National Semiconductor

PRELIMINARY
September 1996

COP684BC/COP884BC
8-Bit Microcontrollers with CAN Interface

General Description

The COP684BC/COP884BC are members of the COP8™ feature family of microcontrollers which uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. Each device is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Key Features

- CAN Interface
- On chip reset
- One 16-bit timer, with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- High speed, constant resolution 8-bit PWM/frequency monitor timer with 2 output pins
- 2048 bytes on-board ROM
- 64 bytes on-board RAM

Additional Peripheral Features

- Idle Timer
- Multi-Input Wake Up (MIWU) with optional interrupts (7)
- Two analog comparators
- 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: 28 SO with 18 I/O pins

CPU/Instruction Set Features

- 1 μs instruction cycle time
- Eleven multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Timer T1 (with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — PWM Timer
  - — CAN Interface (with 3 interrupts)
- 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)

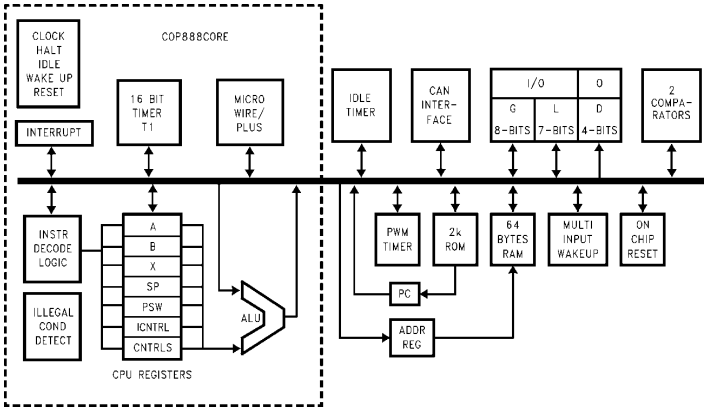
Fully Static CMOS

- Two power saving modes: HALT and IDLE
- Low current drain (typically <1 μA)
- Single supply operation: 4.5V–5.5V
- Temperature ranges: −40°C to +85°C, −55°C to +125°C

Development Support

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

Block Diagram



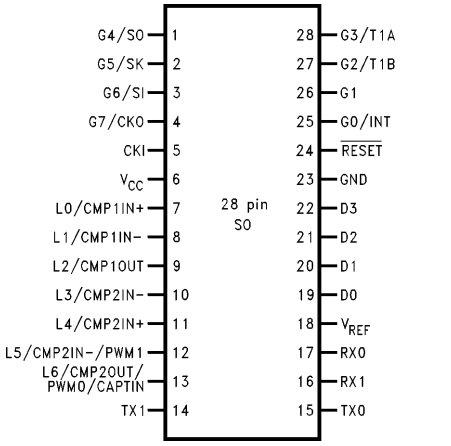
TL/DD/12067–1

FIGURE 1

General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, a 16-bit timer/counter supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), a CAN interface, two comparators, 8-bit, high speed, constant resolution PWM/frequency monitor timer, and two power savings modes (HALT and IDLE), both with a multi-sourced wake up/ interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The device operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The device has low EMI emissions. Low radiated emissions are achieved by gradual turn-on output drivers and internal I_{CC} smoothing filters on the chip logic and crystal oscillator.

Connection Diagram



TL/DD/12067–2

Top View

Order Number COP884BC-xxx/M or COP684BC-xxx/M
See NS Package Number M28B

FIGURE 2

Pinouts for 28-Pin SO Package

Port Pin	Type	Alt. Function	28-Pin SO
G0	I/O	INTR	25
G1	I/O		26
G2	I/O	T1B	27
G3	I/O	T1A	28
G4	I/O	SO	1
G5	I/O	SK	2
G6	I	SI	3
G7	I	CKO	4
L0	I/O	CMP1IN+/MIWU	7
L1	I/O	CMP1IN−/MIWU	8
L2	I/O	CMP10UT/MIWU	9
L3	I/O	CMP2IN−/MIWU	10
L4	I/O	CMP2IN+/MIWU	11
L5	I/O	CMP2IN−/PWM1/MIWU	12
L6	I/O	CMP2OUT/PWM0/ CAPTIN/MIWU	13
D0	O		19
D1	O		20
D2	O		21
D3	O		22
CAN V_{REF}			18
CAN Tx0	O		15
CAN Tx1	O		14
CAN Rx0	I	MIWU (Note A)	17
CAN Rx1	I	MIWU	16
V_{CC}			6
GND			23
CKI	I		5
RESET	I		24

Note A: The MIWU function for the CAN interface is internal (see CAN interface block diagram)

Absolute Maximum Ratings (Note)

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

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	90 mA
Total Current out of GND Pin (Sink)	100 mA
Storage Temperature Range	$-65°C$ to $+150°C$

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 COP884BC: $-40°C \le T_A \le +85°C$

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current					
CKI = 10 MHz (Note 2)	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			15	mA
HALT Current (Notes 3, 4)	$V_{CC} = 5.5V$, CKI = 0 MHz				
	Power-On Reset Enabled		<300	480	μA
	Power-On Reset Disabled		<250	380	μA
IDLE Current (Note 4)					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			5.5	mA
Input Levels (V_{IH}, V_{IL})					
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	μA
Input Pull-up Current	$V_{CC} = 5.5V$, $V_{IN} = 0V$	−40		−250	μA
G and L Port Input Hysteresis	(Notes 6, 7)		0.05 V_{CC}		V
Output Current Levels D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−0.4			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	10			mA
Comparator Output (L2, L6)					
Source (Push-Pull)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−1.6			mA
Sink (Push-Pull)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
CAN Transmitter Outputs					
Source (Tx1)	$V_{CC} = 4.5V$, $V_{OH} = V_{CC} - 0.1V$	−1.5			mA
	$V_{CC} = 4.5V$, $V_{OH} = V_{CC} - 0.6V$	−10			mA
Sink (Tx0)	$V_{CC} = 4.5V$, $V_{OL} = 0.1V$	1.5			mA
	$V_{CC} = 4.5V$, $V_{OL} = 0.6V$	10			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V$, $V_{OH} = 2.7V$	−10		−110	μA
Source (Push-Pull)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−0.4			mA
Sink (Push-Pull)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$			±2.0	μA

DC Electrical Characteristics COP884BC: $-40°C \leq T_A \leq +85°C$ (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
Tx0 (Sink) (Note 7)				30	mA
Tx1 (Source) (Note 7)				30	mA
All Other				3	mA
Maximum Input Current without Latchup (Notes 5, 7)	Room Temp			±100	mA
RAM Retention Voltage, V_r (Note 6)	500 ns Rise and Fall Time	2.0			V
Input Capacitance	(Note 7)			7	pF
Load Capacitance on D2				1000	pF

Note 1: Maximum 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 V_{CC} or GND, and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the Crystal configurations. Halt test conditions: All inputs tied to V_{CC}; L, and G port I/Os configured as outputs and programmed low; D outputs programmed low. 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 4: HALT and IDLE current specifications assume CAN block and comparators are disabled.

Absolute Maximum Ratings (Note)

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

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	$-65°C$ to $+150°C$

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 COP684BC: $-55°C \le T_A \le +125°C$

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current					
CKI = 10 MHz (Note 2)	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			15	mA
HALT Current (Notes 3, 4)	$V_{CC} = 5.5V$, CKI = 0 MHz				
	Power-On Reset Enabled		<300	480	μA
	Power-On Reset Disabled		<250	380	μA
IDLE Current (Note 4)					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			5.5	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	$V_{CC} = 5.5V$			±5	μA
Input Pull-up Current	$V_{CC} = 5.5V$, $V_{IN} = 0V$	−35		−250	μA
G and L Port Input Hysteresis	(Note 6)		0.05 V_{CC}		V
Output Current Levels D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−0.4			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	9.0			mA
Comparator Output (L2, L6)					
Source (Push-Pull)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−1.6			mA
Sink (Push-Pull)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
CAN Transmitter Outputs					
Source (Tx1)	$V_{CC} = 4.5V$, $V_{OH} = V_{CC} - 0.1V$	−1.5			mA
	$V_{CC} = 4.5V$, $V_{OH} = V_{CC} - 0.6V$	−10			mA
Sink (Tx0)	$V_{CC} = 4.5V$, $V_{OL} = 0.1V$	1.5			mA
	$V_{CC} = 4.5V$, $V_{OL} = 0.6V$	10			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V$, $V_{OH} = 2.7V$	−9.0		−100	μA
Source (Push-Pull)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	−0.4			mA
Sink (Push-Pull)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$			±5.0	μA

DC Electrical Characteristics COP684BC: $-55°C \leq T_A \leq +125°C$ (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				12	mA
Tx0 (Sink) (Note 7)				30	mA
Tx1 (Source) (Note 7)				30	mA
All Other				2.5	mA
Maximum Input Current without Latchup (Notes 5, 7)	Room Temp			±100	mA
RAM Retention Voltage, V_r (Note 6)	500 ns Rise and Fall Time	2.0			V
Input Capacitance	(Note 7)			7	pF
Load Capacitance on D2				1000	pF

Note 5: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

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

Note 7: Parameter characterized but not tested.

AC Electrical Characteristics: COP684BC and COP884BC: $-55°C \le T_A \le +125°C$

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crytal/Resonator	$V_{CC} \ge 4.5V$	1.0		DC	μs
Inputs					
t_{SETUP}	$V_{CC} \ge 4.5V$	200			ns
t_{HOLD}	$V_{CC} \ge 4.5V$	60			ns
PWM Capture Input					
t_{SETUP}	$V_{CC} \ge 4.5V$	30			ns
t_{HOLD}	$V_{CC} \ge 4.5V$	70			ns
Output Propagation Delay (t_{PD1}, t_{PD0}) (Note 8)	$C_L = 100$ pF, $R_L = 2.2$ kΩ				
SK, SO	$V_{CC} \ge 4.5V$			0.7	μs
PWM Outputs	$V_{CC} \ge 4.5V$			75	ns
All Others	$V_{CC} \ge 4.5V$			1	μs
MICROWIRE					
Setup Time (t_{UWS}) (Note 9)		20			ns
Hold Time (t_{UWH}) (Note 9)		56			ns
Output Prop Delay (t_{UPD})				220	ns
Input Pulse Width (Note 10)					
Interrupt High Time		1			t_c
Interrupt Low Time		1			t_c
Timer 1,2 High Time		1			t_c
Timer 1,2 Low Time		1			t_c
Reset Pulse Width (Note 9)		1.0			μs
Power Supply Rise Time for Proper Operation of On-Chip RESET		50 μs		256*t_c	

Note: For device testing purposes of all AC parameters, V_{OH} will be tested at $0.5*V_{CC}$.

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

Note 9: Parameter not tested.

Note 10: t_c = Instruction Cycle Time.

On-Chip Voltage Reference: $-55°C \le T_A \le +125°C$

Parameter	Conditions	Min	Max	Units
Reference Voltage V_{REF}	$I_{OUT} < 80$ μA, $V_{CC} = 5V$	$0.5\ V_{CC} - 0.12$	$0.5\ V_{CC} + 0.12$	V
Reference Supply Current, I_{DD}	$I_{OUT} = 0$A, (No Load) $V_{CC} = 5V$ (Note A)		120	μA

Note A: Reference supply I_{DD} is supplied for information purposes only, it is not tested.

Comparator DC/AC Characteristics: $4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_A \le +125°C$

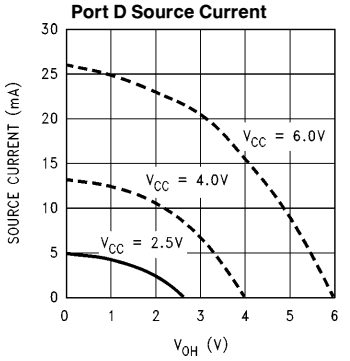
Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V < V_{IN} < V_{CC} - 1.5V$		±10	±25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Voltage Gain			300k		V/V
Outputs Sink/Source	See I/O-Port DC Specifications				
DC Supply Current (when enabled)	$V_{CC} = 6.0V$			250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs

CAN Comparator DC and AC Characteristics: $4.8V \le V_{CC} \le 5.2V$, $-40°C \le T_A \le +125°C$

Parameters	Conditions	Min	Typ	Max	Units
Differential Input Voltage				±25	mV
Input Offset Voltage	$1.5V < V_{IN} < V_{CC} - 1.5V$			±10	mV
Input Common Mode Voltage Range		1.5		$V_{CC} - 1.5$	V
Input Hysteresis		8			mV

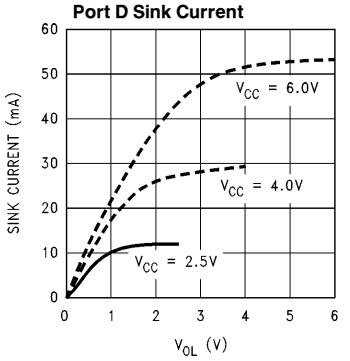
Typical Performance Characteristics $-55°C \le T_A \le +125°C$

Port D Source Current



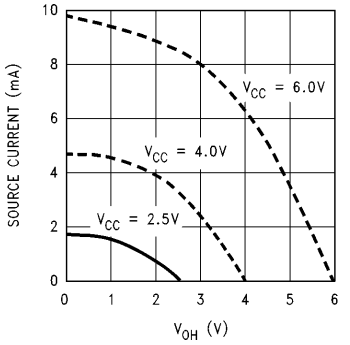
TL/DD/12067–39

Port D Sink Current



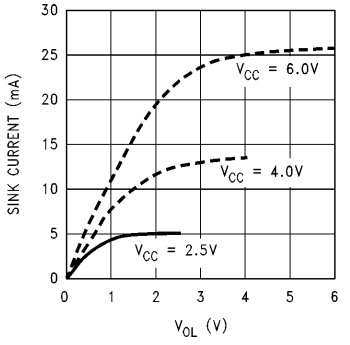
TL/DD/12067–40

Ports G/L Source Current



TL/DD/12067–41

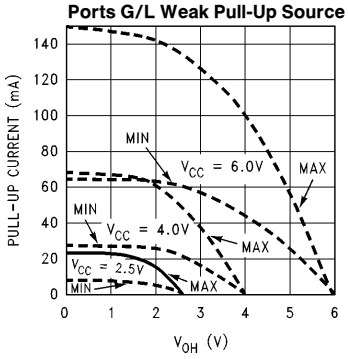
Port G/L Sink Current



TL/DD/12067–42

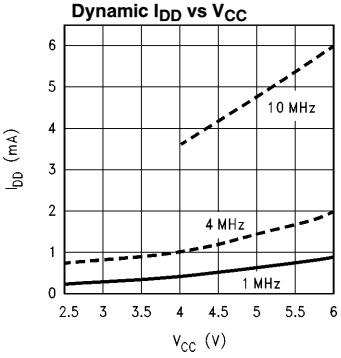
Typical Performance Characteristics $-55°C \le T_A \le +125°C$ (Continued)

Ports G/L Weak Pull-Up Source Current



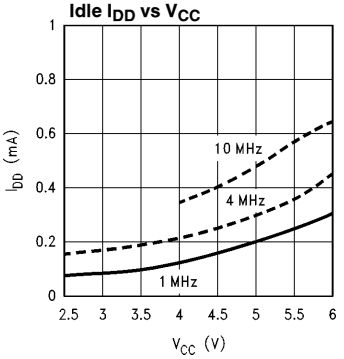
TL/DD/12067–43

Dynamic I_{DD} vs V_{CC}



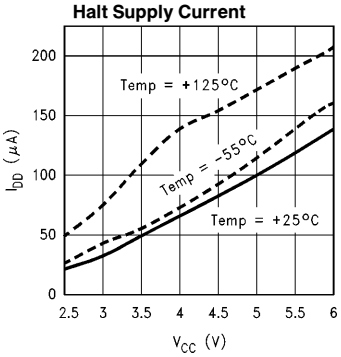
TL/DD/12067–44

Idle I_{DD} vs V_{CC}



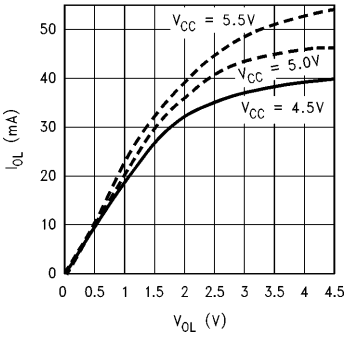
TL/DD/12067–45

Halt Supply Current



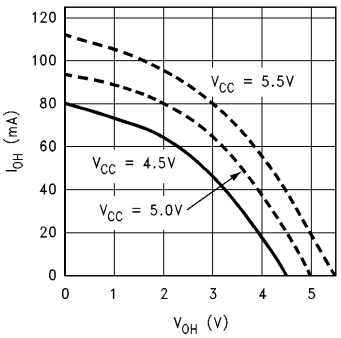
TL/DD/12067–46

CAN Tx0 Sink Current



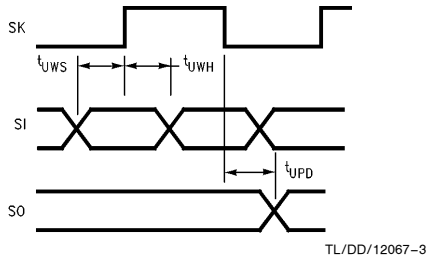
TL/DD/12067–47

CAN Tx1 Source Current



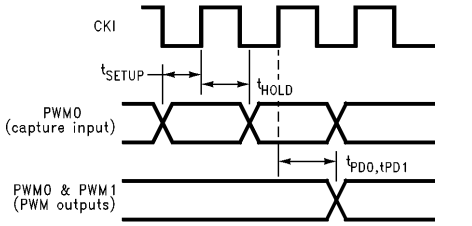
TL/DD/12067–48

AC Electrical Characteristics (Continued)



TL/DD/12067–3

FIGURE 3. MICROWIRE/PLUS Timing Diagram



TL/DD/12067–4

FIGURE 4. PWM/CAPTURE Timer Input/Output Timing Diagram

Pin Descriptions

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

CKI is the clock input. The clock can come 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 one bidirectional 8-bit I/O port (G), and one 7-bit bidirectional I/O port (L) where each individual bit may be independently configured as an input (Schmitt trigger inputs on ports G and L), output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the memory map for the various addresses associated with the I/O ports.) *Figure 5* shows the I/O port configurations for the device. 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 a 7-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wake Up (MIWU) on all seven pins.

