------------|--------|-----------------|-------------------------------|
| 3.3 | 82 | 2.2 to 2.7 | 3.7 to 4.6 |
| 5.6 | 100 | 1.1 to 1.3 | 7.4 to 9 |
| 6.8 | 100 | 0.9 to 1.1 | 8.8 to 10.8 |

Note: $3\text{k} \leq R \leq 200\text{ k}\Omega$, $50\text{ pF} \leq C \leq 200\text{ pF}$.

HALT MODE

The device is a fully static device. The device enters the HALT mode by writing a one to the G7 bit of the G data register. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. In this mode the chip will only draw leakage current.

The device supports two different ways of exiting the HALT mode. The first method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO is a dedicated output), and so may be used either with an RC clock configuration (or an external clock configuration). The second method of exiting the HALT mode is to pull the RESET low.

Note: To allow clock resynchronization, it is necessary to program two NOP's immediately after the device comes out of the HALT mode. The user must program two NOP's following the "enter HALT mode" (set G7 data bit) instruction.

Functional Description (Continued)

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, EEPROMS 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 5* shows a block diagram of the MICROWIRE logic.

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

The CNTRL register is used to configure and control the MICROWIRE. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register.

The following table details the different clock rates that may be selected.

SK Divide Clock Rates

| SL1 | SL0 | SK |
|-----|-----|--------|
| 0 | 0 | 2 x tc |
| 0 | 1 | 4 x tc |
| 1 | x | 8 x tc |

Where tc is the instruction cycle clock.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The device may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 5* shows how two microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

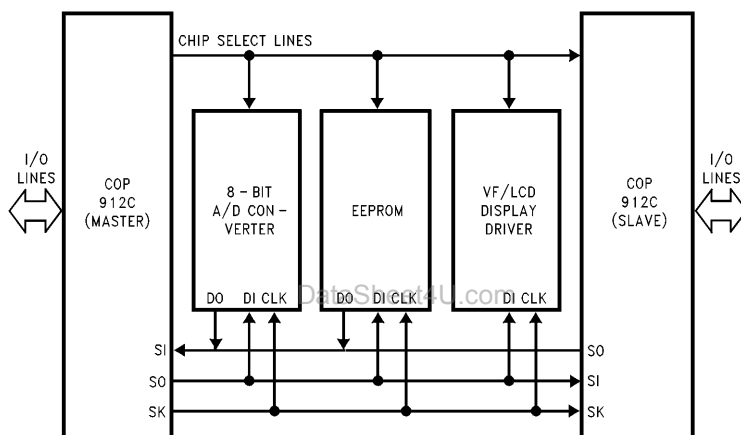


FIGURE 5. MICROWIRE/PLUS Application

TL/DD/12060-7

Functional Description (Continued)

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.

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.

Table III summarizes the settings required to enter the Master/Slave modes of operations.

The table assumes that the control flag MSEL is set.

TABLE III. MICROWIRE/PLUS G Port Configuration

| G4 (SO) Config. Bit | G5 (SK) Config. Bit | G4 Pin | G5 Pin | G6 Pin | Operation |
|---------------------|---------------------|-----------|---------|--------|------------------|
| 1 | 1 | SO | Int. SK | SI | MICROWIRE Master |
| 0 | 1 | TRI-STATE | Int. SK | SI | MICROWIRE Master |
| 1 | 0 | SO | Ext. SK | SI | MICROWIRE Slave |
| 0 | 0 | TRI-STATE | Ext. SK | SI | MICROWIRE Slave |

MICROWIRE/PLUS MASTER MODE OPERATION

In MICROWIRE/PLUS Master mode operation, the SK shift clock is generated internally. The MSEL bit in the CNTRL register must be set to allow the SK and SO functions onto the G5 and G4 pins. The G5 and G4 pins must also be selected as outputs by setting the appropriate bits in the Port G configuration register. The MICROWIRE Master mode always initiates all data exchanges. The MSEL bit in the CNTRL register is set to enable MICROWIRE/PLUS. G4 and G5 are selected as output.

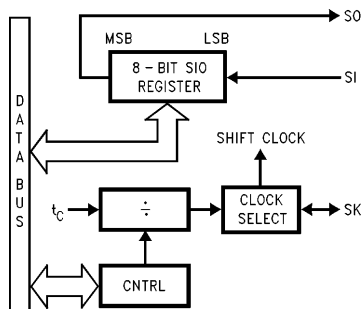


FIGURE 6. MICROWIRE/PLUS Block Diagram

MICROWIRE/PLUS SLAVE MODE

In MICROWIRE/PLUS Slave mode operation, the SK shift 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 as an output by resetting and setting their respective bits in the G port configuration register.

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 in properly. After eight clock pulses, the BUSY flag will be cleared and the sequence may be repeated.

Note: In the Slave mode the SIO register does not stop shifting even after the busy flag goes low. Since SK is an external output, the SIO register stops shifting only when SK is turned off by the master.

Note: 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 register to be narrow. When the BUSY flag is set, the MICROWIRE logic becomes active with the internal SIO shift clock enabled. If SK is high in slave mode, this will cause the internal shift clock to go from low in standby mode to high in active mode. This generates a rising edge, and causes one bit to be shifted into the SIO register from the SI input. For safety, the BUSY flag should only be set when the input SK clock is low.

Note: The SIO register must be loaded only when the SK shift clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register.

Timer/Counter

The device has an on board 16-bit timer/counter (organized as two 8-bit registers) with an associated 16-bit autoreload/capture register (also organized as two 8-bit registers). Both are read/write registers.

The timer has three modes of operation:

PWM (PULSE WIDTH MODULATION) MODE

The timer counts down at the instruction cycle rate (2 μ s max). When the timer count underflows, the value in the autoreload register is copied into the timer. Consequently, the timer is programmable to divide by any value from 1 to 65536. Bit 5 of the timer CNTRL register selects the timer underflow to toggle the G3 output. This allows the user to generate a square wave output or a pulse-width-modulated output. The timer underflow can also be enabled to interrupt the processor. The timer PWM mode is shown in Figure 7.

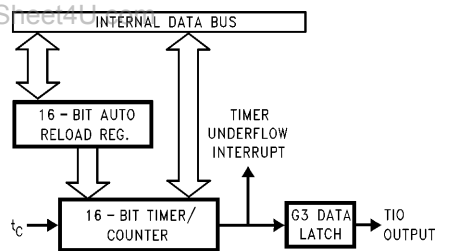


FIGURE 7. Timer in PWM Mode

Functional Description (Continued)

EXTERNAL EVENT COUNTER MODE

In this mode, the timer becomes a 16-bit external event counter, clocked from an input signal applied to the G3 input. The maximum frequency for this G3 input clock is 250 kHz (half of the 0.5 MHz instruction cycle clock). When the external event counter underflows, the value in the autoreload register is copied into the timer. This timer underflow may also be used to generate an interrupt. Bit 5 of the CNTRL register is used to select whether the external event counter clocks on positive or negative edges from the G3 input. Consequently, half cycles of an external input signal could be counted. The External Event counter mode is shown in Figure 8.