Port L has the following alternate features:

L0 MIWU or CMP1IN+
L1 MIWU or CMP1IN−
L2 MIWU or CMP1OUT
L3 MIWU or CMP2IN−
L4 MIWU or CMP2IN+
L5 MIWU or CMP2IN− or PWM1
L6 MIWU or CMP2OUT or PWM0 or CAPTIN

Port G is an 8-bit port with 5 I/O pins (G0–G5), an input pin (G6), and one dedicated output pin (G7). Pins G0–G6 all have Schmitt Triggers on their inputs. 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.

Since G6 is an input only pin and G7 is the 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 zeroes.

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. Register	Data Register
G7		HALT
G6	Alternate SK	IDLE

CAN pins: For the on-chip CAN interface this device has five dedicated pins with the following features:

V_{REF} On-chip reference voltage with the value of $V_{CC}/2$
Rx0 CAN receive data input pin.
Rx1 CAN receive data input pin.
Tx0 CAN transmit data output pin. This pin may be put in the TRI-STATE mode with the TXEN0 bit in the CAN Bus control register.
Tx1 CAN transmit data output pin. This pin may be put in the TRI-STATE mode with the TXEN1 bit in the CAN Bus control register.

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 function:

G7 CKO Oscillator dedicated output

Port D is a 4-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs (except D2) together in order to get a higher drive.

Note: Care must be exercised with the D2 pin operation. At RESET, the external loads on this pin must ensure that the output voltages stay above 0.8 V_{CC} to prevent the chip from entering special modes. Also keep the external loading on D2 to less than 1000 pF.

Pin Descriptions (Continued)

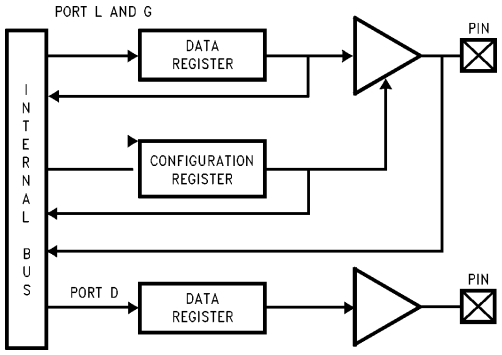


FIGURE 5. I/O Port Configurations

Functional Description

The architecture of the device utilizes a 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 five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

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

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

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

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

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 02F with reset.

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

PROGRAM MEMORY

Program memory for the device consists of 2048 bytes of ROM. These bytes may hold program instructions or constant data (data tables tor the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the device vector to program memory location 0FF Hex.

DATA MEMORY

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

The device has 64 bytes of RAM. Sixteen bytes of RAM are mapped as ''registers'' at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except A and PC) are 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.

RESET

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the data and configuration registers for Ports L and G, are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Port D is initialized high with $\overline{\text{RESET}}$. The PC, PSW, CNTRL, and ICNTRL control registers are cleared. The Multi-Input Wake Up registers WKEN, WKEDG, and WKPND are cleared. The Stack Pointer, SP, is initialized to 02F Hex.

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

Port L: TRI-STATE

Port G: TRI-STATE

Port D: HIGH

PC: CLEARED

PSW, CNTRL and ICNTRL registers: CLEARED

Accumulator and Timer 1:

RANDOM after RESET with power already applied

RANDOM after RESET at power-on

SP (Stack Pointer): Loaded with 2F Hex

CMPSL (Comparator control register): CLEARED

PWMCON (PWM control register): CLEARED

B and X Pointers:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-up

RAM:

UNAFFECTED after RESET with power already applied

RANDOM after RESET at power-up

CAN:

The CAN Interface comes out of external reset in the ''error-active'' state and waits until the user's software sets either one or both of the TXEN0, TXEN1 bits to ''1''. After that, the device will not start transmission or reception of a frame until eleven consecutive ''recessive'' (undriven) bits have been received. This is done to ensure that the output drivers are not enabled during an active message on the bus.

CSCAL, CTIM, TCNTL, TEC, REC: CLEARED

RTSTAT: CLEARED with the exception of the TBE bit which is set to 1

RID, RIDL, TID, TDLC: RANDOM

Functional Description (Continued)

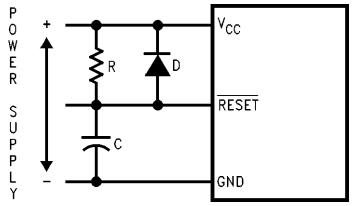
ON-CHIP POWER-ON RESET

The device is designed with an on-chip power-on reset circuit which will trigger a 256 t_c delay as V_{CC} rises above the minimum RAM retention voltage (V_r). This delay allows the oscillator to stabilize before the device exits the reset state. The contents of data registers and RAM are unknown following an on-chip power-on reset. The external reset takes priority over the on-chip reset and will deactivate the 256 t_c delay if in progress.

When using external reset, 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.

Under no circumstances should the $\overline{\text{RESET}}$ pin be allowed to float. If the on-chip power-on reset feature is being used, $\overline{\text{RESET}}$ should be connected directly to V_{CC}. Be aware of the Power Supply Rise Time requirements specified in the DC Specifications Table. These requirements must be met for the on-chip power-on reset to function properly.

The on-chip power-on reset circuit may reset the device if the operating voltage (V_{CC}) goes below V_r.



TL/DD/12067–6

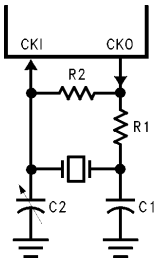
RC > 5 x Power Supply Rise Time

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. The CKI input frequency is divided by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 7 shows the Crystal diagram.



TL/DD/12067–7

FIGURE 7. Crystal Oscillator Diagram

CRYSTAL OSCILLATOR

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

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

TABLE I. 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
T1C0	Timer T1 Start/Stop control in timer Timer 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

Control Registers (Continued)

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 bit 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

Timers

The device contains a very versatile set of timers (T0, T1, and an 8-bit PWM timer). All timers and associated autoreload/capture registers power up containing random data.

Figure 8 shows a block diagram for timers T1 and T0 on the device.

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 4.096 ms at the maximum clock frequency ($t_C = 1\ \mu s$). 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

The device has a powerful timer/counter block, T1.

The timer block consists of a 16-bit timer, T1, and two supporting 16-bit autoreload/capture registers, R1A and R1B. The timer block has two pins associated with it, T1A and T1B. The pin T1A supports I/O required by the timer block, while the pin T1B 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 T1C3, T1C2, and T1C1 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 T1 counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, R1A and R1B. The very first underflow of the timer causes the timer to reload from the register R1A. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register R1B.

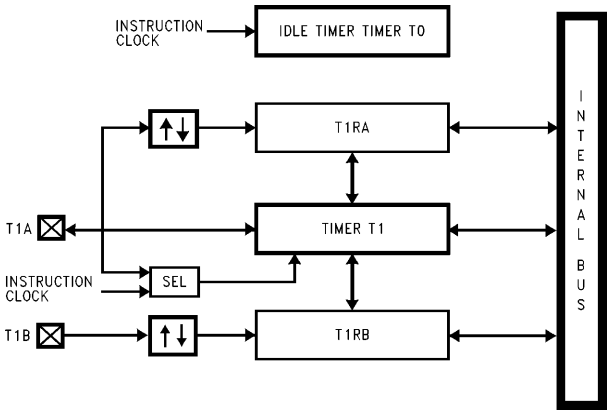
The T1 Timer control bits, T1C3, T1C2 and T1C1 set up the timer for PWM mode operation.

Figure 9 shows a block diagram of the timer in PWM mode. The underflows can be programmed to toggle the T1A output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, T1PNDA and T1PNDB. The user must reset these pending flags under software control. Two control enable flags, T1ENA and T1ENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag T1ENA will cause an interrupt when a timer underflow causes the R1A register to be reloaded into the timer. Setting the timer enable flag T1ENB will cause an interrupt when a timer underflow causes the R1B register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

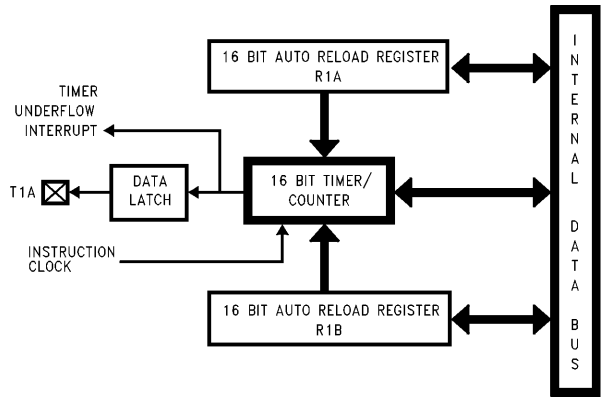
Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Timers (Continued)



TL/DD/12067–8

FIGURE 8. Timers T1 and T0



TL/DD/12067–9

FIGURE 9. Timer 1 in PWM MODE

Mode 2. External Event Counter Mode

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

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

Figure 10 shows a block diagram of the timer in External Event Counter mode.

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

Mode 3. Input Capture Mode

The device can precisely measure external frequencies or time external events by placing the timer block, T1, in the input capture mode.

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

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, T1C3, T1C2 and T1C1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

Timers (Continued)

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

Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer T1C0 pending flag (the T1C0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the T1C0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the T1ENA control flag. When a T1A interrupt occurs in the Input Capture mode, the user must check both the T1PNDA and T1C0 pending flags in order to determine whether a T1A input capture or a timer underflow (or both) caused the interrupt.

Figure 11 shows a block diagram of the timer in Input Capture mode.

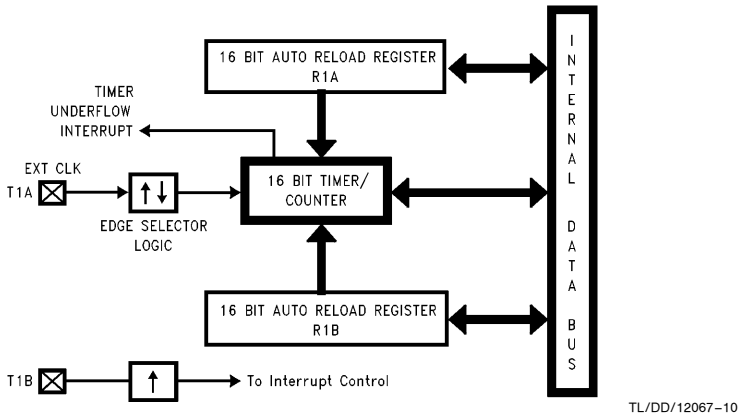


FIGURE 10. Timer 1 in External Event Counter Mode

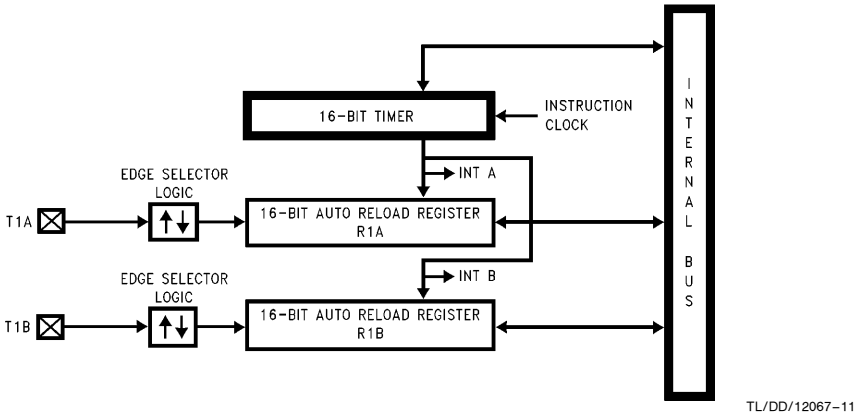


FIGURE 11. Timer 1 in Input Capture Mode

Timers (Continued)

TIMER CONTROL FLAGS

The control bits and their functions are summarized below.

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

T1PNDA Timer Interrupt Pending Flag

T1PNDB Timer Interrupt Pending Flag

T1ENA Timer Interrupt Enable Flag

T1ENB Timer Interrupt Enable Flag
1 = Timer Interrupt Enabled
0 = Timer Interrupt Disabled

T1C3 Timer mode control

T1C2 Timer mode control

T1C1 Timer mode control

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

T1C3	T1C2	T1C1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Positive T1B Edge	T1A Positive Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Positive T1B Edge	T1A Negative Edge
1	0	1	MODE 1 (PWM) T1A Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No T1A Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: T1A Positive Edge T1B Positive Edge	Positive T1A Edge or Timer Underflow	Positive T1B Edge	t_C
1	1	0	MODE 3 (Capture) Captures: T1A Positive Edge T1B Negative Edge	Positive T1A Edge or Timer Underflow	Negative T1B Edge	t_C
0	1	1	MODE 3 (Capture) Captures: T1A Negative Edge T1B Positive Edge	Negative T1A Edge or Timer Underflow	Positive T1B Edge	t_C
1	1	1	MODE 3 (Capture) Captures: T1A Negative Edge T1B Negative Edge	Negative T1A Edge or Timer Underflow	Negative T1B Edge	t_C

HIGH SPEED, CONSTANT RESOLUTION PWM TIMER

The device has one processor independent PWM timer. The PWM timer operates in two modes: PWM mode and capture mode. In PWM mode the timer outputs can be programmed to two pins PWM0 and PWM1. In capture mode, pin PWM0 functions as the capture input. *Figure 12* shows a block diagram for this timer in capture mode and *Figure 13* shows a block diagram for the timer in PWM mode.

PWM Timer Registers

The PWM Timer has three registers: PWMCON, the PWM control register, RLON, the PWM on-time register and PSCAL, the prescaler register.

PWM Prescaler Register (PSCAL) (Address X'00A0)

The prescaler is the clock source for the counter in both PWM mode and in frequency monitor mode.

PSCAL is a read/write register that can be used to program the prescaler. The clock source to the timer in both PWM and capture modes can be programmed to CKI/N where N = PSCAL + 1, so the maximum PWM clock frequency = CKI and the minimum PWM clock frequency = CKI/256. The processor is able to modify the PSCAL register regardless of whether the counter is running or not and the change in frequency occurs with the next underflow of the prescaler (CK-PWM).

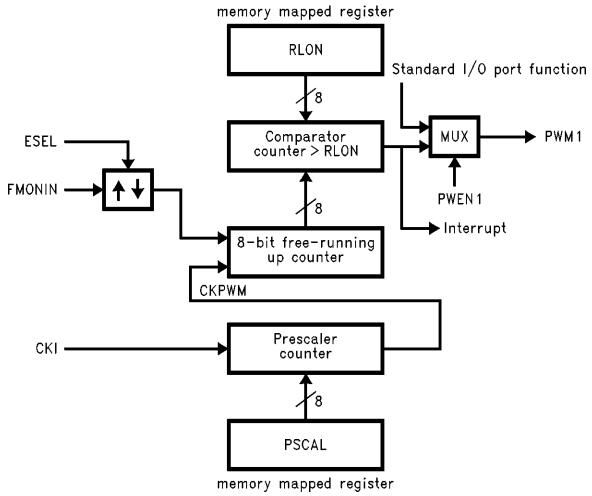
PWM On-time Register (RLON) (Address X'00A1)

RLON is a read/write register. In PWM mode the timer output will be a "1" for RLON counts out of a total cycle of 255 PWM clocks. In capture mode it is used to program the threshold frequency.