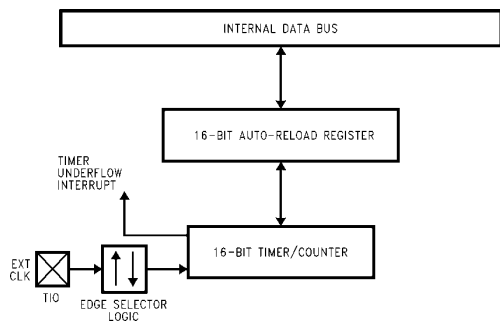


FIGURE 8. Timer in External Event Mode

INPUT CAPTURE MODE

In this mode, the timer counts down at the instruction clock rate. When an external edge occurs on pin G3, the value in the timer is copied into the capture register. Consequently, the time of an external edge on the G3 pin is "captured". Bit 5 of the CNTRL register is used to select the polarity of the external edge. This external edge capture can also be programmed to generate an interrupt. The duration of an input signal can be computed by capturing the time of the leading edge, saving this captured value, changing the capture edge, capturing the time of the trailing edge, and then subtracting this trailing edge time from the earlier leading edge time. The Input Capture mode is shown in Figure 9.

grammed to generate an interrupt. The duration of an input signal can be computed by capturing the time of the leading edge, saving this captured value, changing the capture edge, capturing the time of the trailing edge, and then subtracting this trailing edge time from the earlier leading edge time. The Input Capture mode is shown in Figure 9.

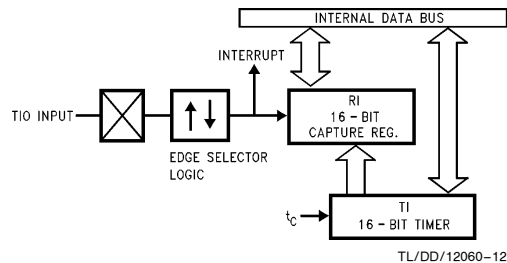


FIGURE 9. Timer in Input Capture Mode

Table IV below details the TIMER modes of operation and their associated interrupts. Bit 4 of CNTRL is used to start and stop the timer/counter. Bits 5, 6 and 7 of the CNTRL register select the timer modes. The ENTI (Enable Timer Interrupt) and TPND (Timer Interrupt Pending) bits in the PSW register are used to control the timer interrupts.

Care must be taken when reading from and writing to the timer and its associated autoreload/capture register. The timer and autoreload/capture register are both 16-bit, but they are read from and written to one byte at a time. It is recommended that the timer be stopped before writing a new value into it. The timer may be read "on the fly" without stopping it if suitable precautions are taken. One method of reading the timer "on the fly" is to read the upper byte of the timer first, and then read the lower byte. If the most significant bit of the lower byte is then tested and found to be high, then the upper byte of the timer should be read again and this new value used.

TABLE IV. Timer Modes and Control Bits

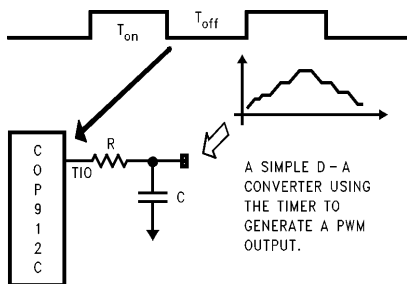
| CNTRL Bits | | | Operation Mode | Timer Interrupt | Timer Counts On |
|------------|---|---|---|-------------------|-------------------|
| 7 | 6 | 5 | | | |
| 0 | 0 | 0 | External Event Counter with Autoreload Register | Timer Underflow | TIO Positive Edge |
| 0 | 0 | 1 | External Event Counter with Autoreload Register | Timer Underflow | TIO Negative Edge |
| 0 | 1 | 0 | Not Allowed | Not Allowed | Not Allowed |
| 0 | 1 | 1 | Not Allowed | Not Allowed | Not Allowed |
| 1 | 0 | 0 | Timer with Autoreload Register | Timer Underflow | tc |
| 1 | 0 | 1 | Timer with Autoreload Register and Toggle TIO Out | Timer Underflow | tc |
| 1 | 1 | 0 | Timer with Capture Register | TIO Positive Edge | tc |
| 1 | 1 | 1 | Timer with Capture Register | TIO Negative Edge | tc |

Functional Description (Continued)

TIMER APPLICATION EXAMPLE

The timer has an autoreload register that allows any frequency to be programmed in the timer PWM mode. The timer underflow can be programmed to toggle output bit G3, and may also be programmed to generate a timer interrupt. Consequently, a fully programmable PWM output may be easily generated.

The timer counts down and when it underflows, the value from the autoreload register is copied into the timer. The CNTRL register is programmed to both toggle the G3 output and generate a timer interrupt when the timer underflows. Following each timer interrupt, the user's program alternately loads the values of the "on" time and the "off" time into the timer autoreload register. Consequently, a pulse-width-modulated (PWM) output waveform is generated to a resolution of one instruction cycle time. This PWM application example is shown in Figure 10.



TL/DD/12060-13

FIGURE 10. Timer Based PWM Application

Interrupts

There are three interrupt sources:

1. A maskable interrupt on external G0 input positive or negative edge sensitive under software control
 2. A maskable interrupt on timer underflow or timer capture
 3. A non-maskable software/error interrupt on opcode zero.
- The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled. IEDG selects the external interrupt edge (1 = rising edge, 0 = falling edge). The user can get an interrupt on both rising

and falling edges by toggling the state of IEDG bit after each interrupt.

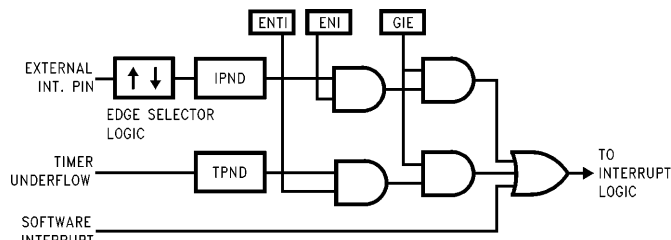
IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. The user can prioritize the interrupt and clear the pending bit that corresponds to the interrupt being serviced. The user can also enable GIE at this point for nesting interrupts. Two things have to be kept in mind when using the software interrupt. The first is that executing a simple RET instruction will take the program control back to the software interrupt instruction itself. In other words, the program will be stuck in an infinite loop. To avoid the infinite loop, the software interrupt service routine should end with a RETSK instruction or with a JMP instruction. The second thing to keep in mind is that unlike the other interrupt sources, the software interrupt does not reset the GIE bit. This means that the device can be interrupted by other interrupt sources while servicing the software interrupt.

Interrupts push the PC to the stack, reset the GIE bit to disable further interrupts and branch to address 00FF. The RETI instruction will pop the stack to PC and set the GIE bit to enable further interrupts. The user should use the RETI or the RET instruction when returning from a hardware (maskable) interrupt subroutine. The user should use the RETSK instruction when returning from a software interrupt subroutine to avoid an infinite loop situation.

The software interrupt is a special kind of non-maskable interrupt which occurs when the INTR instruction (opcode 00 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 the software interrupt 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.

Hardware and Software interrupts are treated differently. The software interrupt is not gated by the GIE bit. However, it has the lowest arbitration ranking. Also the fact that all interrupts vector to the same address 00FF Hex means that a software interrupt happening at the same time as a hardware interrupt will be missed.

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.



TL/DD/12060-14

FIGURE 11. Interrupt Block Diagram

Interrupts (Continued)

DETECTION OF ILLEGAL CONDITIONS

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

Note: A software interrupt is acted upon only when a timer or external interrupt is not pending as hardware interrupts have priority over software interrupt. In addition, the Global Interrupt bit is not set when a software interrupt is being serviced thereby opening the door for the hardware interrupts to occur. The subroutine stack grows down for each call and grows up for each return. If the stack pointer is initialized to 2F Hex, then if there are more returns than calls, the stack pointer will point to addresses 30 and 31 (which are undefined RAM). Undefined RAM is read as all 1's, thus, the program will return to address FFFF. This is a undefined ROM location and the instruction fetched will generate a software interrupt signalling an illegal condition. The device can detect the following illegal conditions:

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

Illegal conditions may occur from coding errors, "brown out" voltage drops, static, supply noise, etc. 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 RESET but might not clear the RAM). Examination of the stack can help in identifying the source of the error. For example, upon a software interrupt, if the SP = 30, 31 it implies that the stack was over "POP"ed (with the SP = 2F hex initially). If the SP contains a legal value (less than or equal to the initialized SP value), then the value in the PC gives a clue as to where in the user program an attempt to access an illegal (an address over 300 Hex) was made. The opcode returned in this case is 00 which is a software interrupt.

The detection of illegal conditions is illustrated with an example:

```
0043 CLRA
0044 RC
0045 JMP 04FF
0046 NOP
```

When the device is executing this program, it seemingly "locks-up" having executed a software interrupt. To debug this condition, the user takes a look at the SP and the contents of the stack. The SP has a legal value and the contents of the stack are 04FF. The perceptive user immediately realizes that an illegal ROM location (04FF) was accessed and the opcode returned (00) was a software interrupt. Another way to decode this is to run a trace and follow the sequence of steps that ended in a software interrupt. The damaging jump statement is changed.

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE control register contains the following bits:

| | |
|-------------|---|
| SL1 and SL0 | Select the MICROWIRE clock divide-by (00 = 2, 01 = 4, 1x = 8) |
| IEDG | External interrupt edge polarity select |
| MSEL | Selects G5 and G4 as MICROWIRE signals SK and SO respectively |
| TRUN | Used to start and stop the timer/counter (1 = run, 0 = stop) |
| TC1 | Timer Mode Control Bit |
| TC2 | Timer Mode Control Bit |
| TC3 | Timer Mode Control Bit |

| | | | | | | | |
|-----|-----|-----|------|------|------|-----|-----|
| 7 | | | | | | | 0 |
| TC1 | TC2 | TC3 | TRUN | MSEL | IEDG | SL1 | SL0 |

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

| | |
|------|---|
| GIE | Global interrupt enable (enables interrupts) |
| ENI | External interrupt enable |
| BUSY | MICROWIRE busy shifting flag |
| IPND | External interrupt pending |
| ENTI | Timer interrupt enable |
| TPND | Timer interrupt pending (timer underflow or capture edge) |
| C | Carry Flip/flop |
| HC | Half carry Flip/flop |

| | | | | | | | |
|----|---|------|------|------|------|-----|-----|
| 7 | | | | | | | 0 |
| HC | C | TPND | ENTI | IPND | BUSY | ENI | GIE |

The Half-Carry bit is also effected by all the instructions that effect the Carry flag. The flag values depend upon the instruction. For example, after executing the ADC instruction the values of the Carry and the Half-Carry flag depend upon the operands involved. However, instructions like SET C and RESET C will set and clear both the carry flags. Table V lists out the instructions that effect the HC and the C flags.

TABLE V. Instructions Effecting HC and C Flags

| Instr. | HC Flag | C Flag |
|---------|---------------------|---------------------|
| ADC | Depends on Operands | Depends on Operands |
| SUBC | Depends on Operands | Depends on Operands |
| SETC | Set | Set |
| RESET C | Set | Set |
| RRC | Depends on Operands | Depends on Operands |

MEMORY MAP

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

Control Registers (Continued)**TABLE VI. Memory Map**

| Address | Contents |
|----------|--|
| 00 to 2F | On-chip RAM Bytes (48 Bytes) |
| 30 to 7F | Unused RAM Address Space (Reads as all ones) |
| 80 to BF | Expansion Space for On-Chip EERAM (Reads Undefined Data) |
| C0 to CF | Expansion Space for I/O and Registers |
| D0 | Port L Data Register |
| D1 | Port L Configuration Register |
| D2 | Port L Input Pins (read only) |
| D3 | Reserved for Port L |
| D4 | Port G Data Register |
| D5 | Port G Configuration Register |
| D6 | Port G Input Pins (read only) |
| D7 | Reserved |
| D8 to DB | Reserved |
| DC to DF | Reserved |
| E0 to EF | On-Chip Functions and Registers |
| E0 to E7 | Reserved for Future Parts |
| E8 | Reserved |
| E9 | MICROWIRE Shift Register |
| EA | Timer Lower Byte |
| EB | Timer Upper Byte |
| EC | Timer Autoreload Register Lower Byte |
| ED | Timer Auto reload Register Upper Byte |
| EE | CNTRL Control Register |
| EF | PSW Register |
| F0 to FF | On-Chip RAM Mapped as Registers (16 Bytes) |
| FC | X Register |
| FD | SP Register |
| FE | B Register |

Reading other unused memory locations will return undefined data.

Addressing Modes

The device has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES**Register Indirect**

This is the “normal” addressing mode for the chip. The operand is the data memory addressed by the **B** or **X** pointer.

Register Indirect With Auto Post Increment Or Decrement

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the **B** or **X** pointer. This is a register indirect mode that automatically post increments or post decrements the **B** or **X** pointer 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 issued with the LD B,# instruction, where the immediate # is less than 16. 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 produce the next instruction address. JP has a range from -31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no “blocks” or “pages” when using JP since all 15 bits of the PC are used.

Absolute

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

Absolute Long

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

Indirect

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

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

| | |
|-----|---|
| A | 8-Bit Accumulator Register |
| B | 8-Bit Address Register |
| X | 8-Bit Address Register |
| SP | 8-Bit Stack Pointer Register |
| S | 8-Bit Data Segment Address 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 |