The PWM timer is specially designed to have a resolution of 255 PWM clocks. This allows the duty cycle of the PWM output to be selected between 1/255 and 254/255. A value of 0 in the RLON register will result in the PWM output being continuously low and a value of 255 will result in the PWM output being continuously high.

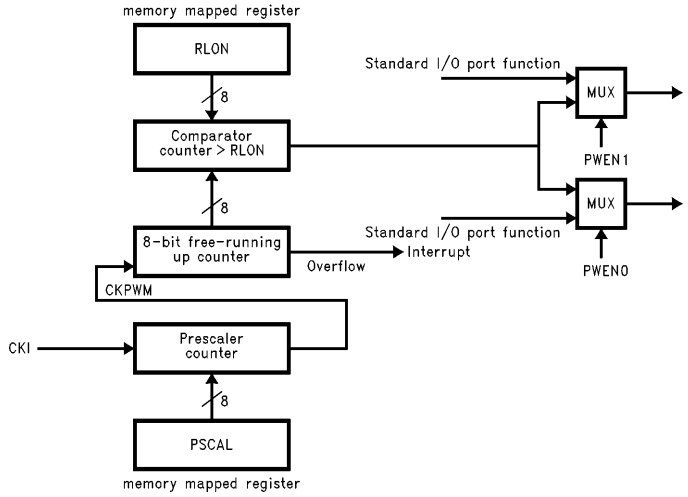
Note: The effect of changing the RLON register during active PWM mode operation is delayed until the boundary of a PWM cycle. In capture mode the effect takes place immediately.

Timers (Continued)



TL/DD/12067–12

FIGURE 12. PWM Timer Capture Mode Block Diagram



TL/DD/12067–13

FIGURE 13. PWM Timer PWM Mode Block Diagram

Timers (Continued)

PWM Control Register (PWMCON) (Address X'00A2)

The PWMCON Register Bits are:

PWEN0 Enable PWM0 output/input function on I/O port.

PWEN1 Enable PWM1 output function on I/O port.

Note: The associated bits in the configuration and data register of the I/O-port have to be setup as outputs and/or inputs in addition to setting the PWEN bits.

PWON PWM start Bit, ''1'' to start timer, ''0'' to stop timer.

PWMD PWM Mode bit, ''1'' for PWM mode, ''0'' for frequency monitor mode.

PWIE PWM interrupt enable bit.

PWPND PWM interrupt pending bit.

ESEL Edge select bit, ''1'' for falling edge, ''0'' for rising edge.

unused	ESEL	PWPND	PWIE	PWMD	PWON	PWEN1	PWEN0

Bit 7 Bit 0

PWM Mode

The PWM timer can generate PWM signals at frequencies up to 39 kHz (@ $t_c = 1\ \mu s$) with a resolution of 255 parts. Lower PWM frequencies can be programmed via the prescaler.

If the PWM mode bit (PWMD) in the PWM configuration register (PWMCON) is set to ''1'' the timer operates in PWM mode. In this mode, the timer generates a PWM signal with a fixed, non-programmable repetition rate of 255 PWM clock cycles. The timer is clocked by the output of an 8-bit, programmable prescaler, which is clocked with the chip's CKI frequency. Thus the PWM signal frequency can be calculated with the formula:

$$\text{fpwm} = \frac{\text{CKI}}{(1 + (\text{PSCAL-contents})) \times 255}$$

Selecting the PWM mode by setting PWMD to ''1'', but not yet starting the timer (PWON is ''0''), will set the timer output to ''1''.

The contents of an 8-bit register, RLON, multiplied by the clock cycle of the prescaler output defines the time between overflow (or starting) and the falling edge of the PWM output.

Once the timer is started, the timer output goes low after RLON cycles and high after a total of 255 cycles. The procedure is continually repeated. In PWM mode the timer is available at pins PWM0 and/or PWM1, provided the port configuration bits for those pins are defined as outputs and the PWEN0 and/or PWEN1 bits in the PWMCON register are set.

The PWM timer is started by the software setting the PWON bit to ''1''. Starting the timer initializes the timer register. From this point, the timer will continually generate the PWM signal, independent of any processor activity, until the timer is stopped by software setting the PWON bit to ''0''. The processor is able to modify the RLON register regardless of whether the timer is running. If RLON is changed while the timer is running, the previous value of RLON is used for comparison until the next overflow occurs, when the new value of RLON is latched into the comparator inputs.

When the timer overflows, the PWM pending flag (PWPND) is set to ''1''. If the PWM interrupt enable bit (PWIE) is also set to ''1'', timer overflow will generate an interrupt. The PWPND bit remains set until the user's software writes a ''0'' to it. If the software writes a ''1'' to the PWPND bit, this has no effect. If the software writes a ''0'' to the PWPND bit at the same time as the hardware writes to the bit, the hardware has precedence.

Note: The software controlling the duty cycle is able to change the PWM duty cycle without having to wait for the timer overflow.

Figure 14 shows how the PWM output is implemented. The PWM Timer output is set to ''1'' on an overflow of the timer and set to ''0'' when the timer is greater than RLON. The output can be multiplexed to two pins.

Capture Mode

If the PWM mode bit (PWMD) is set to ''0'' the PWM Timer operates in capture mode. Capture mode allows the programmer to test whether the frequency of an external source exceeds a certain threshold.

If PWMD is ''0'' and PWON is ''0'', the timer output is set to ''0''. In capture mode the timer output is available at pin PWM1, provided the port configuration register bit for that pin is set up as an output and the PWEN1 bit in the PWMCON register is set. Setting PWON to ''1'' will initialize the timer register and start the counter. A rising edge, or if selected, a falling edge, on the FMONIN input pin will initialize the timer register and clear the timer output. The counter continues to count up after being initialized. The ESEL bit determines whether the active edge is a rising or a falling edge.

Timers (Continued)

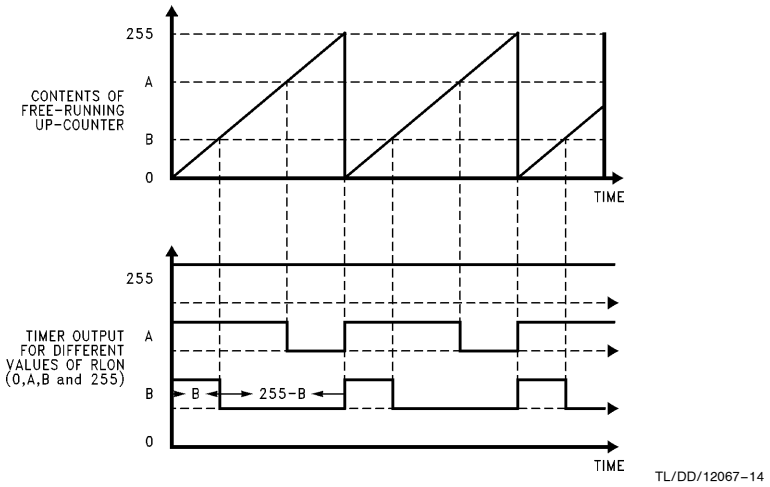


FIGURE 14. PWM Mode Operation

If, in capture mode PWM0 is configured incorrectly as an output and is enabled via the PWEN0 bit, the timer output will feedback into the PWM block as the timer input.

The contents of the counter are continually compared with the RLON register. If the frequency of the input edges is sufficiently high, the contents of the counter will always be less than the value in RLON. However, if the frequency of the input edges is too low, the free-running counter value will count up beyond the value in RLON.

When the counter is greater than RLON, the PWM timer output is set to ''1''. It is set to ''0'' by a detected edge on the timer input or when the counter overflows. When the counter becomes greater than RLON, the PWPND bit in the PWM control register is set to ''1''. If the PWIE bit is also set to ''1'', the PWPND bit is enabled to request an interrupt.

It should be noted that two other conditions could also set the PWPND bit:

1. If the mode of operation is changed on the fly the timer output will toggle. If frequency monitor mode is entered on the fly such that the timer output changes from 0 to 1, PWPND will be set.
2. If the timer is operating in frequency monitor mode and the RLON value is changed on the fly so that RLON becomes less than the current timer value, PWPND will be set.

The PWPND bit remains set until the user's software writes a ''0'' to it. If the software writes a ''1'' to the PWPND bit, this has no effect. If the software writes a ''0'' to the PWPND bit at the same time as the hardware writes to the bit, the hardware has precedence. (See *Figure 15* for Frequency Monitor Mode Operation.)

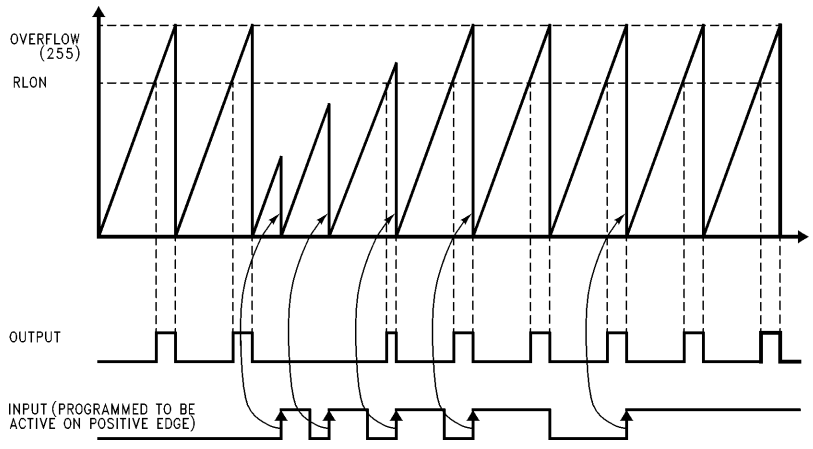


FIGURE 15. Frequency Monitor Mode Operation

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 contents of all PWM Timer registers are frozen during HALT mode and are left unchanged when exiting HALT mode. The PWM timer resumes its previous mode of operation when exiting HALT mode.

The device is 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 the 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 Wake Up feature on the L port. The second method of exiting the HALT mode is by pulling the RESET pin low.

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

The device has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the device will enter and exit the HALT mode as described above. With the HALT disable mask option, the device cannot be placed in the HALT mode (writing a ''1'' to the HALT flag will have no effect).

IDLE MODE

The device is placed in the IDLE mode by writing a ''1'' to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry, and the IDLE Timer T0, are stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

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

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

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

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the device will first execute the Timer T0 interrupt service routine and then return to the instruction following the ''Enter Idle Mode'' instruction.

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 Wake Up

The Multi-Input Wake Up feature is used to return (wake up) the device from either the HALT or IDLE modes. Alternately, the Multi-Input Wake Up/Interrupt feature may also be used to generate up to 7 edge selectable external interrupts.

Figure 16 shows the Multi-Input Wake Up logic for the microcontroller. 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 Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wake Up from the associated port pin.