Symbols

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

Instruction Set (Continued)**TABLE VII. Instruction Set**

| Instr | | Function | Register Operation |
|-------|-----------------|-----------------------------|---|
| ADD | A, Meml | Add | $A \leftarrow A + \text{Meml}$ |
| ADC | A, Meml | Add with Carry | $A \leftarrow A + \text{Meml} + C, C \leftarrow \text{Carry}$ |
| SUBC | A, Meml | Subtract with Carry | $A \leftarrow A - \text{Meml} + C, C \leftarrow \text{Carry}$ |
| AND | A, Meml | Logical AND | $A \leftarrow A \text{ and Meml}$ |
| OR | A, Meml | Logical OR | $A \leftarrow A \text{ or Meml}$ |
| XOR | A, Meml | Logical Exclusive-OR | $A \leftarrow A \text{ xor Meml}$ |
| IFEQ | A, Meml | IF Equal | Compare A and Meml, Do Next if $A = \text{Meml}$ |
| IFGT | A, Meml | IF Greater than | Compare A and Meml, Do Next if $A > \text{Meml}$ |
| IFBNE | # | IF B not Equal | Do Next If Lower 4 Bits of B not = Imm |
| DRSZ | Reg | Decrement Reg, Skip if Zero | $\text{Reg} \leftarrow \text{Reg} - 1$, Skip if Reg Goes to Zero |
| SBIT | #, Mem | Set Bit | 1 to Mem.Bit (Bit = 0 to 7 Immediate) |
| RBIT | #, Mem | Reset Bit | 0 to Mem.Bit (Bit = 0 to 7 Immediate) |
| IFBIT | #, Mem | If Bit | If Mem.Bit is True, Do Next Instruction |
| X | A, Mem | Exchange A with Memory | $A \leftrightarrow \text{Mem}$ |
| LD | A, Meml | Load A with Memory | $A \leftarrow \text{Meml}$ |
| LD | Mem, Imm | Load Direct Memory Immed. | $\text{Mem} \leftarrow \text{Imm}$ |
| LD | Reg, Imm | Load Register Memory Immed. | $\text{Reg} \leftarrow \text{Imm}$ |
| X | A, [B \pm] | Exchange A with Memory [B] | $A \leftrightarrow [B] (B \leftarrow B \pm 1)$ |
| X | A, [X \pm] | Exchange A with Memory [X] | $A \leftrightarrow [X] (X \leftarrow X \pm 1)$ |
| LD | A, [B \pm] | Load A with Memory [B] | $A \leftarrow [B] (B \leftarrow B \pm 1)$ |
| LD | A, [X \pm] | Load A with Memory [X] | $A \leftarrow [X] (X \leftarrow X \pm 1)$ |
| LD | [B \pm], Imm | Load Memory Immediate | $[B] \leftarrow \text{Imm} (B \leftarrow B \pm 1)$ |
| CLRA | | Clear A | $A \leftarrow 0$ |
| INC | | Increment A | $A \leftarrow A + 1$ |
| DEC | | Decrement A | $A \leftarrow A - 1$ |
| LAID | A | Load A Indirect from ROM | $A \leftarrow \text{ROM}(\text{PU}, A)$ |
| DCOR | A | Decimal Correct A | $A \leftarrow \text{BCD Correction (follows ADC, SUBC)}$ |
| RRC | | Rotate Right Through Carry | $C \rightarrow A7 \rightarrow \dots \rightarrow A0 \rightarrow C$ |
| SWAP | A | Swap Nibbles of A | $A7 \dots A4 \leftrightarrow A3 \dots A0$ |
| SC | A | Set C | $C \leftarrow 1$ |
| RC | A | Reset C | $C \leftarrow 0$ |
| IFC | | If C | If C is True, do Next Instruction |
| IFNC | | If Not C | If C is not True, do Next Instruction |
| JMPL | | Jump Absolute Long | $\text{PC} \leftarrow \text{ii} (\text{ii} = 15 \text{ Bits}, 0\text{k to } 32\text{k})$ |
| JMP | | Jump Absolute | $\text{PC}11 \dots \text{PC}0 \leftarrow \text{i} (\text{i} = 12 \text{ Bits})$ $\text{PC}15 \dots \text{PC}12 \text{ Remain Unchanged}$ |
| JP | | Jump Relative Short | $\text{PC} \leftarrow \text{PC} + r (r \text{ is } -31 \text{ to } +32, \text{ not } 1)$ |
| JSRL | Addr. | Jump Subroutine Long | $[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC} \leftarrow \text{ii}$ |
| JSR | Addr. | Jump Subroutine | $[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC}11..PC0 \leftarrow \text{ii}$ |
| JID | Disp. | Jump Indirect | $\text{PL} \leftarrow \text{ROM}(\text{PU}, A)$ |
| RET | Addr. | Return from Subroutine | $\text{SP}+2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1]$ |
| RETSK | Addr. | Return and Skip | $\text{SP}+2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1]$, Skip next Instr. |
| RETI | | Return from Interrupt | $\text{SP}+2, \text{PL} \leftarrow [\text{SP}], \text{PU} \leftarrow [\text{SP}-1], \text{GIE} \leftarrow 1$ |
| INTR | | Generate an Interrupt | $[\text{SP}] \leftarrow \text{PL}, [\text{SP}-1] \leftarrow \text{PU}, \text{SP}-2, \text{PC} \leftarrow 0\text{FF}$ |
| NOP | | No Operation | $\text{PC} \leftarrow \text{PC} + 1$ |

Instruction Set (Continued)

- Most instructions are single byte (with immediate addressing mode instructions requiring 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.

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

Arithmetic and Logic Instructions (Bytes/Cycles)

| Instr | [B] | Direct | Immediate |
|-------|-----|--------|-----------|
| ADD | 1/1 | 3/4 | |
| ADC | 1/1 | 3/4 | 2/2 |
| SUBC | 1/1 | 3/4 | 2/2 |
| AND | 1/1 | 3/4 | 2/2 |
| OR | 1/1 | 3/4 | 2/2 |
| XOR | 1/1 | 3/4 | 2/2 |
| IFEQ | 1/1 | 3/4 | 2/2 |
| IFNE | 1/1 | 3/4 | 2/2 |
| IFGT | 1/1 | 3/4 | 2/2 |
| IFBNE | 1/1 | | 2/2 |
| DRSZ | 1/1 | 1/3 | |
| SBIT | 1/1 | 3/4 | |
| RBIT | 1/1 | 3/4 | |
| IFBIT | 1/1 | 3/4 | |

Instructions Using A and C (Bytes/Cycles)

| Instr | Bytes/Cycles |
|-------|--------------|
| CLRA | 1/1 |
| INCA | 1/1 |
| DECA | 1/1 |
| LAID | 1/3 |
| DCOR | 1/1 |
| RRCA | 1/1 |
| SWAPA | 1/1 |
| SC | 1/1 |
| RC | 1/1 |
| IFC | 1/1 |
| IFNC | 1/1 |

Transfer of Control Instructions (Bytes/Cycles)

| Instr | Bytes/Cycles |
|-------|--------------|
| JMPL | 3/4 |
| JMP | 2/3 |
| JP | 1/3 |
| JSRL | 3/5 |
| JSR | 2/5 |
| JID | 1/3 |
| RET | 1/5 |
| RETSK | 1/5 |
| RETI | 1/5 |
| INTR | 1/7 |
| NOP | 1/1 |

Memory Transfer Instructions (Bytes/Cycles)

| Instr | Register Indirect | | Direct | Immed. | Register Indirect Auto Incr and Decr | |
|-------------------|-------------------|-----|--------|------------------|--------------------------------------|----------|
| | [B] | [X] | | | [B+, B-] | [X+, X-] |
| X A, ^a | | | 2/3 | | | |
| LD A,* | 1/1 | | 2/3 | | 1/2 | |
| LD B,Imm | 1/1 | 1/3 | | 2/2 | 1/2 | 1/3 |
| LD B,Imm | | 1/3 | | 1/1 ^b | | 1/3 |
| LD Mem,Imm | | | 3/3 | 2/3 ^c | | |
| LD Reg,Imm | 2/2 | | 2/3 | | 2/2 | |