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

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

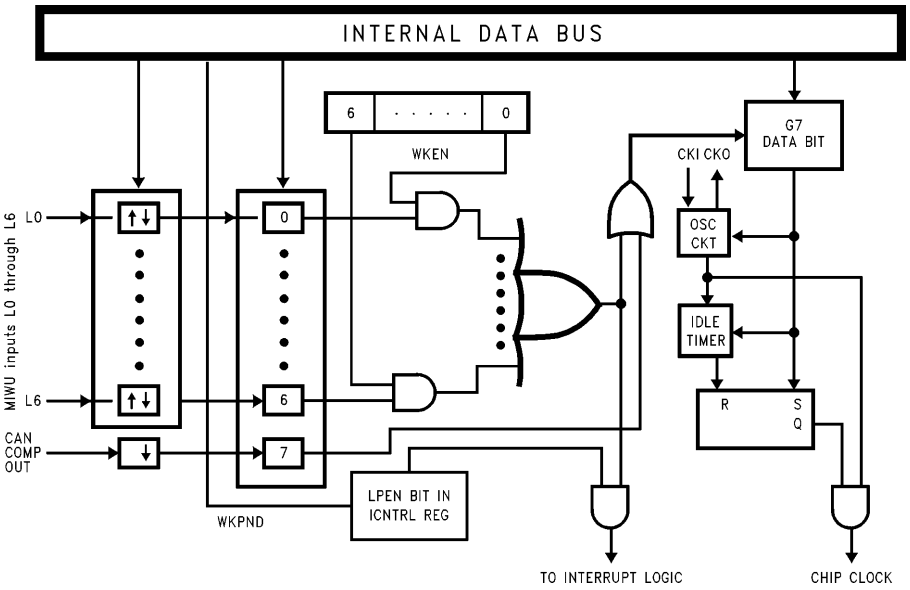
```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

Multi-Input Wake Up (Continued)

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 inherited pseudo wake up 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 has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

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



TL/DD/12067–16

FIGURE 16. Multi-Input Wake Up Logic

Multi-Input Wake Up (Continued)

CAN RECEIVE WAKE UP

The CAN Receive Wake Up source is always enabled and is always active on a falling edge of the CAN comparator output. There is no specific enable bit for the CAN Wake Up feature. Although the wake up feature on pins L0..L6 can be programmed to generate an interrupt (L-port interrupt), no interrupt is generated upon a CAN receive wake up condition. The CAN block has its own, dedicated receiver interrupt upon receive buffer full.

PORT L INTERRUPTS

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

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (global interrupt enable) bit enables the interrupt function. A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

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

The Wake Up signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the device to execute instructions. In this case, upon detecting a valid Wake Up signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_C instruction cycle clock. The t_C clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The start-up time-out from the IDLE timer enables the clock signals to be routed to the rest of the chip.

Interrupts

The device supports a vectored interrupt scheme. It supports a total of eleven interrupt sources. The following table lists all the possible device interrupt sources, their arbitration ranking 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 a Pending bit. 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 ff 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.

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.

Interrupts (Continued)

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).

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

Arbitration Ranking	Source	Vector Address Hi-Low Byte
1	Software Trap	0yFE–0yFF
2	Reserved	0yFC–0yFD
3	CAN Receive	0yFA–0yFB
4	CAN Error (transmit/receive)	0yF8–0yF9
5	CAN Transmit	0yF6–0yF7
6	Pin G0 Edge	0yF4–0yF5
7	IDLE Timer Underflow	0yF2–0yF3
8	Timer T1A/Underflow	0yF0–0yF1
9	Timer T1B	0yEE–0yEF
10	MICROWIRE/PLUS	0yEC–0yED
11	PWM timer	0YEA–0yEB
12	Reserved	0yE8–0yE9
13	Reserved	0yE6–0yE7
14	Reserved	0yE4–0yE5
15	Port L/Wake Up	0yE2–0yE3
16	Default VIS Interrupt	0yE0–0yE1

y is VIS page, y $\neq$ 0

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 context 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 17 shows the Interrupt Block diagram.

SOFTWARE TRAP

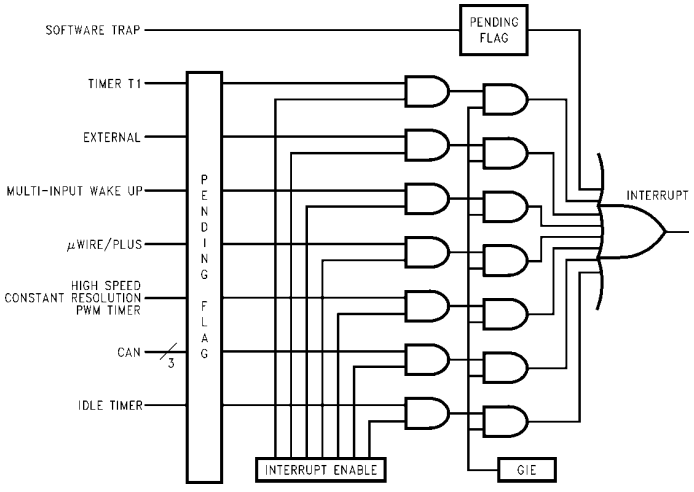
The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is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



TL/DD/12067–17

FIGURE 17. Interrupt Block Diagram

CAN Block Description *

This device contains a CAN serial bus interface as described in the CAN Specification Rev. 2.0 part B.

*Patents Pending.

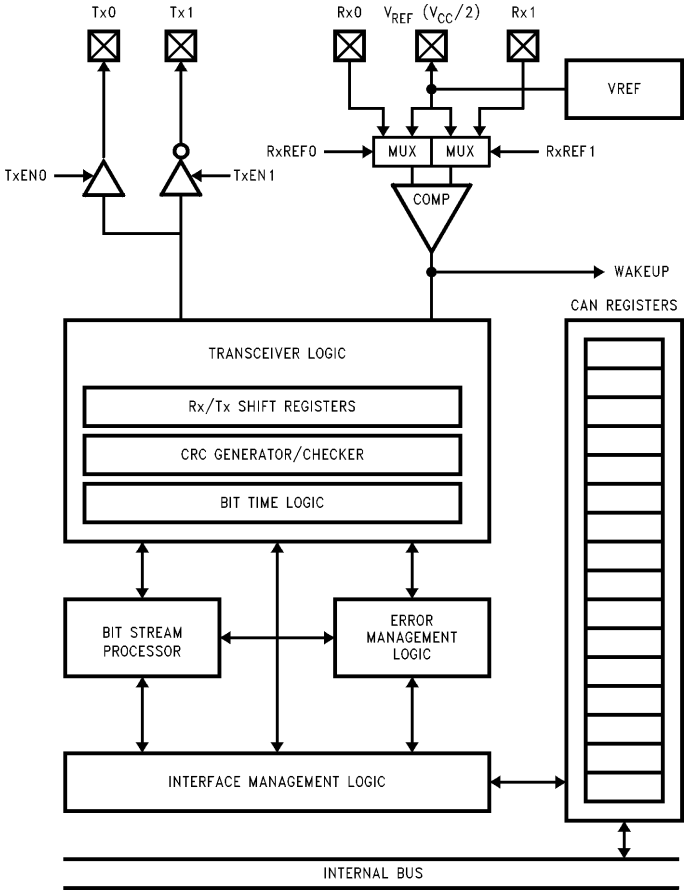
CAN Interface Block

This device supports applications which require a low speed CAN interface. It is designed to be programmed with two transmit and two receive registers. The user's program may check the status bytes in order to get information of the bus state and the received or transmitted messages. The device has the capability to generate an interrupt as soon as one byte has been transmitted or received. Care must be taken if more than two bytes in a message frame are to be transmitted/received. In this case the user's program must poll the transmit buffer empty (TBE)/receive buffer full (RBF) bits or enable their respective interrupts and perform a data exchange between the user data and the Tx/Rx registers.

Fully automatic transmission on error is supported for messages not longer than two bytes. Messages which are longer than two bytes have to be processed by software.

The interface is compatible with CAN Specification 2.0 part B, without the capability to receive/transmit extended frames. Extended frames on the bus are checked and acknowledged according to the CAN specification.

The maximum bus speed achievable with the CAN interface is a function of crystal frequency, message length and software overhead. The device can support a bus speed of up to 1 Mbit/s with a 10 MHz oscillator and 2 byte messages. The 1 Mbit/s bus speed refers to the rate at which protocol and data bits are transferred on the bus. Longer messages require slower bus speeds due to the time required for software intervention between data bytes. The device will support a maximum of 125k bit/s with eight byte messages and a 10 MHz oscillator.



TL/DD/12067–49

FIGURE 15. CAN Interface Block Diagram

Functional Block Description of the CAN Interface

Interface Management Logic (IML)

The IML executes the CPU's transmission and reception commands and controls the data transfer between CPU, Rx/Tx and CAN registers. It provides the CAN Interface with Rx/Tx data from the memory mapped Register Block. It also sets and resets the CAN status information and generates interrupts to the CPU.

Bit Stream Processor (BSP)

The BSP is a sequencer controlling the data stream between The Interface Management Logic (parallel data) and the bus line (serial data). It controls the transceive logic with regard to reception and arbitration, and creates error signals according to the bus specification

Transceive Logic (TCL)

The TCL is a state machine which incorporates the bit stuff logic and controls the output drivers, CRC logic and the Rx/Tx shift registers. It also controls the synchronization to the bus with the CAN clock signal generated by the BTL.

Error Management Logic (EML)

The EML is responsible for the fault confinement of the CAN protocol. It is also responsible for changing the error counters, setting the appropriate error flag bits and interrupts and changing the error status (passive, active and bus off).

Cyclic Redundancy Check (CRC) Generator and Register

The CRC Generator consists of a 15-bit shift register and the logic required to generate the checksum of the destuffed bit-stream. It informs the EML about the result of a receiver checksum.

The checksum is generated by the polynomial:

$$\chi^{15} + \chi^{14} + \chi^{10} + \chi^{8} + \chi^{7} + \chi^{4} + \chi^{3} + 1$$

Receive/Transmit (Rx/Tx) Registers

The Rx/Tx registers are 8-bit shift registers controlled by the TCL and the BSP. They are loaded or read by the Interface Management Logic, which holds the data to be transmitted or the data that was received.

Bit Time Logic (BTL)

The bit time logic divider divides the CKI input clock by the value defined in the CAN prescaler (CSCAL) and bus timing register (CTIM). The resultig bit time (tcan) can be computed by the formula:

$$t_{can} = \frac{CKI}{(1 + divider) \times (1 + 2 \times PS + PPS)}$$

Where *divider* is the value of the clock prescaler, *PS* is the programmable value of phase segment 1 and 2 (1..8) and *PPS* the programmed value of the propagation segment (1..8) (located in CTIM).

Bus Timing Considerations

The internal architecture of the CAN interface has been optimized to allow fast software response times within messages of more than two data bytes. The TBE (Transmit Buffer Empty) bit is set on the last bit of odd data bytes when CAN internal sample points are high.

It is the user's responsibility to ensure that the time between setting TBE and a reload of TxD2 is longer than the length of phase segment 2 as indicated in the following equation:

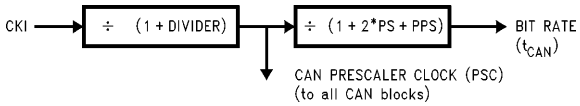
$$t_{LOAD} > \frac{(PS + 1) \times (CSCAL + 1)}{10} t_C = \text{absolute length of PS2}$$

Table V shows examples of the minimum required t_{LOAD} for different CSCAL settings based on a clock frequency of 10 MHz. Lower clock speeds require recalculation of the CAN bit rate and the mimimum t_{LOAD}.

TABLE V. CAN Timing (CKI = 10 MHz t_c = 1 μs)

PS	CSCAL	CAN Bit Rate (kbit/s)	Minimum t_{LOAD} (μs)
4	3	250	2.0
4	9	100	5.0
4	15	62	8.0
4	24	40	12.5
4	39	25	20
4	99	10	50
4	199	5	100

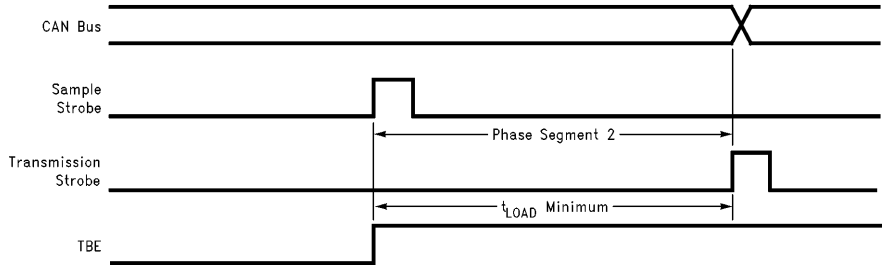
Functional Block Description of the CAN Interface (Continued)



TL/DD/12067–50

FIGURE 16. Bit Rate Generation

Figure 17 illustrates the minimum time required for t_{LOAD}.



TL/DD/12067–51

FIGURE 17. TBE Timing

In the case of an interrupt driven CAN interface, the calculation of the actual t_{LOAD} time would be done as follows:

```
INT:                                    ;Interrupt latency = 7tc = 7 µs
                PUSH        A           ;3tc = 3 µs
                LD          A,B         ;2tc = 2 µs
                PUSH        A           ;3tc = 3 µs
                VIS                     ;5tc = 5 µs
CANTX:                                  ;20tc = µs to this point
                •                       ;additional time for instructions which check
                •                       ;status prior to reloading the transmit data
                •                       ;registers with subsequent data bytes.
                LD          TXD2,DATA
                •
                •
                •
```

Functional Block Description of the CAN Interface (Continued)

Interrupt driven programs use more time than programs which poll the TBE flag, however programs which operate at lower baud rates (which are more likely to be sensitive to this issue) have more time for interrupt response.

Output Drivers/Input Comparators

The output drivers/input comparators are the physical interface to the bus. Control bits are provided to TRI-STATE the output drivers.

A dominant bit on the bus is represented as a ''0'' in the data registers and a recessive bit on the bus is represented as a ''1'' in the data registers.

TABLE VI. Bus Level Definition

Bus Level	Pin Tx0	Pin Tx1	Data
''dominant''	drive low (GND)	dirve high (V_{CC})	0
''recessive''	TRI-STATE	TRI-STATE	1

Register Block

The register block consists of fifteen 8-bit registers which are described in more detail in the following paragraphs.

Note: The contents of the receiver related registers RxD1, RxD2, RDLC, RIDH and RTSTAT are only changed if a received frame passes the acceptance filter or the Receive Identifier Acceptance Filter bit (RIAF) is set to accept all received messages.

TRANSMIT DATA REGISTER 1 (TXD1) (Address X'00B0)

The Transmit Data Register 1 contians the first data byte to be transmitted within a frame and then the successive odd byte numbers (i.e., bytes number 1,3,..,7).

TRANSMIT DATA REGISTER 2 (TXD2) (Address X'00B1)

The Transit Data Register 2 contains the second data byte to be transmitted within a frame and then the successive even byte numbers (i.e., bytes number 2,4,..,8).

TRANSMIT DATA LENGTH CODE AND IDENTIFIER LOW REGISTER (TDLC) (Address X'00B2)

TID3	TID2	TID1	TID0	TDLC3	TDLC2	TCLC1	TDLC0

Bit 7 Bit 0

This register is read/write.

TID3..TIDO Transmit Identifier Bits 3..0 (lower 4 bits)

The transmit identifier is composed of eleven bits in total, bits 3 to 0 of the TID are stored in bits 7 to 4 of this register.

TDLC3..TDLC0 Transmit Data Length Code

These bits determine the number of data bytes to be transmitted within a frame. The CAN specification allows a maximum of eight data bytes in any message.

TRANSMIT IDENTIFIER HIGH (TID) (Address X'00B3)

TRTR	TID10	TID9	TID8	TID7	TID6	TID5	TID4

Bit 7 Bit 0

This register is read/write.

TRTR Transmit Remote Frame Request

This bit is set if the frame to be transmitted is a remote frame request.

TID10..TID4 Transmit Identifier Bits 10 .. 4 (higher 7 bits)

Bits TID10..TID4 are the upper 7 bits of the 11 bit transmit identifier.

RECEIVER DATA REGISTER 1 (RXD1) (Address X'00A4)

The Receive Data Register 1 (RXD1) contains the first data byte received in a frame and then successive odd byte numbers (i.e., bytes 1, 3,..7). This register is read-only.

RECEIVE DATA REGISTER 2 (RXD2) (Address X'00A5)

The Receive Data Register 2 (RXD2) contains the second data byte received in a frame and then successive even byte numbers (i.e., bytes 2,4,..,8). This register is read-only.

REGISTER DATA LENGTH CODE AND IDENTIFIER LOW REGISTER (RIDL) (Address X'00B6)

RID3	RID2	RID1	RID0	RDLC3	RDLC2	RDLC1	RDLC0

Bit 7 Bit 0

This register is read only.

RID3..RID0 Receive Identifier bits (lower four bits)

The RID3..RID0 bits are the lower four bits of the eleven bit long Receive Identifier. Any received message that matches the upper 7 bits of the Receive Identifier (RID10..RID4) is accepted if the Receive Identifier Acceptance Filter (RIAF) bit is set to zero.

RDLC3..RDLC0 Receive Data Length Code bits

The RDLC3..RDLC0 bits determine the number of data bytes within a received frame.

RECEIVE IDENTIFIER HIGH (RID) (Address X'00B7)

unused	RID10	RID9	RID8	RID7	RID6	RID5	RID4

Bit 7 Bit 0

This register is read/write.

RID10..RID4 Receive Identifier bits (upper bits)

The RID10...RID4 bits are the upper 7 bits of the eleven bit long Receive Identifier. If the Receive Identifier Acceptance Filter (RIAF) bit (see CBUS register) is set to zero, bits 4 to 10 of the received identifier are compared with the mask bits of RID4..RID10. If the corresponding bits match, the message is accepted. If the RIAF bit is set to a one, the filter function is disabled and all messages, independent of identifier, will be accepted.

Functional Block Description of the CAN Interface (Continued)

CAN PRESCALER REGISTER (CSCAL)
(Address X'00B8)

CKS7	CKS6	CKS5	CKS4	CKS3	CKS2	CKS1	CKS0
Bit 7							Bit 0

This register is read/write.

CKS7..0 Prescaler divider select.

The resulting clock value is the CAN Prescaler clock.

CAN BUS TIMING REGISTER (CTIM) (00B9)

PPS2	PPS1	PPS0	PPS0	PS2	PS1	PS0	Reserved
Bit 7							Bit 0

This register is read/write.

PPS2..PPS0 Propagation Segment, bits 2..0

The PPS2..PPS0 bits determine the length of the propagation delay in Prescaler clock cycles (PSC) per bit time. (For a more detailed discussion of propagation delay and phase segments, see SYNCHRONIZATION on page 41.)

PS2..PS0 Phase Segment 1, bits 2..0

The PS2..PS0 bits fix the number of Prescaler clock cycles per bit time for phase segment 1 and phase segment 2. The PS2..PS0 bits also set the synchronization Jump Width to a value equal to the lesser of the 4 PSC or the length of PS1/2 (Min: 4 l length of PS1/2).

TABLE VII. Synchronization Jump Width

PS2	PS1	PS0	Length of Phase Segment ½	Synchronization Jump Width
0	0	0	1 t_{can}	1 t_{can}
0	0	1	2 t_{can}	2 t_{can}
0	1	0	3 t_{can}	3 t_{can}
0	1	1	4 t_{can}	4 t_{can}
1	0	0	5 t_{can}	4 t_{can}
1	0	1	6 t_{can}	4 t_{can}
1	1	0	7 t_{can}	4 t_{can}
1	1	1	8 t_{can}	4 t_{can}

LENGTH OF TIME SEGMENTS (See *Figure 29*)

- The Synchronization Segment is 1 CAN Prescaler clock (PSC)
- The Propagation Segment can be programmed (PPS) to be 1,2...,8 PSC in length.
- Phase Segment 1 and Phase Segment 2 are programmable (PS) to be 1,2,..,8 PSC long.

Note: (BTL settings at high speed; PSC = 0) Due to the on-chip delay from the rx-pins through the receive comparator (worst case assumption: 3 clocks delay * 2 (devices on the bus) + 1 tx delay) the user needs to set the sample point to > (2*3 + 1) i.e., > 7 CKI clocks to ensure correct communication on the bus under all circumstances. With prescaler settings of > 0 this is a given (i.e., no caution has to be applied).

Example: for 1 Mbit CTIM = b'10000100 (PSS = 5; PS1 = 2). Example for 500 kbit CTIM = b'01011100 (PPS = 3; PS1 = 8). – all at 10 MHz CKI and CSCAL = 0.

CAN BUS CONTROL REGISTER (CBUS) (00BA)

Reserved	RIAF	TxEN1	TxEN0	RxREF1	RxREF0	Reserved	FMOD
Bit 7							Bit 0

Reserved This bit is reserved and should be zero.

RIAF Receive identifier acceptance filter bit

If the RIAF bit is set to zero, bits 4 to 10 of the received identifier are compared with the mask bits of RID4..RID10 and if the corresponding bits match, the message is accepted. If the RIAF bit is set to a one, the filter function is disabled and all messages independent of the identifier will be accepted.

TxEN0, TxEN1 TxD Output Driver Enable

TABLE VIII. Output Drivers

TxEN1	TxEN0	Output
0	0	Tx0, Tx1 TRI-STATE, CAN input comparator disabled
0	1	Tx0 enabled
1	0	Tx1 enabled
1	1	Tx0 and Tx1 enabled

Bus synchronization of the device is done in the following way:

If the output was disabled (TxEN1, TxEN0 = "0") and either TxEN1 or TxEN0, or both are set to 1, the device will not start transmission or reception of a frame until eleven consecutive "recessive" bits have been received. Resetting the TxEN1 and TxEN0 bits will disable the output drivers and the CAN input comparator. All other CAN related registers and flags will be unaffected. It is recommended that the user reset the TxEN1 and TxEN0 bits before switching the device into the HALT mode (the CAN receive wakeup will still work) in order to reduce current consumption and to assure a proper resychronization to the bus after exiting the HALT mode.

Note: A "bus off" condition will also cause Tx0 and Tx1 to be at TRI-STATE (independent of the values of the TxEN1 and TxEN0 bits).

RXREF1 Reference voltage applied to Rx1 if bit is set

RXREF0 Reference voltage applied to Rx0 if bit is set

FMOD Fault Confinement Mode select

Setting the FMOD bit to "0" (default after power on reset) will select the Standard Fault Confinement mode. In this mode the device goes from "bus off" to "error active" after monitoring 128*11 recessive bits (including bus idle) on the bus. This mode has been implemented for compatibility with existing solutions. Setting the FMOD bit to "1" will select the Enhanced Fault Confinement mode. In this mode the device goes from "bus off" to "error active" after monitoring 128 "good" messages, as indicated by the reception of 11 consecutive "recessive" bits including the End of Frame, whereas the standard mode may time out after 128 x 11 recessive bits (e.g., bus idle).

Functional Block Description of the CAN Interface (Continued)

TRANSMIT CONTROL/STATUS (TCNTL) (00BB)

NS1	NS0	TERR	RERR	CEIE	TIE	RIE	TXSS
Bit 7							Bit 0

NS1..NS0 Node Status, i.e., Error Status.

TABLE IX. Node Status

NS1	NS0	Output
0	0	Error active
0	1	Error passive
1	0	Bus off
1	1	Bus off

The Node Status bits are read only.

TERR Transmit Error

This bit is automatically set when an error occurs during the transmission of a frame. TERR can be programmed to generate an interrupt by setting the Can Error Interrupt Enable bit (CEIE). This bit must be cleared by the user's software.

Note: This is used for messages for more than two bytes. If an error occurs during the transmission of a frame with more than 2 data bytes, the user's software has to handle the correct reloading of the data bytes to the TxD registers for retransmission of the frame. For frames with 2 or less data bytes the interface logic of this chip does an automatic retransmission. Regardless of the number of data bytes, the user's software must reset this bit if CEIE is enabled. Otherwise a new interrupt will be generated immediately after return from the interrupt service routine.

RERR Receiver Error

This bit is automatically set when an error occurred during the reception of a frame. RERR can be programmed to generate an interrupt by setting the Can Error Interrupt Enable bit (CEIE). This bit has to be cleared by the user's software.

CEIE CAN Error Interrupt Enable

If set by the user's software, this bit enables the tansmit and receive error interrupts. The interrupt pending flags are TERR and RERR. Resetting this bit with a pending error interrupt will inhibit the interrupt, but will not clear the cause of the interrupt (RERR or TERR). If the bit is then set without clearing the cause of the interrupt, the interrupt will reoccur.

TIE Transmit Interrupt Enable

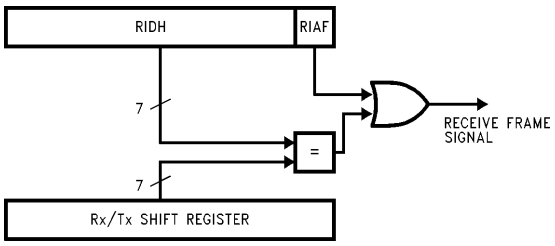
If set by the user's software, this bit enables the transmit interrupt. (See TBE and TXPND.) Resetting this bit with a pending transmit interrupt will inhibit the interrupt, but will not clear the cause of the interrupt. If the bit is then set without clearing the cause of the interrupt, the interrupt will reoccur.

RIE Receive Interrupt Enable

If set by the user's software, this bit enables the receive interrupt or a remote transmission request interrupt (see RBF, RFV and RRTR). Resetting this bit with a pending receive interrupt will inhibit the interrupt, but will not clear the cause of the interrupt. If the bit is then set without clearing the cause of the interrupt, the interrupt will reoccur.

TXSS Transmission Start/Stop

This bit is set by the user's software to initiate the transmission of a frame. Once this bit is set, a transmission is pending, as indicated by the TXPND flag being set. It can be reset by software to cancel a pending transmission. Resetting the TXSS bit will only cancel a transmission, if the transmission of a frame hasn't been started yet (bus idle), if arbitration has been lost (receiving) or if an error occurs during transmission. If the device has already started transmission (won arbitration) the TXPND and TXSS flags will stay set until the transmission is completed, even if the user's software has written zero to the TXSS bit. If one or more data bytes are to be transmitted, care must be taken by the user, that the Transmit Data Register(s) have been loaded before the TXSS bit is set. TXSS will be cleared on three conditions only: Successful completion of a transmitted message; successful cancellation of a pending transmision; Transition of the CAN interface to the bus-off state.



TL/DD/12067–52

FIGURE 18. Acceptance Filter Block-Diagram

Functional Block Description of the CAN Interface (Continued)

Writing a zero to the TXSS bit will request cancellation of a pending transmission but TXSS will not be cleared until completion of the operation. If an error occurs during transmission of a frame, the logic will check for cancellation requests prior to restarting transmission. If zero has been written to TXSS, retransmission will be canceled.

RECEIVE/TRANSMIT STATUS (RTSTAT) (Address X'00BC)

TBE	TXPND	RRTR	ROLD	RORN	RFV	RCV	RBF
1	0	0	0	0	0	0	0

Bit 7 Bit 0

This register is read only.

TBE Transmit Buffer Empty

This bit is set as soon as the TxD2 register is copied into the Rx/Tx shift register, i.e., the 1st data byte of each pair has been transmitted. The TBE bit is automatically reset if the TxD2 register is written (the user should write a dummy byte to the TxD2 register when transmitting an odd number of bytes of zero bytes). TBE can be programmed to generate an interrupt by setting the Transmit Interrupt Enable bit (TIE). When servicing the interrupt the user has to make sure that TBE gets cleared by executing a WRITE instruction on the TxD2 register, otherwise a new interrupt will be generated immediately after return from the interrupt service routine. The TBE bit is read only. It is set to 1 upon reset. TBE is also set upon completion of transmission of a valid message.

TXPND Transmission Pending

This bit is set as soon as the Transmit Start/Stop (TXSS) bit is set by the user. It will stay set until the frame was successfully transmitted, until the transmission was successfully canceled by writing zero to the Transmission Start/Stop bit (TXSS), or the device enters the bus-off state. Resetting the TXSS bit will only cancel a transmission if the transmission of a frame hasn't been started yet (bus idle) or if arbitration has been lost (receiving). If the device has already started transmission (won arbitration) the TXPND flag will stay set until the transmission is completed, even if the user's software has requested cancellation of the message. If an error occurs during transmission, a requested cancellation may occur prior to the begining of retransmission.

RRTR Received Remote Transmission Request

This bit is set when the remote transmission request (RTR) bit in a received frame was set. It is automatically reset through a read of the RXD1 register.

To detect RRTR the user can either poll this flag or enable the receive interrupt (the reception of a remote transmission request will also cause an interrupt if the receive interrupt is enabled). If the receive interrupt is enabled, the user should check the RRTR flag in the service routine in order to distinguish between a RRTR interrupt and a RBF interrupt. It is the responsibility of the user to clear this bit by reading the RXD1 register, before the next frame is received.

ROLD Received Overload Frame

This bit is automatically set when an Overload Frame was received on the bus. It is automatically reset through a read of the Receive/Transmit Status register. It is the responsibility of the user to clear this bit by reading the Receive/Transmit Status register, before the next frame is received.

RORN Receiver Overrun

This bit is automatically set on an overrun of the receive data register, i.e., if the user's program does not maintain the RxDn registers when receiving a frame. It it automatically reset through a read of the Receive/Transmit Status register. It is the responsibility of the user to clear this bit by reading the Receive/Transmit Status register before the next frame is received.

RFV Received Frame Valid

This bit is set if the received frame is valid, i.e., after the penultimate bit of the End of Frame is received. It is automatically reset through a read of the Receive/Transmit Status register. It is the responsibility of the user to clear this bit by reading the receive/transmit status register (RTSTAT), before the next frame is received. RFV will cause a Receive Interrupt if enabled by RIE. The user should be careful to read the last data byte (RxD1) of odd length messages (1, 3, 5 or 7 data bytes) on receipt of RFV. RFV is the only indication that the last byte of the message has been received.

RCV Receive Mode

This bit is set after the data length code of a message that passes the device's acceptance filter has been received. It is automatically reset after the CRC-delimiter of the same frame has been received. It indicates to the user's software that arbitration is lost and that data is coming in for that node.

RBF Receive Buffer Full

This bit is set if the second Rx data byte was received. It is reset automatically, after the RxD1-Register has been read by the software. RBF can be programmed to generate an interrupt by setting the Receive Interrupt Enable bit (RIE). When servicing the interrupt, the user has to make sure that RBF gets cleared by executing a LD instruction from the RxD1 register, otherwise a new interrupt will be generated immediately after return from the interrupt service routine. The RBF bit is read only.

TRANSMIT ERROR COUNTER (TEC) (Address X'00BD)

TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0

Bit 7 Bit 0

This register is read/write.

Functional Block Description of the CAN Interface (Continued)

For test purposes and to identify the node status, the transmit error counter, an 8-bit error counter, is mapped into the data memory. If the lower seven bits of the counter overflow, i.e., TEC7 is set, the device is error passive.

CAUTION

To prevent interference with the CAN fault confinement, the user must not write to the REC/TEC registers. Both counters are automatically updated following the CAN specification.

RECEIVE ERROR COUNTER (REC) (00BE)

ROVL	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Bit 7							Bit 0

This register is read/write.

ROVL receive error counter overflow

For test purposes and to identify the node status the receive error counter, a 7-bit error counter, is mapped into the data memory. If the counter overflows the ROVL bit is set to indicate that the device is error passive and won't transmit any active error frames. If ROVL is set then the counter is frozen.

MESSAGE IDENTIFICATION

a. Transmitted Message

The user can select all 11 Transmit Identifier Bits to transmit any message whigh fulfills the CAN2.0, part B spec without an extended identifier (see note below). Fully automatic retransmission is supported for messages no longer than 2 bytes.

b. Received Messages

The lower four bits of the Receive Identifier are don't care, i.e., the controller will receive all messages that fit in that window (16 messages). The upper 7 bits can be defined by the user in the Receive Identifier High Register to mask out groups of messages. If the RIAF bit is set, all messages will be received.

Note: The CAN interface tolerates the extended CAN frame format of 29 identifier bits and gives an acknowledgment. If an error occurs the receive error counter will be increased, and decreased if the frame is valid.

BUS SYNCHRONIZATION DURING OPERATION

Resetting the TxEN1 and TxEN0 bits in Bus Control Register will disable the output drivers and do a resynchronization to the bus. All other CAN related registers and flags will be unaffected.

Bus synchronization of the device is this case is done in the following way:

If the output was disabled (TxEN1, TxEN0 = ''0'') and either TxEN1 or TxEN0, or both are set to 1, the device will not start transmission or reception of a frame until eleven consecutive ''recessive'' bits have been received.

A ''bus off'' condition will also cause the output drivers Tx1 and Tx0 to be at TRI-STATE (independent of the status of TxEN1 and TxEN0). The device will switch from ''bus off'' to ''error active'' mode as described under the FMOD-bit description (see Can Bus Control register). This will ensure that the device is synchronized to the bus, before starting to transmit or receive.

For information on bus synchronization and status of the CAN related registers after external reset refer to the RESET section.

ON-CHIP VOLTAGE REFERENCE

The on-chip voltage reference is a ratiometric reference. For electrical characteristics of the voltage reference refer to the electrical specifications section.

ANALOG SWITCHES

Analog switches are used for selecting between Rx0 and V_{REF} and between Rx1 and V_{REF}.

Basic CAN Concepts

The following paragraphs provide a generic overview of the basic concepts of the Controller Area Network (CAN) as described in *Chapter 4 of ISO/DIS11519-1*. Implementation related issues of the National Semiconductor device will be discussed as well.

This device will process standard frame format only. Extended frame formats will be acknowledged, however the data will be discarded. For this reason the description of frame formats in the following section will cover only the standard frame format.

The following section provides some more detail on how the device will handle received extended frames:

If the device's remote identifier acceptance filter bit (RIAF) is set to ''1'', extended frame messages will be acknowledged. However, the data will be discarded and the device will not reply to a remote transmission request received in extended frame format. If the device's RIAF bit is set to ''0'', the upper 7 received ID bits of an extended frame that match the device's receive identifier (RID) acceptance filtler bits, are stroed in the device's RID register. However, the device does not reply to an RTR and any data is discarded. The device will only acknowledge the message.

MULTI-MASTER PRIORITY BASED BUS ACCESS

The CAN protocol is message based protocol that allows a total of 2032 ($= 2^{11} - 16$) different messages in the standard format and 512 million ($= 2^{29} - 16$) different messages in the extended frame format.

MULTICAST FRAME TRANSFER BY ACCEPTANCE FILTERING

Every CAN Frame is put on the common bus. Each module receives every frame and filters out the frames which are not required for the module's task.

REMOTE DATA REQUEST

A CAN master module has the ability to set a specific bit called the ''remote transmission request bit'' (RTR) in a frame. This causes another module, either another master or a slave, to transmit a data frame after the current frame has been completed.

Basic CAN Concepts (Continued)

SYSTEM FLEXIBILITY

Additional modules can be added to an existing network without a configuration change. These modules can either perform completely new functions requiring new data or process existing data to perform a new function.

SYSTEM WIDE DATA CONSISTENCY

As the CAN network is message oriented, a message can be used like a variable which is automatically updated by the controlling processor. If any module cannot process information it can send an overload frame. The device is incapable of initiating an overload frame, but will join a overload frame initiated by another device as required by CAN specifications.

NON-DESTRUCTIVE CONTENTION-BASED ARBITRATION

The CAN protocol allows several transmitting modules to start a transmission at the same time as soon as they monitor the bus to be idle. During the start of transmission every node monitors the bus line to detect whether its message is overwritten by a message with a higher priority. As soon as a transmitting module detects another module with a higher priority accessing the bus, it stops transmitting its own frame and switches to receive mode. For illustration see *Figure 19*.

AUTOMATIC RETRANSMISSION OF FRAMES

If a data or remote frame is overwritten by either a higher-prioritized data frame, remote frame or an error frame, the transmitting module will automatically retransmit it. This device will handle the automatic retransmission of up to two data bytes automatically. Messages with more than 2 data bytes require the user's software to update the transmit registers.

ERROR DETECTION AND ERROR SIGNALING

All messages on the bus are checked by each CAN node and acknowledge if they are correct. If any node detects an error it starts the transmission of an error frame.

Switching Off Defective Nodes

There are two error counters, one for transmitted data and one for received data, which are incremented, depending on the error type, as soon as an error occurs. If either counter goes beyond a specific value the node goes to an error state. A valid frame causes the error counters to decrease.

The device can be in one of three states with respect to error handling:

- Error active
  An error active unit can participate in bus communication and sends an active (''dominant'') error flag.
- Error passive
  An error passive unit can participate in bus communication. However, if the unit detects an error it is not allowed to send an active error flag. The unit sends only a passive (''recessive'') error flag.
- Bus off

A unit that is ''bus off'' has the output drivers disabled, i.e., it does not participate in any bus activity.

(See ERROR MANAGEMENT AND DETECTION for more detailed information.)

Frame Formats

INTRODUCTION

There are basically two different types of frames used in the CAN protocol.

The data transmission frames are: data/remote frame

The control frames are: error/overload frame

Note: This device cannot send an overload frame as a result of not being able to process all information. However, the device is able to recognize an overload condition and join overload frames initiated by other devices.

If no message is being transmitted, i.e., the bus is idle, the bus is kept at the ''recessive'' level. *Figure 20* and *Figure 21* give an overview of the various CAN frame formats.

DATA AND REMOTE FRAME

Data frames consist of seven bit fields and remote frames consist of six different bit fields:

1. Start of Frame (SOF)
2. Arbitration field
3. Control field (IDE bit, R0 bit, and DLC field)
4. Data field (not in remote frame)
5. CRC field
6. ACK field
7. End of Frame (EOF)

A remote frame has no data field and is used for requesting data from other (remote) CAN nodes. *Figure 22* shows the format of a CAN data frame.

FRAME CODING

Remote and Data Frames are NRZ codes with bit-stuffing in every bit field which holds computable information for the interface, i.e., Start of Frame arbitration field, control field, data field (if present) and CRC field.

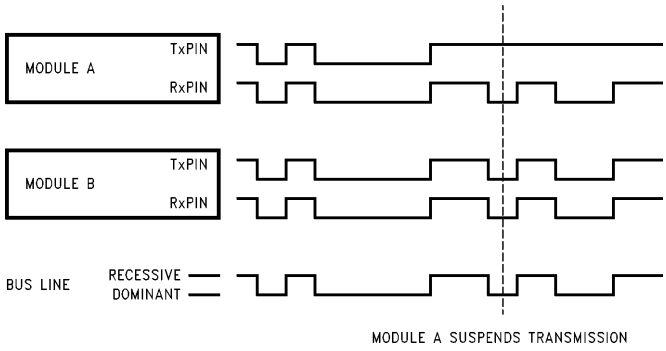
Error and overload frames are NRZ coded without bit stuffing.

BIT STUFFING

After five consecutive bits of the same value, a stuff bit of the inverted value is inserted by the transmitter and deleted by the receiver.

Destuffed Bit Stream	100000x	011111x
Stuffed Bit Stream	1000001x	0111110x
		x = {0,1}

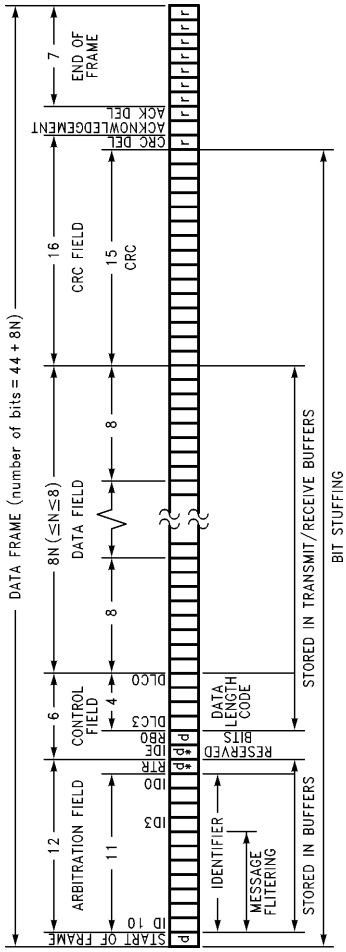
Basic CAN Concepts (Continued)



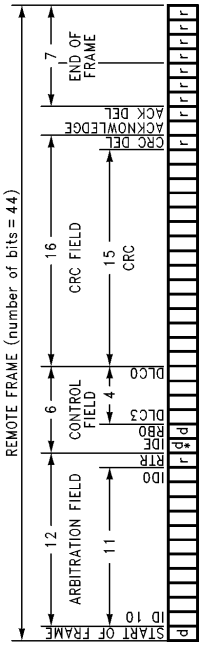
TL/DD/12067–53

FIGURE 19. CAN Message Arbitration

Basic CAN Concepts (Continued)



TL/DD/12067–54

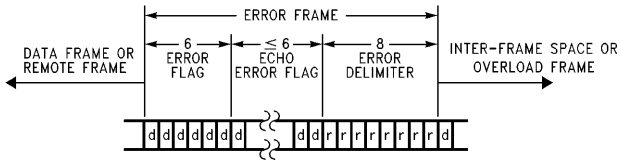


TL/DD/12067–55

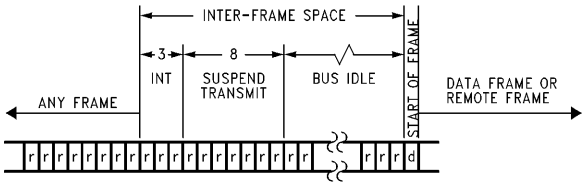
A remote frame is identical to a data frame, except that the RTR bit is "recessive", and there is no data field.

IDE = Identifier Extension Bit
The IDE bit in the standard format is transmitted "dominant", whereas in the extended format the IDE bit is "recessive" and the id is expanded to 29 bits.
r = recessive
d = dominant

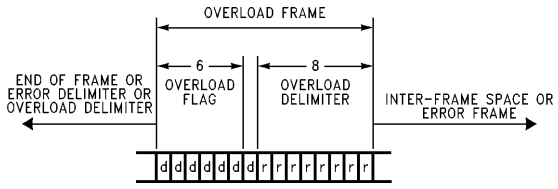
FIGURE 20. CAN Data Transmission Frames



An error frame can start anywhere in the middle of a frame.

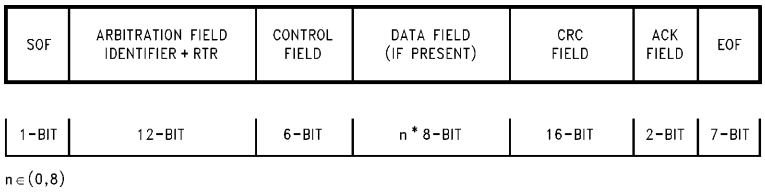


INT = Intermission
Suspend Transmission is only for error passive nodes.



An overload frame can only start at the end of a frame.

FIGURE 21. CAN Control Frames



TL/DD/12067–59

FIGURE 22. CAN Frame Format

START OF FRAME (SOF)

The Start of Frame indicates the beginning of data and remote frames. It consists of a single ''dominant'' bit. A node is only allowed to start transmission when the bus is idle. All nodes have to synchronize to the leading edge (first edge after the bus was idle) caused by SOF of the node which starts transmission first.

ARBITRATION FIELD

The arbitration field is composed of the identifier field and the RTR (Remote Transmission Request) bit. The value of the RTR bit is ''dominant'' in a data frame and ''recessive'' in a remote frame.

CONTROL FIELD

The control field consists of six bits. It starts with two bits reserved for future expansion followed by the four-bit Data Length Code. Receivers must accept all possible combinations of the two reserved bits. Until the function of these reserved bits is defined, the transmitter only sends ''0'' (dominant) bits. The first reserved bit (IDE) is actually defined to indicate an extended frame with 29 Identifier bits if set to ''1''. CAN chips must tolerate extended frames, even if they can only understand standard frames, to prevent the destruction of an extended frames on an existing network.

The Data Length Code indicates the number of bytes in the data field. This Data Length Code consists of four bits. The data field can be of length zero. The permissible number of data bytes for a data frame ranges from 0 to 8.

DATA FIELD

The Data field consists of the data to be transferred within a data frame. It can contain 0 to 8 bytes and each byte contains 8 bits. A remote frame has no data field.

CRC FIELD