a. Memory location addressed by B or X directly

b. IF B < 16

c. IF B > 15

| UPPER NIBBLE BITS 7-4 | | | | | | | | | | | LOWER NIBBLE BITS 3-0 | | | | | | | | | | | | | | | | | | | | |
|-----------------------|-------|------------|----------|------------|------------|-------------|-------------|--------------|-------------|----------|-----------------------|---------------|---------------|-------|--------|-------|-------|------------|----------|------------|------------|-------------|-------------|--------------|-------------|----------|----------|---------------|---------------|-------|--------|
| F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| JP-15 | JP-31 | LD 0F0, #i | DRSZ 0F0 | RRCA | RC | ADCA, #i | ADCA, (B) | IFBIT 0, (B) | * | LD B, 0F | IFBNE 0 | JSR 0000-00FF | JMP 0000-00FF | JP+17 | INTR | JP-15 | JP-31 | LD 0F0, #i | DRSZ 0F0 | RRCA | RC | ADCA, #i | ADCA, (B) | IFBIT 0, (B) | * | LD B, 0F | IFBNE 0 | JSR 0000-00FF | JMP 0000-00FF | JP+17 | INTR |
| JP-14 | JP-30 | LD 0F1, #1 | DRSZ 0F1 | * | SC | SUBCA, #i | SUBC A, (B) | IFBIT 1, (B) | * | LD B, 0E | IFBNE 1 | JSR 0100-01FF | JMP 0100-01FF | JP+18 | JP+2 | JP-14 | JP-30 | LD 0F1, #1 | DRSZ 0F1 | * | SC | SUBCA, #i | SUBC A, (B) | IFBIT 1, (B) | * | LD B, 0E | IFBNE 1 | JSR 0100-01FF | JMP 0100-01FF | JP+18 | JP+2 |
| JP-13 | JP-29 | LD 0F2, #i | DRSZ 0F2 | XA (X+) | XA, (X+) | IFEQA, #i | IFEQ, #i | IFBIT A, (B) | * | LD B, 0D | IFBNE 2 | JSR 0200-02FF | JMP 0200-02FF | JP+19 | U,JP+3 | JP-13 | JP-29 | LD 0F2, #i | DRSZ 0F2 | XA (X+) | XA, (X+) | IFEQA, #i | IFEQ, #i | IFBIT A, (B) | * | LD B, 0D | IFBNE 2 | JSR 0200-02FF | JMP 0200-02FF | JP+19 | U,JP+3 |
| JP-12 | JP-28 | LD 0F3, #i | DRSZ 0F3 | XA, (X-) | XA, (B-) | IFGT A, #i | IFGT A, (B) | IFBIT 3, (B) | * | LDB, 0C | IFBNE 3 | JSR 0300-03FF | JMP 0300-03FF | JP+20 | JP+4 | JP-12 | JP-28 | LD 0F3, #i | DRSZ 0F3 | XA, (X-) | XA, (B-) | IFGT A, #i | IFGT A, (B) | IFBIT 3, (B) | * | LDB, 0C | IFBNE 3 | JSR 0300-03FF | JMP 0300-03FF | JP+20 | JP+4 |
| JP-11 | JP-27 | LD 0F4, #i | DRSZ 0F4 | * | LAID | ADD A, #i | ADD A, (B) | IFBIT 4, (B) | CLRA | LD B, 0B | IFBNE 4 | JSR 0400-04FF | JMP 0400-04FF | JP+21 | JP+5 | JP-11 | JP-27 | LD 0F4, #i | DRSZ 0F4 | * | LAID | ADD A, #i | ADD A, (B) | IFBIT 4, (B) | CLRA | LD B, 0B | IFBNE 4 | JSR 0400-04FF | JMP 0400-04FF | JP+21 | JP+5 |
| JP-10 | JP-26 | LD 0F5, #i | DRSZ 0F5 | * | JID | AND A, #i | AND A, (B) | IFBIT 5, (B) | SWAPA | LD B, 0A | IFBNE 5 | JSR 0500-05FF | JMP 0500-05FF | JP+22 | JP+6 | JP-10 | JP-26 | LD 0F5, #i | DRSZ 0F5 | * | JID | AND A, #i | AND A, (B) | IFBIT 5, (B) | SWAPA | LD B, 0A | IFBNE 5 | JSR 0500-05FF | JMP 0500-05FF | JP+22 | JP+6 |
| JP-9 | JP-25 | LD 0F6, #i | DRSZ 0F6 | XA, (X) | XA, (B) | XOR A, #i | XOR A, (B) | IFBIT 6, (B) | DCORA | LD B, 9 | IFBNE 6 | JSR 0600-06FF | JMP 0600-06FF | JP+23 | JP+7 | JP-9 | JP-25 | LD 0F6, #i | DRSZ 0F6 | XA, (X) | XA, (B) | XOR A, #i | XOR A, (B) | IFBIT 6, (B) | DCORA | LD B, 9 | IFBNE 6 | JSR 0600-06FF | JMP 0600-06FF | JP+23 | JP+7 |
| JP-8 | JP-24 | LD 0F7, #i | DRSZ 0F7 | * | * | OR A, #i | OR A, (B) | IFBIT 7, (B) | * | LD B, 8 | IFBNE 7 | JSR 0700-07FF | JMP 0700-07FF | JP+24 | JP+8 | JP-8 | JP-24 | LD 0F7, #i | DRSZ 0F7 | * | * | OR A, #i | OR A, (B) | IFBIT 7, (B) | * | LD B, 8 | IFBNE 7 | JSR 0700-07FF | JMP 0700-07FF | JP+24 | JP+8 |
| JP-7 | JP-23 | LD 0F8, #i | DRSZ 0F8 | NOP | * | LD A, #i | LD A, (B) | IFBIT 8, (B) | SBIT 0, (B) | LD B, 7 | IFBNE 8 | JSR 0800-08FF | JMP 0800-08FF | JP+25 | JP+9 | JP-7 | JP-23 | LD 0F8, #i | DRSZ 0F8 | NOP | * | LD A, #i | LD A, (B) | IFBIT 8, (B) | SBIT 0, (B) | LD B, 7 | IFBNE 8 | JSR 0800-08FF | JMP 0800-08FF | JP+25 | JP+9 |
| JP-6 | JP-22 | LD 0F9, #i | DRSZ 0F9 | * | * | * | IFNC | IFBIT 1, (B) | IFNC | LD B, 6 | IFBNE 9 | JSR 0900-09FF | JMP 0900-09FF | JP+26 | JP+10 | JP-6 | JP-22 | LD 0F9, #i | DRSZ 0F9 | * | * | * | IFNC | IFBIT 1, (B) | IFNC | LD B, 6 | IFBNE 9 | JSR 0900-09FF | JMP 0900-09FF | JP+26 | JP+10 |
| JP-5 | JP-21 | LD 0FA, #i | DRSZ 0FA | LDA, X(+) | LD A, (B+) | LD (B+), #i | INCA | IFBIT 2, (B) | IFNC | LD B, 5 | IFBNE 0A | JSR 0A00-0AFF | JMP 0A00-0AFF | JP+27 | JP+11 | JP-5 | JP-21 | LD 0FA, #i | DRSZ 0FA | LDA, X(+) | LD A, (B+) | LD (B+), #i | INCA | IFBIT 2, (B) | IFNC | LD B, 5 | IFBNE 0A | JSR 0A00-0AFF | JMP 0A00-0AFF | JP+27 | JP+11 |
| JP-4 | JP-20 | LD 0FB, #i | DRSZ 0FB | LDA, X(-) | LD A, (B-) | LD (B-), #i | DECA | IFBIT 3, (B) | IFNC | LD B, 4 | IFBNE 0B | JSR 0B00-0BFF | JMP 0B00-0BFF | JP+28 | JP+12 | JP-4 | JP-20 | LD 0FB, #i | DRSZ 0FB | LDA, X(-) | LD A, (B-) | LD (B-), #i | DECA | IFBIT 3, (B) | IFNC | LD B, 4 | IFBNE 0B | JSR 0B00-0BFF | JMP 0B00-0BFF | JP+28 | JP+12 |
| JP-3 | JP-19 | LD 0FC, #i | DRSZ 0FC | LD Mld, #i | JMPL | X A, Mld | * | IFBIT 4, (B) | IFNC | LD B, 3 | IFBNE 0C | JSR 0C00-0CFF | JMP 0C00-0CFF | JP+29 | JP+13 | JP-3 | JP-19 | LD 0FC, #i | DRSZ 0FC | LD Mld, #i | JMPL | X A, Mld | * | IFBIT 4, (B) | IFNC | LD B, 3 | IFBNE 0C | JSR 0C00-0CFF | JMP 0C00-0CFF | JP+29 | JP+13 |
| JP-2 | JP-18 | LD 0D, #i | DRSZ 0D | DIR | JSRL | LD A, Mld | RETSK | IFBIT 5, (B) | IFNC | LD B, 2 | IFBNE 0D | JSR 0D00-0DFF | JMP 0D00-0DFF | JP+30 | JP+14 | JP-2 | JP-18 | LD 0D, #i | DRSZ 0D | DIR | JSRL | LD A, Mld | RETSK | IFBIT 5, (B) | IFNC | LD B, 2 | IFBNE 0D | JSR 0D00-0DFF | JMP 0D00-0DFF | JP+30 | JP+14 |
| JP-1 | JP-17 | LD 0FE, #i | DRSZ 0FE | LD A, (X) | LD A, (B) | LD B, #i | RET | IFBIT 6, (B) | IFNC | LD B, 1 | IFBNE 0E | JSR 0E00-0EFF | JMP 0E00-0EFF | JP+31 | JP+15 | JP-1 | JP-17 | LD 0FE, #i | DRSZ 0FE | LD A, (X) | LD A, (B) | LD B, #i | RET | IFBIT 6, (B) | IFNC | LD B, 1 | IFBNE 0E | JSR 0E00-0EFF | JMP 0E00-0EFF | JP+31 | JP+15 |
| JP-0 | JP-16 | LD 0FF, #i | DRSZ 0FF | * | * | * | RETI | IFBIT 7, (B) | IFNC | LD B, 0 | IFBNE 0F | JSR 0F00-0FFF | JMP 0F00-0FFF | JP+32 | JP+16 | JP-0 | JP-16 | LD 0FF, #i | DRSZ 0FF | * | * | * | RETI | IFBIT 7, (B) | IFNC | LD B, 0 | IFBNE 0F | JSR 0F00-0FFF | JMP 0F00-0FFF | JP+32 | JP+16 |

Option List

The mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

- = 1 Crystal (CKI/10) CKO for crystal configuration
- = 2 NA
- = 3 R/C (CKI/10) CKO available as G7 input

OPTION 2: BONDING

- = 1 NA
- = 2 NA
- = 3 20 pin DIP package
- = 4 20 pin SO package
- = 5 NA

The following option information is to be sent to National along with the EPROM.

Option Data

Option 1 Value__is: CKI Input

Option 2 Value__is: COP Bonding

Development Support

SUMMARY

- iceMASTER™: IM-COP8/400—Full feature in-circuit emulation for all COP8 products. A full set of COP8 Basic and Feature Family device and package specific probes are available.
- COP8 Debug Module: Moderate cost in-circuit emulation and development programming unit.
- COP8 Evaluation and Programming Unit: EPU-COP8780—low cost in-circuit simulation and development programming unit.
- Assembler: COP8-DEV-IBMA. A DOS installable cross development Assembler, Linker, Librarian and Utility Software Development Tool Kit.
- C Compiler: COP8C. A DOS installable cross development Software Tool Kit.
- OTP/EPROM Programmer Support: Covering needs from engineering prototype, pilot production to full production environments.

Development Support (Continued)

IceMASTER (IM) IN-CIRCUIT EMULATION

The iceMASTER IM-COP8/400 is a full feature, PC based, in-circuit emulation tool developed and marketed by Meta-Link Corporation to support the whole COP8 family of products. National is a resale vendor for these products.

See *Figure 12* for configuration.

The iceMASTER IM-COP8/400 with its device specific COP8 Probe provides a rich feature set for developing, testing and maintaining product:

- Real-time in-circuit emulation; full 2.4V–5.5V operation range, full DC-10 MHz clock. Chip options are programmable or jumper selectable.
- Direct connection to application board by package compatible socket or surface assembly.
- Full 32 kbyte of loadable programming space that overlays (replaces) the on-chip ROM or EPROM. On-chip RAM and I/O blocks are used directly or recreated on the probe as necessary.
- Full 4k frame synchronous trace memory. Address, instruction, and 8 unspecified, circuit connectable trace lines. Display can be HLL source (e.g., C source), assembly or mixed.
- A full 64k hardware configurable break, trace on, trace off control, and pass count increment events.
- Tool set integrated interactive symbolic debugger—supports both assembler (COFF) and C Compiler (.COD) linked object formats.
- Real time performance profiling analysis; selectable bucket definition.
- Watch windows, content updated automatically at each execution break.
- Instruction by instruction memory/register changes displayed on source window when in single step operation.
- Single base unit and debugger software reconfigurable to support the entire COP8 family; only the probe personality needs to change. Debugger software is processor customized, and reconfigured from a master model file.
- Processor specific symbolic display of registers and bit level assignments, configured from master model file.
- Halt/Idle mode notification.
- On-Line HELP customized to specific processor using master model file.
- Includes a copy of COP8-DEV-IBMA assembler and linker SDK.

IM Order Information

| Base Unit | |
|------------------------|--|
| IM-COP8/400-1 | iceMASTER Base Unit, 110V Power Supply |
| IM-COP8/400-2 | iceMASTER Base Unit, 220V Power Supply |
| iceMASTER Probe | |
| MHW-880C-20DWPC | 20 DIP |
| Adapter for SO Package | |
| MHW-SOIC20 | 20 SO |

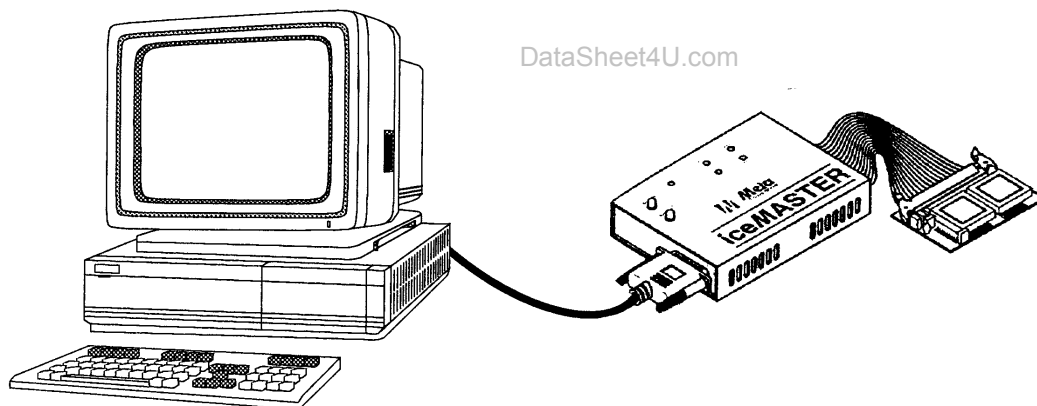


FIGURE 12. COP8 iceMASTER Environment

TL/DD/12060-15

Development Support (Continued)

iceMASTER DEBUG MODULE (DM)

The iceMASTER Debug Module is a PC based, combination in-circuit emulation tool and COP8 based OTP/EPROM programming tool developed and marketed by MetaLink Corporation to support the whole COP8 family of products. National is a resale vendor for these products.

See *Figure 13* for configuration.