The CRC field consists of the CRC sequence followed by the CRC delimiter. The CRC sequence is derived by the transmitter from the modulo 2 division of the preceding bit fields, starting with the SOF up to the end of the data field, excluding stuff-bits, by the generator polynomial:

$$\chi^{15} + \chi^{14} + \chi^{10} + \chi^{8} + \chi^{7} + \chi^{4} + \chi^{3} + 1$$

The remainder of this division is the CRC sequence transmitted over the bus. On the receiver side the module divides all bit fields up to the CRC delimiter, excluding stuff-bits, and checks if the result is zero. This will then be interpreted as a valid CRC. After the CRC sequence a single ''recessive'' bit is transmitted as the CRC delimiter.

ACK FIELD

The ACK field is two bits long and contains the ACK slot and the ACK delimiter. The ACK slot is filled with a ''recessive'' bit by the transmitter. This bit is overwritten with a ''dominant'' bit by every receiver that has received a correct CRC sequence. The second bit of the ACK field is a ''recessive'' bit called the acknowledge delimiter. As a consequence the acknowledge flag of a valid frame is surrounded by two ''recessive'' bits, the CRC-delimiter and the ACK delimiter.

EOF FIELD

The End of Frame Field closes a data and a remote frame. It consists of seven ''recessive'' bits.

INTERFRAME SPACE

Data and remote frames are separate from every preceding frame (data, remote, error and overload frames) by the interframe space see *Figure 23* and *Figure 24* for details. Error and overload frames are not preceded by an interframe space. They can be transmitted as soon as the condition occurs. The interframe space consists of a minimum of three bit fields depending on the error state of the node.

These bit fields are coded as follows:

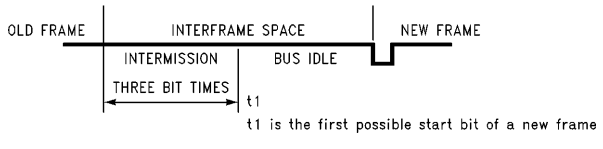
The intermission has the fixed form of three ''recessive'' bits. While this bit field is active, no node is allowed to start a transmission of a data or a remote frame. The only action to be taken is signaling an overload condition. This means that an error in this bit field would be interpreted as an overload condition. Suspend transmission has to be inserted by error-passive nodes that were transmitter for the last message. This bit field has the form of eight ''recessive'' bits. However, it may be overwritten by a ''dominant'' start-bit from another non error passive node which starts transmission. The bus idle field consists of ''recessive'' bits. Its length is not specified and depends on the bus load.

ERROR FRAME

The Error Frame consists of two bit fields: the error flag and the error delimiter. The error field is built up from the various error flags of the different nodes. Therefore, its length may vary from a minimum of six bits up to a maximum of twelve bits depending on when a module detects the error. Whenever a bit error, stuff error, form error, or acknowledgment error is detected by a node, this node starts transmission of the error flag at the next bit. If a CRC error is detected, transmission of the error flag starts at the bit following the acknowledge delimiter, unless an error flag for a previous error condition has already been started. *Figure 25* shows how a local fault at one module (module 2) leads to a 12-bit error frame on the bus.

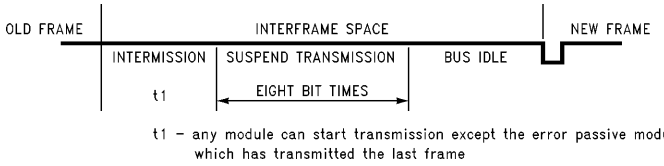
The bus level may either be ''dominant'' for an error-active node or ''recessive'' for an error-passive node. An error active node detecting an error, starts transmitting an active error flag consisting of six ''dominant'' bits. This causes the destruction of the actual frame on the bus. The other nodes detect the error flag as either a violation of the rule of bit-stuffing or the value of a fixed bit field is destroyed. As a consequence all other nodes start transmission of their own error flag. This means, that the error sequence which can be monitored on the bus as a maximum length of twelve bits. If an error passive node detects an error it transmits six ''recessive'' bits on the bus. This sequence does not destroy a message sent by another node and is not detected by other nodes. However, if the node detecting an error was the transmitter of the frame the other modules will get an error condition by a violation of the fixed bit or stuff rule. *Figure 24* shows how an error passive transmitter transmits a passive error frame and when it is detected by the receivers.

After any module has transmitted its active or passive error flag it waits for the error delimiter which consists of eight ''recessive'' bits before continuing.



TL/DD/12067–60

FIGURE 23. Interframe Space for Nodes Which Are Not Error Passive or Have Been Receiver for the Last Frame



TL/DD/12067–61

FIGURE 24. Interframe Space for Nodes Which Are Error Passive and Have Been Transmitter for the Last Frame

Basic CAN Concepts (Continued)

OVERLOAD FRAME

Like an error frame, an overload frame consists of two bit fields: the overload flag and the overload delimiter. The bit fields have the same length as the error frame field: six bits for the overload flag and eight bits for the delimiter. The overload frame can only be sent after the end of frame (EOF) field and in they way destroys the fixed form of the intermission field.

ORDER OF BIT TRANSMISSION

A frame is transmitted starting with the Start of Frame, sequentially followed by the remaining bit fields. In every bit field the MSB is transmitted first.

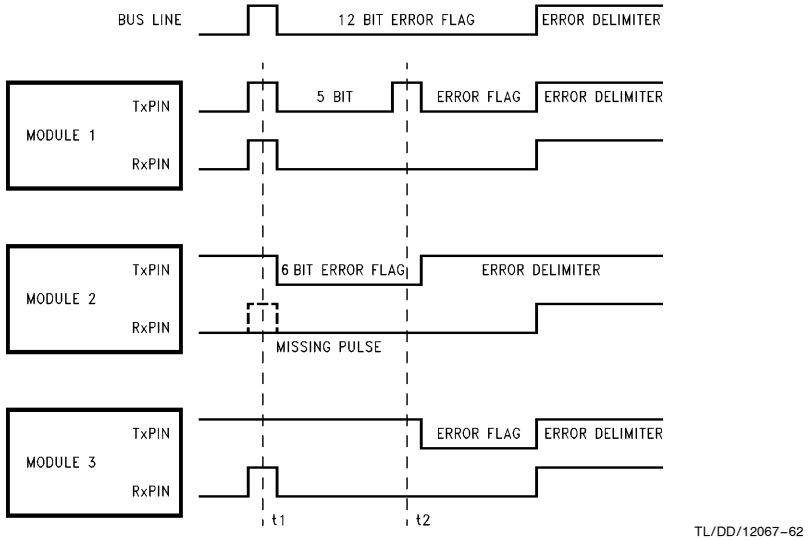
FRAME VALIDATION

Frames have a different validation point for transmitters and receivers. A frame is valid for the transmitter of a message, if there is no error until the end of the last bit of the End of Frame field. A frame is valid for a receiver, if there is no error until and including the end of the penultimate bit of the End of Frame.

FRAME ARBITRATION AND PRIORITY

Except for an error passive node which transmitted the last frame, all nodes are allowed to start transmission of a frame after the intermission, which can lead to two or more nodes starting transmission at the same time. To prevent a node from destroying another node's frame, it monitors the bus during transmission of the identifier field and the RTR-bit. As soon as it detects a ''dominant'' bit while transmitting a ''recessive'' bit it releases the bus, immediately stops transmission and starts receiving the frame. This causes no data or remote frame to be destroyed by another. Therefore the highest priority message with the identifier 0x000 out of 0x7EF (including the remote data request (RTR) bit) always gets the bus. This is only valid for standard CAN frame format. Note that while the CAN specification allows valid standard identifiers only in the range 0x000 to 0x7EF, the device will allow identifiers to 0x7FF.

There are three more items that should be taken into consideration to avoid unrecoverable collisions on the bus:



TL/DD/12067–62

module 1 = error active transmitter detects bit error at t2
module 2 = error active receiver with a local fault at t1
module 3 = error active receiver detects stuff error at t2

FIGURE 25. Error Frame—Error Active Transmitter

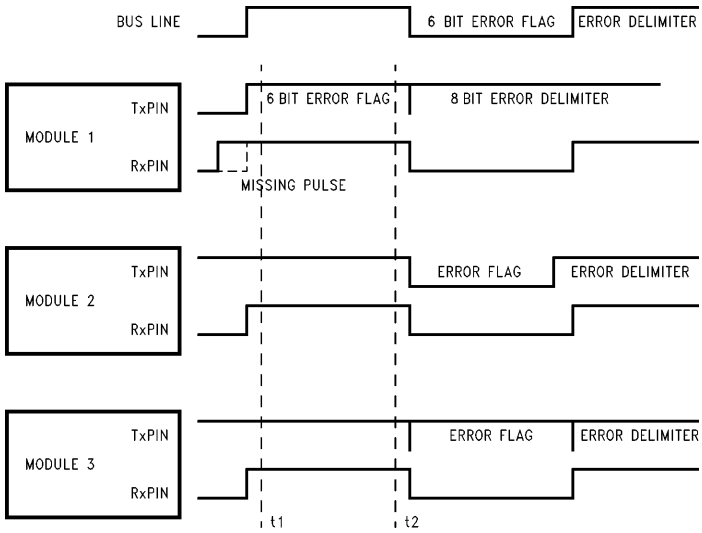
- Within one system each message must be assigned a unique identifier. This is to prevent bit errors, as one module may transmit a ''dominant'' data bit while the other is transmitting a ''recessive'' data bit. This could happen if two or more modules start transmission of a frame at the same time and all win arbitration.
- Data frames with a given identifier and a non-zero data length code may be initiated by one node only. Otherwise, in worst case, two nodes would count up to the bus-off state, due to bit errors, if they always start transmitting the same ID with different data.
- Every remote frame should have a system-wide data length code (DLC). Otherwise two modules starting transmission of a remote frame at the same time will overwrite each other's DLC which result in bit errors.

ACCEPTANCE FILTERING

Every node may perform acceptance filtering on the identifier of a data or a remote frame to filter out the messages which are not required by the node. In they way only the data of frames which match the acceptance filter is stored in the corresponding data buffers. However, every node which is not in the bus-off state and has received a correct CRC-sequence acknowledges each frame.

ERROR MANAGEMENT AND DETECTION

There are multiple mechanisms in the CAN protocol, to detect errors and to inhibit erroneous modules from disabling all bus activities.

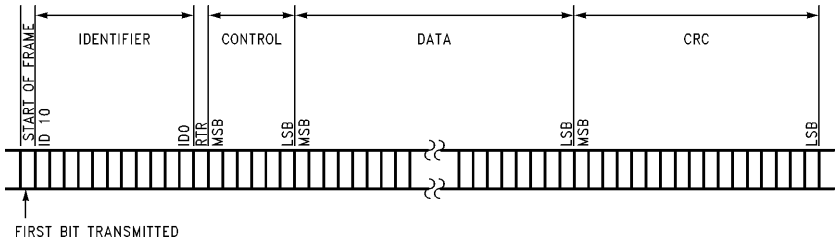


TL/DD/12067–63

module 1 = error active receiver with a local fault at t1
module 2 = error passive transmitter detects bit error at t2
module 3 = error passive receiver detects stuff error at t2

FIGURE 26. Error Frame—Error Passive Transmitter

Basic CAN Concepts (Continued)



TL/DD/12067–64

FIGURE 27. Order of Bit Transmission within a CAN Frame

The following errors can be detected:

- Bit Error
  A CAN device that is sending also monitors the bus. If the monitored bit value is different from the bit value that is sent, a bit error is detected. The reception of a ''dominant'' bit instead of a ''recessive'' bit during the transmission of a passive error flag, during the stuffed bit stream of the arbitration field or during the acknowledge slot, is not interpreted as a bit error.
- Stuff error
  A stuff error is detected, if the bit level after 6 consecutive bit times has not changed in a message field that has to be coded according to the bit stuffing method.
- Form Error
  A form error is detected, if a fixed frame bit (e.g., CRC delimiter, ACK delimiter) does not have the specified value. For a receiver a ''dominant'' bit during the last bit of End of Frame does NOT constitute a form error.
- Bit CRC Error
  A CRC error is detected if the remainder of the CRC calculation of a received CRC polynomial is non-zero.
- Acknowledgment Error
  An acknowledgment error is detected whenever a transmitting node does not get an acknowledgment from any other node (i.e., when the transmitter does not receive a ''dominant'' bit during the ACK frame).

The device can be in one of three states with respect to error handling:

- Error active
  An error active unit can participate in bus communication and sends an active (''dominant'') error flag.
- Error passive
  An error passive unit can participate in bus communication. However, if the unit detects an error it is not allowed to send an active error flag. The unit sends only a passive (''recessive'') error flag. A device is error passive when the transmit error counter is greater than 127 or when the receive error counter is greater than 127. A device becoming error passive sends an active error flag. An error passive device becomes error active again when both transmit and receive error counter are less than 128.
- Bus off
  A unit that is ''bus off'' has the output drivers disabled, i.e., it does not participate in any bus activity. A device is bus off when the transmit error counter is greater than 255. A bus off device will become error active again in one of two ways depending on which mode is selected by the user through the Fault Confinement Mode select bit (FMOD) in the CAN Bus Control Register (CBUS). Setting the FMOD bit to ''0'' (default after power on reset) will select the Standard Fault Confinement mode. In this mode the device goes from ''bus off'' to ''error active'' after monitoring 128*11 recessive bits (including bus idle) on the bus. This mode has been implemented for compatibility reasons with existing solutions. Setting the FMOD bit to ''1'' will select the Enhanced Fault Confinement mode. In this mode the device goes from ''bus off'' to ''error active'' after monitoring 128 ''good'' messages, as indicated by the reception of 11 consecutive ''recessive'' bits including the End of Frame. The enhanced mode offers the advantage that a ''bus off'' device (i.e., a device with a serious fault) is not allowed to destroy any messages on the bus until other devices can transmit at least 128 messages. This is not guaranteed in the standard mode, where a defective device could seriously impact bus communication. When the device goes from ''bus off'' to ''error active'', both error counters will have the value ''0''.

In each CAN module there are two error counters to perform a sophisticated error management. The receive error counter (REC) is 7 bits wide and switches the device to the error passive state if it overflows. The transmit error counter (TEC) is 8 bits wide. If it is greater than 127, the device is switched to the error passive state. As soon as the TEC overflows, the device is switched bus-off, i.e., it does not participate in any bus activity.

Basic CAN Concepts (Continued)

The counters are modified by the device's hardware according to the following rules:

TABLE X. Receive Error Counter Handling

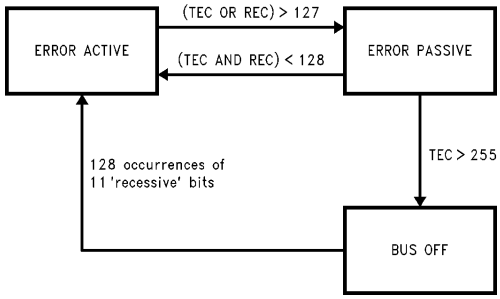
Condition	Receive Error Counter
A receiver detects a Bit Error during sending an active error flag.	Increment by 8
A receiver detects a ''dominant'' bit as the first bit after sending an error flag.	Increment by 8
After detecting the 14th consecutive ''dominant'' bit following an active error flag or overload flag or after detecting the 8th consecutive ''dominant'' bit following a passive error flag. After each sequence of additional 8 consecutive ''dominant'' bits.	Increment by 8
Any other error condition (stuff, frame, CRC, ACK).	Increment by 1
A valid reception or transmission.	Decrement by 1 if Counter is not 0

TABLE XI. Transmit Error Counter Handling

Condition	Transmit Error Counter
A transmitter detects a Bit Error during sending an active error flag.	Increment by 8
After detecting the 14th consecutive ''dominant'' bit following an active error flag or overload flag or after detecting the 8th consecutive ''dominant'' bit following a passive error flag. After each sequence of additional 8 consecutive ''dominant'' bits.	Increment by 8
Any other error condition (stuff, frame, CRC, ACK).	Increment by 8
A valid reception or transmission.	Decrement by 1 if Counter is not 0

Special error handling for the TEC counter is performed in the following situations:

- A stuff error occurs during arbitration, when a transmitted ''recessive'' stuff bit is received as a ''dorminant'' bit. This does not lead to an incrementation of the TEC.
- An ACK-error occurs in an error passive device and no ''dominant'' bits are detected while sending the passive error flag. This does not lead to an incrementation of the TEC.
- If only one device is on the bus and this device transmits a message, it will get no acknowledgment. This will be detected as an error and message will be repeated. When the device goes ''error passive'' and detects an acknowledge error, the TEC counter is not incremented. Therefore the device will not go from ''error passive'' to the ''bus off'' state due to such a condition.



TL/DD/12067–65

FIGURE 28. CAN Bus States

Figure 28 shows the connection of different bus states according to the error counters.

SYNCHRONIZATION

Every receiver starts with a ''hard synchronization'' on the falling edge of the SOF bit. One bit time consists of four bit segments: Synchronization segment, propagation segment, phase segment 1 and phase segment 2.