The iceMASTER Debug Module is a moderate cost development tool. It has the capability of in-circuit emulation for a specific COP8 microcontroller and in addition serves as a programming tool for COP8 OTP and EPROM product families. Summary of features is as follows:

- Real-time in-circuit emulation; full operating voltage range operation, full DC-10 MHz clock.
- All processor I/O pins can be cabled to an application development board with package compatible cable to socket and surface mount assembly.
- Full 32 kbyte of loadable programming space that overlays (replaces) the on-chip ROM or EPROM. On-chip RAM and I/O blocks are used directly or recreated as necessary.
- 100 frames of synchronous trace memory. The display can be HLL source (C source), assembly or mixed. The most recent history prior to a break is available in the trace memory.
- Configured break points; uses INTR instruction which is modestly intrusive.
- Software—only supported features are selectable.
- Tool set integrated interactive symbolic debugger—supports both assembler (COFF) and C Compiler (.COD) SDK linked object formats.
- Instruction by instruction memory/register changes displayed when in single step operation.

- Debugger software is processor customized, and reconfigured from a master model file.
- Processor specific symbolic display of registers and bit level assignments, configured from master model file.
- Halt/Idle mode notification.
- Programming menu supports full product line of programmable OTP and EPROM COP8 products. Program data is taken directly from the overlay RAM.
- Programming of 44 PLCC and 68 PLCC parts requires external programming adapters.
- Includes wallmount power supply.
- On-board V_{PP} generator from 5V input or connection to external supply supported. Requires V_{PP} level adjustment per the family programming specification (correct level is provided on an on-screen pop-down display).
- On-line HELP customized to specific processor using master model file.
- Includes a copy of COP8-DEV-IBMA assembler and linker SDK.

DM Order Information

| Debug Module Unit | |
|------------------------|--------|
| COP8/DM/880C | |
| Cable Adapters | |
| DM-COP8/20D | 20 DIP |
| Adapter for SO Package | |
| MHW-SOIC20 | 20 SO |

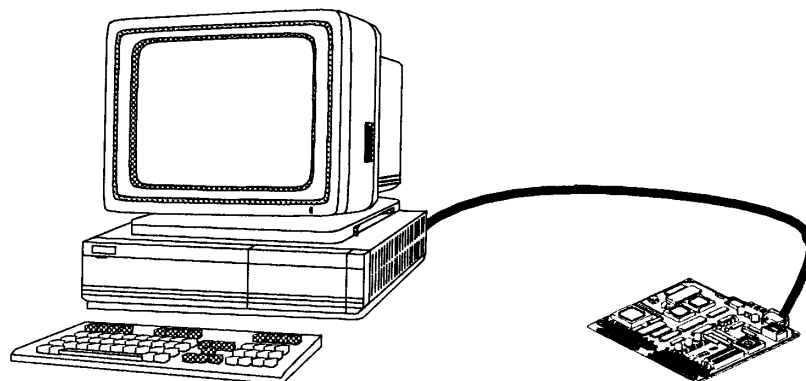


FIGURE 13. COP8-DM Environment

TL/DD/12060-23

Development Support (Continued)

COP8 ASSEMBLER/LINKER SOFTWARE DEVELOPMENT TOOL KIT

National Semiconductor offers a relocatable COP8 macro cross assembler, linker, librarian and utility software development tool kit. Features are summarized as follows:

- Basic and Feature Family instruction set by "device" type.
- Nested macro capability.
- Extensive set of assembler directives.
- Supported on PC/DOS platform.
- Generates National standard COFF output files.
- Integrated Linker and Librarian.
- Integrated utilities to generate ROM code file outputs.
- DUMPCOFF utility.

This product is integrated as a part of MetaLink tools as a development kit, fully supported by the MetaLink debugger. It may be ordered separately or it is bundled with the MetaLink products at no additional cost.

Order Information

| Assembler SDK: | |
|----------------|--|
| COP8-DEV-IBMA | Assembler SDK on installable 3.5" PC®/DOS Floppy Disk Drive format. Periodic upgrades and most recent version is available on National's BBS and Internet. |

COP8 C COMPILER

A C Compiler is developed and marketed by Byte Craft Limited. The COP8C compiler is a fully integrated development tool specifically designed to support the compact embedded configuration of the COP8 family of products.

Features are summarized as follows:

- ANSI C with some restrictions and extensions that optimize development for the COP8 embedded application.
- BITS data type extension. Register declaration #pragma with direct bit level definitions.
- C language support for interrupt routines.
- Expert system, rule based code generation and optimization.
- Performs consistency checks against the architectural definitions of the target COP8 device.
- Generates program memory code.
- Supports linking of compiled object or COP8 assembled object formats.
- Global optimization of linked code.
- Symbolic debug load format fully source level supported by the MetaLink debugger.

SINGLE CHIP OTP/EMULATOR SUPPORT

The COP8 family is supported by single chip OTP emulators. For detailed information refer to the emulator specific datasheet and the emulator selection table below:

Approved List

| Manufacturer | North America | Europe | Asia |
|-----------------|---|---|---|
| BP Microsystems | (800) 225-2102 (713) 688-4600 Fax: (713) 688-0920 | +49-8152-4183 +49-8856-932616 | +852-234-16611 +852-2710-8121 |
| Data I/O | (800) 426-1045 (206) 881-6444 Fax: (206) 882-1043 | +44-0734-440011 | Call North America |
| HI-LO | (510) 623-8860 | Call Asia | +886-2-764-0215 Fax: +886-2-756-6403 |
| ICE Technology | (800) 624-8949 (919) 430-7915 | +44-1226-767404 Fax: 0-1226-370-434 | |
| MetaLink | (800) 638-2423 (602) 926-0797 Fax: (602) 693-0681 | +49-80 9156 96-0 Fax: +49-80 9123 86 | +852-737-1800 |
| Systems General | (408) 263-6667 | +41-1-9450300 | +886-2-917-3005 Fax: +886-2-911-1283 |
| Needhams | (916) 924-8037 Fax: (916) 924-8065 | | |

Development Support (Continued)**OTP Emulator Ordering Information**

| Device Number | Clock Option | Package | Emulates |
|---------------|--------------|---------|----------------------|
| COP8782CN | Programmable | 20 N | COP912C, COP912CH |
| COP8782CWM | Programmable | 20 SO | COP912C, COP912CH |

INDUSTRY WIDE OTP/EPROM PROGRAMMING SUPPORT

Programming support, in addition to the MetaLink development tools, is provided by a full range of independent approved vendors to meet the needs from the engineering laboratory to full production.

AVAILABLE LITERATURE

For more information, please see the COP8 Basic Family User's Manual, Literature Number 620895, COP8 Feature Family User's Manual, Literature Number 620897 and National's Family of 8-bit Microcontrollers COP8 Selection Guide, Literature Number 630009.

DIAL-A-HELPER SERVICE

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Information System that may be accessed as a Bulletin Board System (BBS) via data modem, as an FTP site on the Internet via standard FTP client application or as an FTP site on the Internet using a standard Internet browser such as Netscape or Mosaic.

The Dial-A-Helper system provides access to an automated information storage and retrieval system. The system capabilities include a MESSAGE SECTION (electronic mail, when accessed as a BBS) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found.

DIAL-A-HELPER BBS via a Standard Modem

Modem: CANADA/U.S.: (800) NSC-MICRO
(800) 672-6427

EUROPE: (+ 49) 0-8141-351332

Baud: 14.4k

Set-up: Length: 8-Bit
Parity: None
Stop Bit: 1

Operation: 24 Hours, 7 Days

DIAL-A-HELPER via FTP

ftp nscmicro.nsc.com
user: anonymous
password: username@yourhost.site.domain

DIAL-A-HELPER via a WorldWide Web Browser

ftp://nscmicro.nsc.com

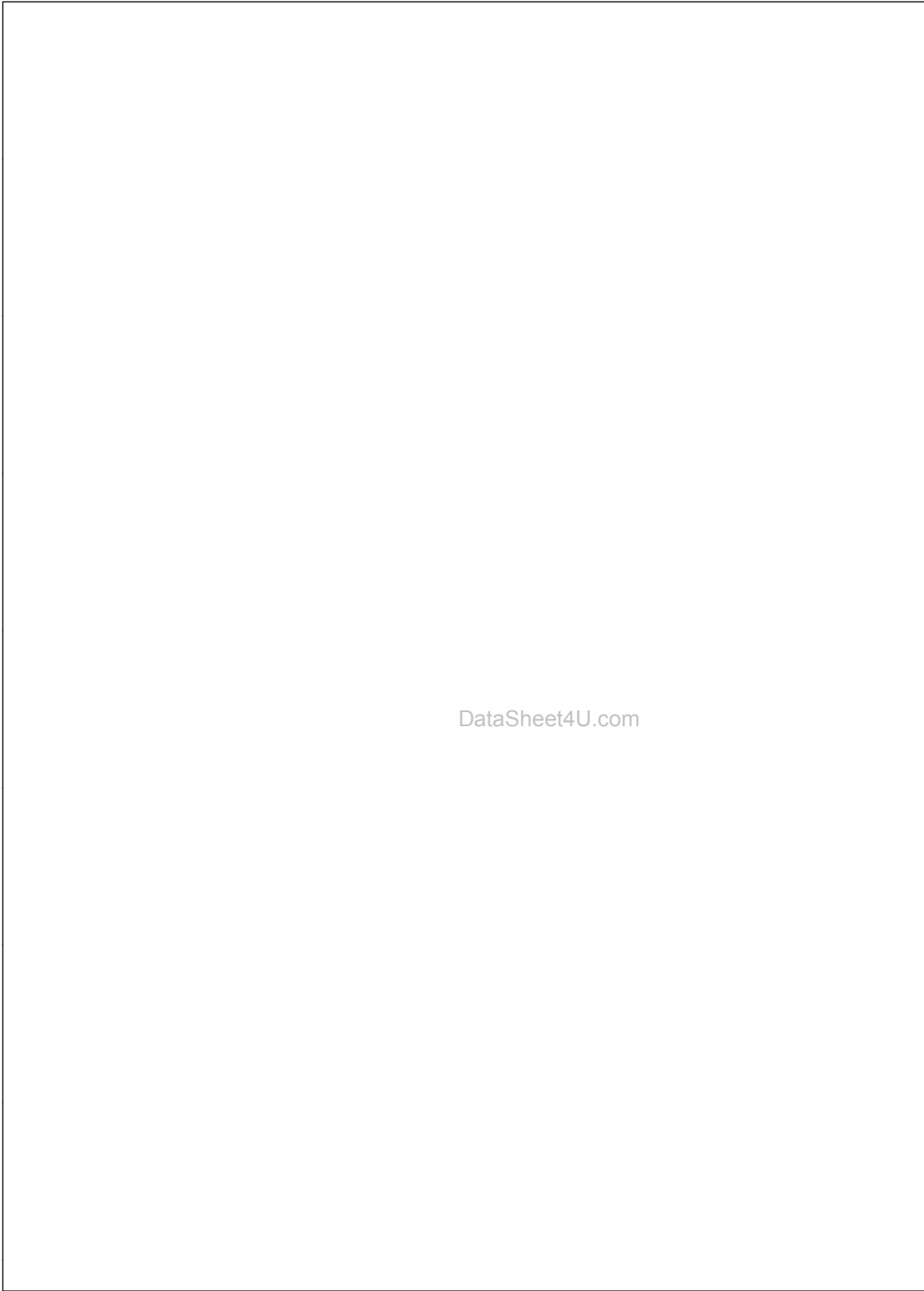
National Semiconductor on the WorldWide Web

See us on the WorldWide Web at: <http://www.national.com>

CUSTOMER RESPONSE CENTER

Complete product information and technical support is available from National's customer response centers.

| | | |
|--------------|----------------|------------------------|
| CANADA/U.S.: | Tel: | (800) 272-9959 |
| | email: | support@tevm2.nsc.com |
| EUROPE: | email: | europa.support@nsc.com |
| | Deutsch Tel: | + 49 (0) 180-530 85 85 |
| | English Tel: | + 49 (0) 180-532 78 32 |
| | Français Tel: | + 49 (0) 180-532 93 58 |
| | Italiano Tel: | + 49 (0) 180-534 16 80 |
| JAPAN: | Tel: | + 81-043-299-2309 |
| S.E. ASIA: | Beijing Tel: | (+ 86) 10-6856-8601 |
| | Shanghai Tel: | (+ 86) 21-6415-4092 |
| | Hong Kong Tel: | (+ 852) 2737-1600 |
| | Korea Tel: | (+ 82) 2-3771-6909 |
| | Malaysia Tel: | (+ 60-4) 644-9061 |
| | Singapore Tel: | (+ 65) 255-2226 |
| | Taiwan Tel: | + 886-2-521-3288 |
| AUSTRALIA: | Tel: | (+ 61) 3-9558-9999 |
| INDIA: | Tel: | (+ 91) 80-559-9467 |

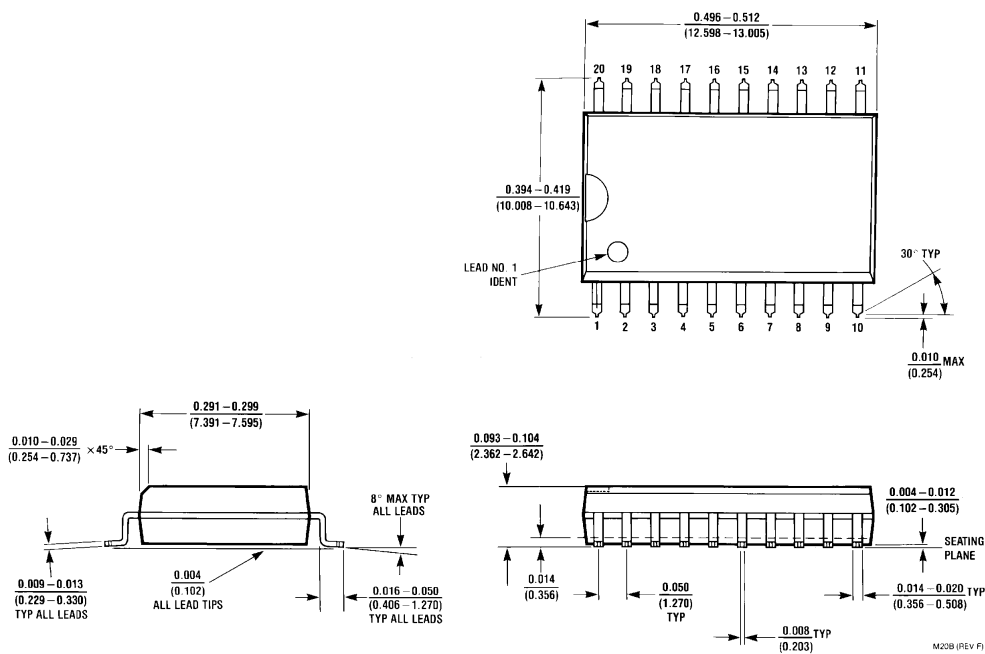


et4U.com

DataShee

DataSheet4U.com

Physical Dimensions inches (millimeters) unless otherwise noted



20-Lead Molded Small Outline Package (M)
Order Number COP912C-XXX/WM, COP912CH-XXX/WM
NS Package Number M20B

DataSheet4U.com