A falling edge of the data signal should be in the synchronization segment. This segment has the fixed length of one time quanta. To compensate for the various delays within a network, the propagation segment is used. Its length is programmable from 1 to 8 time quanta. Phase segment 1 and phase segment 2 are used to resynchronize during an active frame. The length of these segments is from 1 to 8 time quanta long.

Two types of synchronization are supported:

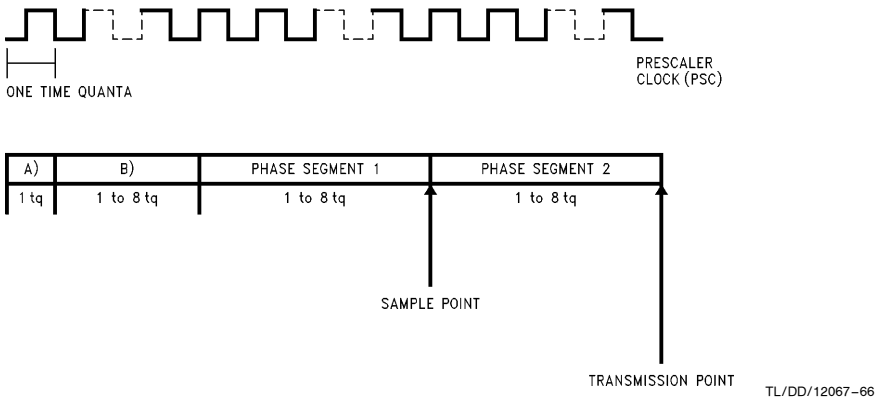
Hard synchronization is done with the falling edge on the bus while the bus is idle, which is then interpreted as the SOF. It restarts the internal logic.

Soft synchronization is used to lengthen or shorten the bit time while a data or remote frame is received. Whenever a falling edge is detected in the propagation segment or in phase segment 1, the segment is lengthened by a specific value, the resynchronization jump width (see *Figure 30*).

If a falling edge lies in the phase segment 2 (as shown in *Figure 30*) it is shortened by the resynchronization jump width. Only one resynchronization is allowed during one bit time. The sample point lies between the two phase segments and is the point where the received data is supposed to be valid. The transmission point lies at the end of phase segment 2 to start a new bit time with the synchronization segment.

Note 1: The resynchronization jump width (RJW) is automatically determined from the programmed value of PS. If a soft resynchronization is done during phase segment 1 or the propagation segment, then RJW will either be equal to 4 internal CAN clocks (CKI/(1 + *divider)*) or the programmed value of PS, whichever is less. PS2 will never be shorter than 1 internal CAN clock.

Note 2: (PS1—BTL settings any PSC setting) The PS1 of the BTL should always be programmed to values greater than 1. To allow device resynchronization for positive and negative phase errors on the bus. (if PS1 is programmed to one, a bit time could only be lengthened and never shortened which basically disables half of the synchronization).



A) Synchronization segment
B) Propagation segment

FIGURE 29. Bit Timing

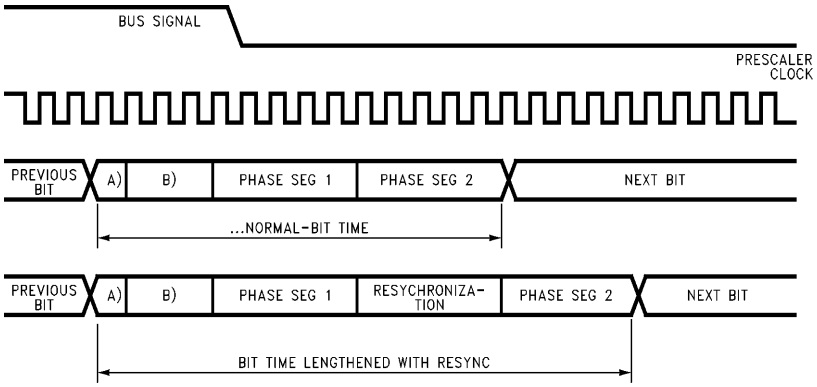


FIGURE 30. Resynchronization 1

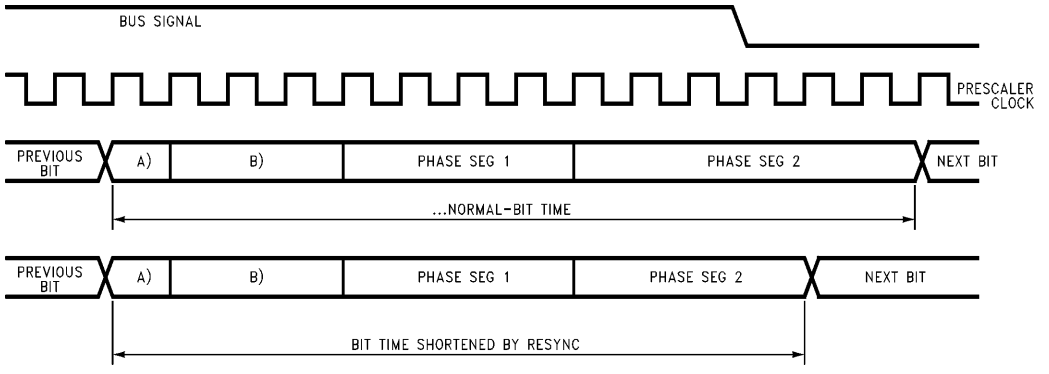


FIGURE 31. Resynchronization 2

Comparators

The device has two differential comparators. Port L is used for the comparators. The output of the comparators is multiplexed out to two pins. The following are the Port L assignments:

L0 Comparator 1 positive input

L1 Comparator 1 negative input

L2 Comparator 1 output

L3 Comparator 2 negative input

L4 Comparator 2 positive input

L5 Comparator 2 negative input

L6 Comparator 2 output

Additionally the comparator output can be connected internally to the L-Port pin of the respective positive input and thereby generate an interrupt using the L-Port interrupt structure (neg/pos. edge, enable/disable).

Note that in *Figure 32*, pin L6 has a second alternate function of supporting the PWM0 output. The comparator 2 output MUST be disabled in order to use PWM0 output on L6.

Figure 32 shows the Comparator Block Diagram.

COMPARATOR CONTROL REGISTER (CMPLSS) (00D3)

These bits reside in the Comparator Register

CMP2 SEL	CMP2 OE	CMP2 RD	CMP2 EN	CMP1 OE	CMP1 RD	CMP1 EN	un- used

Bit 7 Bit 0

The register contains the following bits:

CMP1EN Enables comparator 1 (''1'' = enable). If comparator 1 is disabled the associated L-pins can be used as standard I/O.

CMP1RD Reads comparator 1 output internally (CMP1EN = 1) Read-only, reads as a ''0'' if comparator not enabled.

CMP1OE Enables comparator 1 output (''1'' = enable), CMP1EN bit must be set to enable this function.

CMP2EN Enables comparator 2 (''1'' = enable). If comparator 2 is disabled the associated L-pins can be used as standard 1/O.

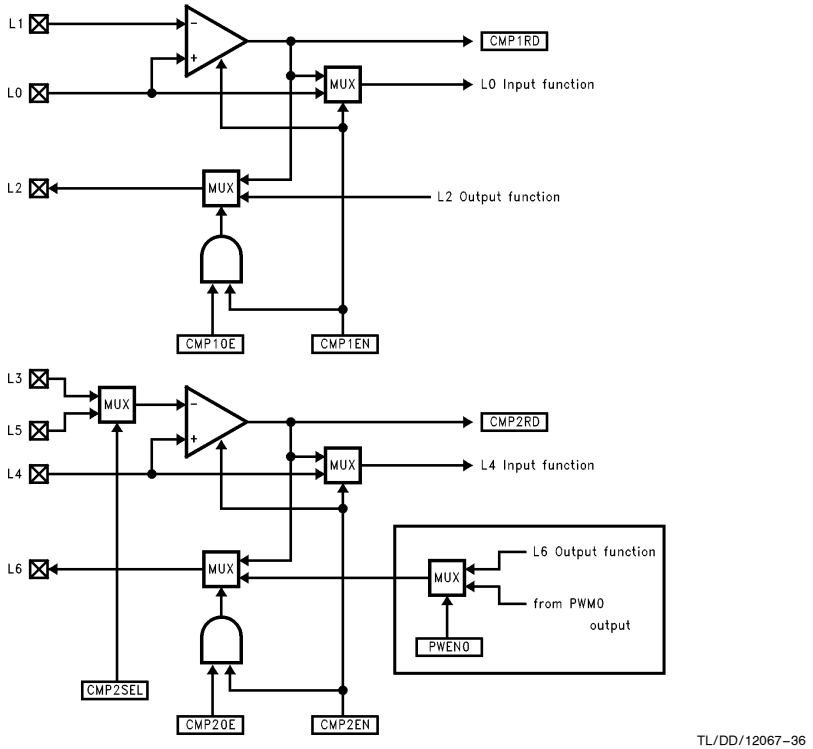
CMP2RD Reads comparator 2 output internally (CMP2EN = 1) Read-only, reads as a ''0'' if comparator not enabled.

CMP2OE Enables comparator 2 output (''1'' = enable), CMP2EN bit must be set to enable this function.

CMP2SEL Selects which L port pin to use for comparator2 negative input. (CMP2SEL = 0 selects L5; CMP2SEL = 1 selects pin L3).

The Comparator Select/Control bits are cleared on RESET (the comparator is disabled). To save power, the program should also disable the comparator before the device enters the HALT mode.

The Comparator rise and fall times are symmetrical. The user program must set up the Configuration and Data registers of the L port correctly for comparator Inputs/Output.



TL/DD/12067–36

Note: The BOXED area shows logic from PWM Timer. Comparator 2 output (CMP2OE) must be disabled in order to use PWM0 output.

FIGURE 32. Comparator Block

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 zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 02F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 030 and 031 Hex (which are undefined RAM). Undefined RAM from addresses 030 to 03F Hex is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

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

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

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 33* 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 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 XI details the different clock rates that may be selected.

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 34* shows how two COP888 family microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

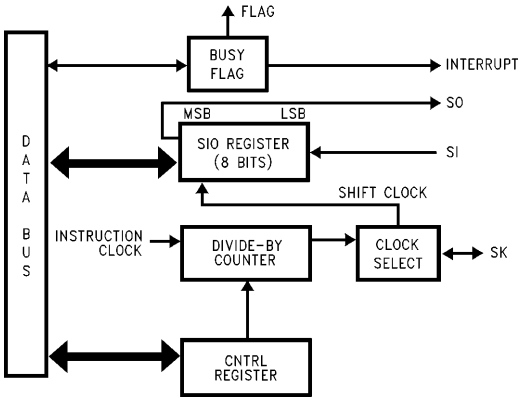
The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

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

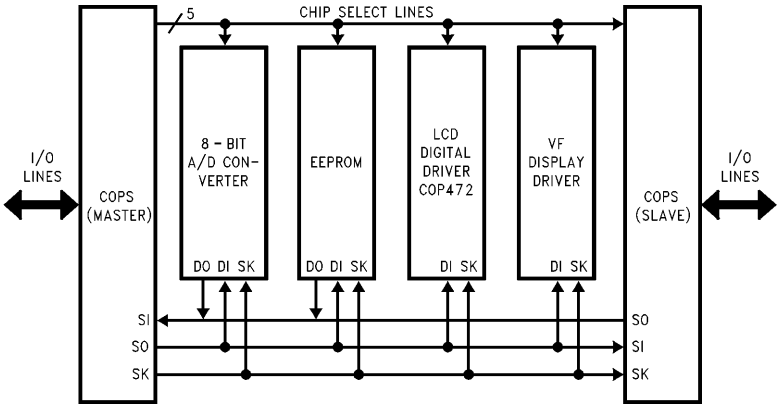
In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table XII summarizes the bit settings required for Master or Slave mode of operation.

MICROWIRE/PLUS (Continued)



TL/DD/12067–37

FIGURE 33. MICROWIRE/PLUS Block Diagram



TL/DD/12067–38

FIGURE 34. MICROWIRE/PLUS Application

MICROWIRE/PLUS (Continued)

TABLE XI. MICROWIRE/PLUS Master Mode Clock Selection

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

Where t_c is the instruction cycle clock

MICROWIRE/PLUS 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 bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

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

Alternate SK Phase Operation

The device allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

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

TABLE XII. MICROWIRE/PLUS Mode Selection

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

This table assumes that the control flag MSEL is set.

Memory Map

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

Address	Contents
00 to 2F	On-Chip RAM bytes (48 bytes)
30 to 7F	Unused RAM Address Space (Reads As All Ones)
80 to 9F	Unused RAM Address Space (Reads Undefined Data)
A0	PSCAL, PWM timer Prescaler Register
A1	RLON, PWM timer On-Time Register
A2	PWMCON, PWM Control Register
B0	TXD1, Transmit 1 Data
B1	TXD2, Transmit 2 Data
B2	TDLC, Transmit Data Length Code and Identifier Low
B3	TID, Transmit Identifier High
B4	RXD1, Receive Data 1
B5	RXD2, Receive Data 2
B6	RIDL, Receive Data Length Code
B7	RID, Receive Identify High
B8	CSCAL, CAN Prescaler
B9	CTIM, Bus Timing Register
BA	CBUS, Bus Control Register
BB	TCNTL, Transmit/Receive Control Register
BC	RTSTAT Receive/Transmit Status Register
BD	TEC, Transmit Error Count Register
BE	REC, Receive Error Count Register
BF	Reserved
C0 to C7	Reserved
C8	WKEDG, MIWU Edge Select Register
C9	WKEN, MIWU Enable Register
CA	WKPND, MIWU Pending Register
CB	Reserved
CC	Reserved
CD to CF	Reserved

Memory Map (Continued)

Address	Contents
D0	PORTLD, Port L Data Register
D1	PORTLC, Port L Configuration Register
D2	PORTLP, Port L Input Pins (Read Only)
D3	CMPSL, Comparator control register
D4	PORTGD, Port G Data Register
D5	PORTGC, Port G Configuration Register
D6	PORTGP, Port G Input Pins (Read Only)
D7 to DB	Reserved
DC	PORTD, Port D output register
DD to DF	Reserved for Port D
E0–E5	Reserved
E6	T1RBLO, Timer T1 Autoload Register Lower Byte
E7	T1RBHI, Timer T1 Autoload Register Upper Byte
E8	ICNTRL, Interrupt Control Register
E9	SIOR, MICROWIRE/PLUS Shift Register
EA	TMR1LO, Timer T1 Lower Byte
EB	TMR1HI, Timer T1 Upper Byte
EC	T1RALO, Timer T1 Autoload Register Lower Byte
ED	T1RAHI, Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL, Control Register
EF	PSW, Processor Status Word Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved (Note A)

Note: Reading memory locations 30–7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Note A: In devices with more than 128 bytes of RAM, location 0FF is used as the Segment register to switch between different Segments of RAM memory. In this device location 0FF can be used as a general purpose, on-chip RAM mapped register. However, the user is advised that caution should be taken in porting software utilizing this memory location to a chip with more than 128 bytes of RAM.

Addressing Modes

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

OPERAND ADDRESSING MODES

Register Indirect

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

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

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

Direct

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

Immediate

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

Short Immediate

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

Indirect

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

TRANSFER OF CONTROL ADDRESSING MODES

Relative

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

Absolute

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

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location up to 32k 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]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A ← A + Meml
ADC	A,Meml	ADD with Carry	A ← A + Meml + C, C ← Carry, HC ← Half Carry
SUBC	A,Meml	Subtract with Carry	A ← A − Meml + C, C ← Carry, HC ← Half Carry
AND	A,Meml	Logical AND	A ← A and $\overline{\text{Meml}}$
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A ← A or Meml
XOR	A,Meml	Logical EXclusive OR	A ← A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A ≠ Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg − 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ⟷ Mem
X	A,[X]	EXchange A with Memory [X]	A ⟷ [X]
LD	A,Meml	LoaD A with Memory	A ← Meml
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imm	LoaD B with Immed.	B ← Imm
LD	Mem,Imm	LoaD Memory Immed.	Mem ← Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg ← Imm
X	A, [B±]	EXchange A with Memory [B]	A ⟷ [B], (B ← B ± 1)
X	A, [X±]	EXchange A with Memory [X]	A ⟷ [X], (X ← ± 1)
LD	A, [B±]	LoaD A with Memory [B]	A ← [B], (B ← B ± 1)
LD	A, [X±]	LoaD A with Memory [X]	A ← [X], (X ← X ± 1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] ← Imm, (B ← B ± 1)
CLR	A	CLeaR A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrementA	A ← A − 1
LAID		Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORrect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C → A7 → . . . → A0 → C
RLC	A	Rotate A Left thru C	C ← A7 ← . . . ← A0 ← C
SWAP	A	SWAP nibbles of A	A7 . . . A4 ⟷ A3 . . . A0
SC		Set C	C ← 1, HC ← 1
RC		Reset C	C ← 0, HC ← 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP − 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0k 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]
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 and 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				2/3			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

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

Opcode Table

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

where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

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

Mask Options

The COP684BC and COP884BC mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

= 1 Crystal Oscillator (CKI/10)

G7 (CKO) is clock generator output to crystal/resonator

CKI is the clock input

OPTION 2: HALT

= 1 Enable HALT mode

= 2 Disable HALT mode

OPTION 3: BOND OUT

= 1 28-Pin SO

OPTION 4: ON-CHIP POWER-ON RESET

= 1 Enable ON-CHIP POWER-ON RESET

= 2 Disable ON-CHIP POWER-ON RESET

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. The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

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-COP888GG—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 development and marketed by MetaLink Corporation to support the whole COP8 family of products. National is a resale vendor for these products.

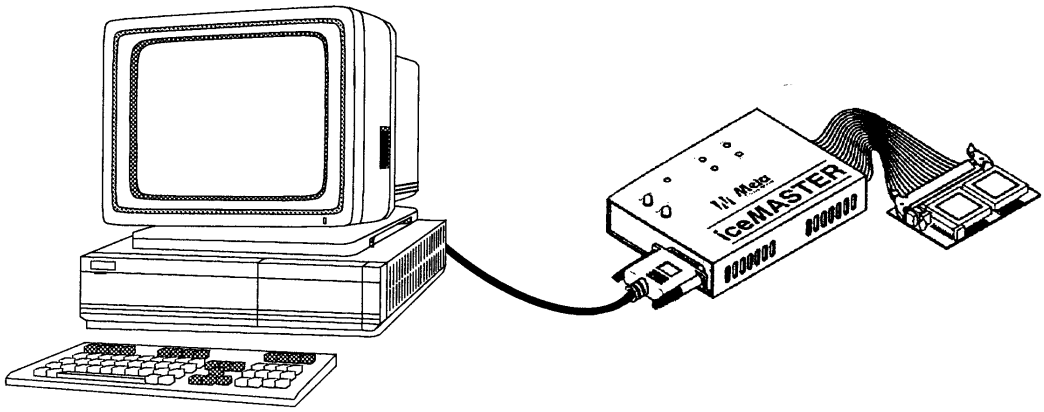
See *Figure 35* for configuration.

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 mount assembly.
- Full 32 kbytes 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 eight 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-888BC28P5PC	28 DIP
28 DIP to 28 SO Adapter	
DM-SOIC28	28 SO



TL/DD/12067–70

FIGURE 35. COP8 iceMASTER Environment

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.

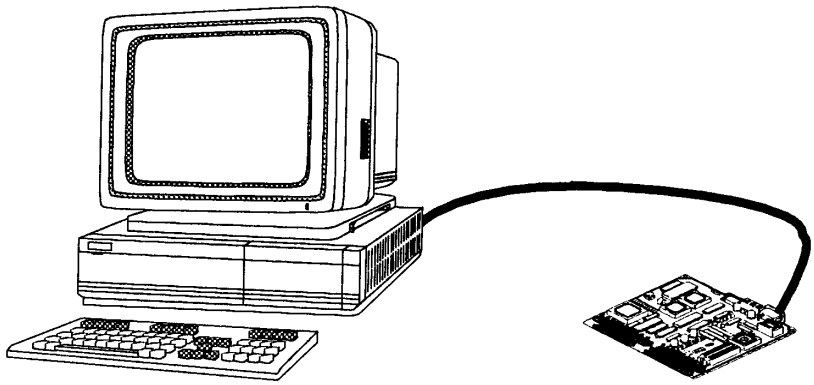
See *Figure 36* for configuration.

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 kbytes 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/888BC	
Cable Adapters	
DM-COP8/28D	28 DIP
28 DIP to 28 SO Adapter	
DM-SOIC28	28 SO



TL/DD/12067–71

FIGURE 36. COP8-DM Environment

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 optiomization of linked code.
- Symbolic debug load format fully source level supported by the MetaLink debugger.

SINGLE CHIP EMULATOR SUPPORT

The COP8 family is fully supported by single chip form, fit and function emulators. For more detailed information refer to the emulation device specific datasheets and the form, fit, function emulator selection table below.

OTP Ordering Information

Device Number	Clock Option	Package	Emulates
COP87L84BC	Crystal	28 SO	COP884BC

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.

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)

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) 0-814-135 13 32
	Baud:	14.4k
	Set-up:	Length: 8-Bit
		Parity: None
		Stop Bit: 1
	Operation:	24 Hrs., 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

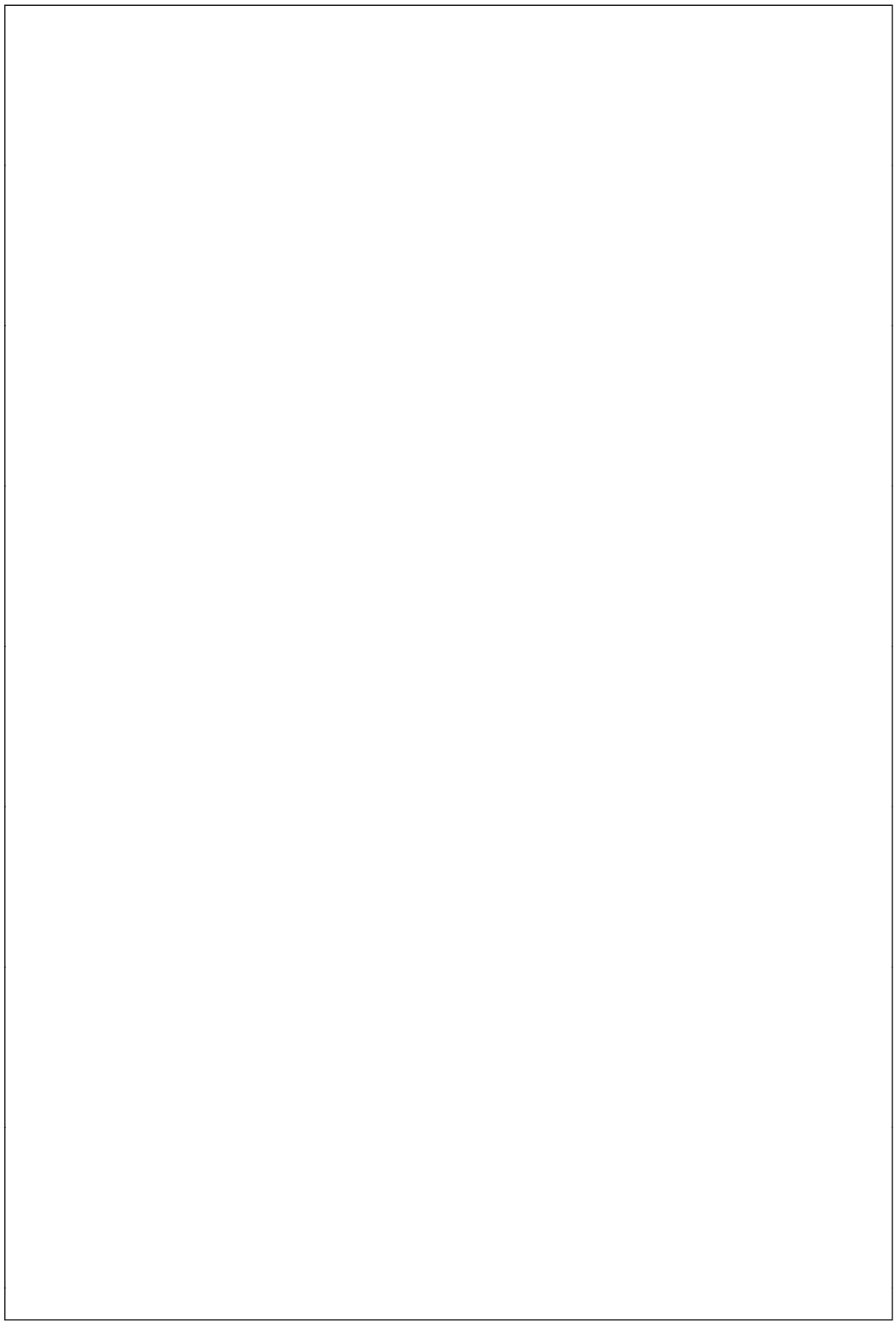
National Semiconductor on the WorldWide Web

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

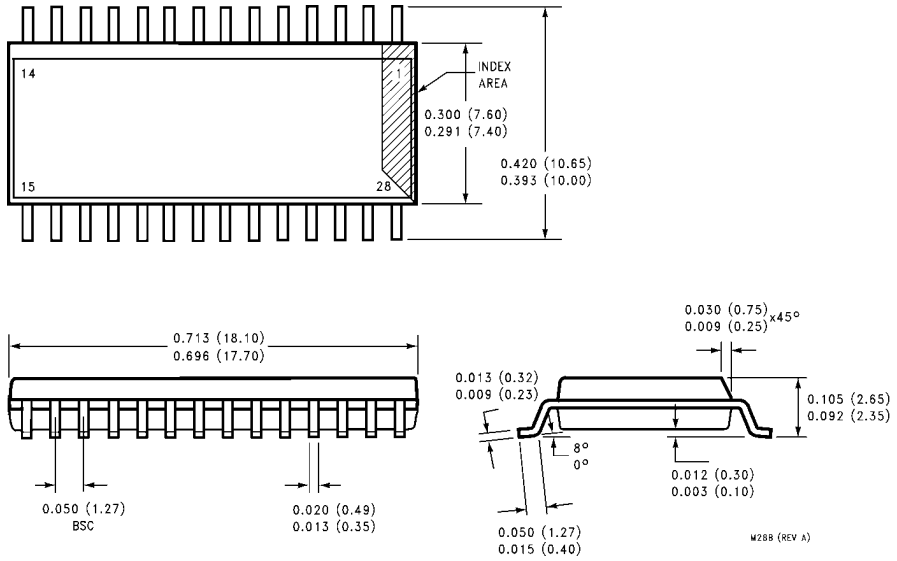
CUSTOMER RESPONSE CENTER

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:		europe.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



Physical Dimensions inches (millimeters) unless otherwise noted



Order Number COP884BC-xxx/M or COP684BC-xxx/M
NS Package Number M28B

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 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
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

http://www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.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

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2308
Fax: 81-043-299-2408