National Semiconductor

September 1999

COP87L88EB/RB Family
8-Bit CMOS OTP Microcontrollers with 16k or 32k Memory, CAN Interface, 8-Bit A/D, and USART

General Description

The COP87L88EB/RB Family OTP (One Time programmable) microcontrollers are highly integrated COP8™ Feature core devices with 16k or 32k memory and advanced features including a CAN 2.0B (passive) interface, A/D and USART. These multi-chip CMOS devices are suited for applications requiring a full featured controller with a CAN interface, low EMI, and versatile communications interfaces, and as pre-production devices for ROM designs. Pin and software compatible 8k ROM versions (COP888EB) are available as well as a range of COP8 software and hardware development tools.

Features include an 8-bit memory mapped architecture, 10 MHz CKI (-XE = crystal oscillator) with 1µs instruction cycle, two multi-function 16-bit timer/counters, WATCHDOG and clock monitor, idle timer, CAN 2.0B (passive) interface, MICROWIRE/PLUS™ serial I/O, SPI master/slave interface, fully buffered USART, 8 bit A/D with 8 channels, two power saving HALT/IDLE modes, MIWU, software selectable I/O options, low EMI 4.5V to 5.5V operation, program code security, and 44/68 pin packages.

Note: A companion device with CAN interface, less I/O and memory, A/D, and PWM timer is the COP87L84BC.

Devices included in this datasheet are:

Device	Memory (bytes)	RAM (bytes)	I/O Pins	Packages	Temperature
COP87L88EB	16k OTP EPROM	192	35	44 PLCC	-40 to +85˚C
COP87L89EB	16k OTP EPROM	192	58	68 PLCC	-40 to +85˚C
COP87L88RB	32k OTP EPROM	192	35	44 PLCC	-40 to +85˚C
COP87L89RB	32k OTP EPROM	192	58	68 PLCC	-40 to +85˚C

Key Features

- CAN 2.0B (passive) bus interface, with Software Power save mode
- 8-bit A/D Converter with 8 channels
- Fully buffered USART
- Multi-input wake up (MIWU) on both Port L and M
- SPI Compatible Master/Slave Interface
- 16 or 32 kbytes of on-board OTP EPROM with security feature

  Note: Mask ROMed device with equivalent on-chip features and program memory size of 8k is available.
- 192 bytes of on-board RAM

Additional Peripheral Features

- Idle timer (programmable)
- Two 16-bit timer, with two 16-bit registers supporting
  - — Processor independent PWM mode
  - — External Event counter mode
  - — Input capture mode
- WATCHDOG™ and Clock Monitor
- MICROWIRE/PLUS serial I/O

I/O Features

- Software selectable I/O options (TRI-STATE® outputs, Push pull outputs, Weak pull up input, High impedance input)
- Schmitt trigger inputs on Port G, L and M
- Packages: 44 PLCC with 35 I/O pins; 68 PLCC with 58 I/O pins

CPU/Instruction Set Features

- 1 µs instruction cycle time
- Fourteen multi-sourced vectored interrupts servicing
  - — External interrupt
  - — Idle Timer T0
  - — Timers (T1 and T2) (4 Interrupts)
  - — MICROWIRE/PLUS and SPI
  - — Multi-input Wake up
  - — Software Trap
  - — CAN interface (3 interrupts)
  - — USART (2 Inputs)
- Versatile easy to use instruction set
- 8-bit stacker pointer (SP) (Stack in RAM)
- Two 8-bit RegisterR Indirect Memory Pointers (B, X)

Fully Static CMOS

- Two power saving modes: HALT, IDLE
- Single supply operation: 4.5V to 5.5V
- Temperature range: −40˚C to +85˚C

Development Support

- Emulation device for COP888EB
- Real time emulation and full program debug offered by MetaLink Development System

Basic Functional Description

- CAN I/F — CAN serial bus interface block as described in the CAN specification part 2.0B (Passive)
  - — Interface rates up to 250k bit/s are supported utilizing standard message identifiers
- Programmable double buffered USART
- A/D — 8-bit, 8 channel, 1-LSB Resolution, with improved Source Impedance and improved channel to channel cross talk immunity
- Multi-Input-Wake-Up (MIWU) — edge selectable wake-up and interrupt capability via input port and CAN interface (Port L, Port M and CAN I/F); supports Wake-Up capability on SPI, USART, and T2
- Port C — 8-bit bi-directional I/O port
- Port D — 8-bit Output port with high current drive capability (10 mA)
- Port F — 8-bit bidirectional I/O
- Port G — 8-bit bidirectional I/O port, including alternate functions for:
  - — MICROWIRE™ Input and Output
  - — Timer 1 Input or Output (Depending on mode selected)
  - — External Interrupt input
  - — WATCHDOG Output
- Port I — 8-bit input port combining either digital input, or up to eight A/D input channels
- Port L — 8-bit bidirectional I/O port, including alternate functions for:
  - — USART Transmit/Receive I/O
  - — Multi-input-wake up (MIWU on all pins)
- Port M — 8-bit I/O port, with the following alternate function
  - — SPI Interface
  - — MIWU
  - — CAN Interface Wake-up (MSB)
  - — Timer 2 Input or Output (Depending on mode selected)
- Port N — 8-bit bidirectional I/O
  - — SPI Slave Select Expander
- Two 16-bit multi-function Timer counters (T1 and T2) plus supporting registers
  - — (I/P Capture, PWM and Event Counting)
- Idle timer — Provides a basic time-base counter, (with interrupt) and automatic wake up from IDLE mode programmable
- MICROWIRE/PLUS — MICROWIRE serial peripheral interface, supporting both Master and Slave operation
- HALT and IDLE — Software programmable low current modes
  - — HALT — Processor stopped, Minimum current
  - — IDLE — Processor semi-active more than 60% power saving
- 16 or 32 kbytes OTP EPROM and 192 bytes of on board static RAM
- SPI Master/Slave interface includes 12 bytes Transmit and 12 bytes Receive FIFO Buffers. Operates up to 1M Bit/S
- On board programmable WATCHDOG and CLOCK Monitor

Applications

- Automobile Body Control and Comfort System
- Integrated Driver Informaiton Systems
- Steering Wheel Control
- Car Radio Control Panel
- Sensor/Actuator Applications in Automotive and Industrial Control

Block Diagram

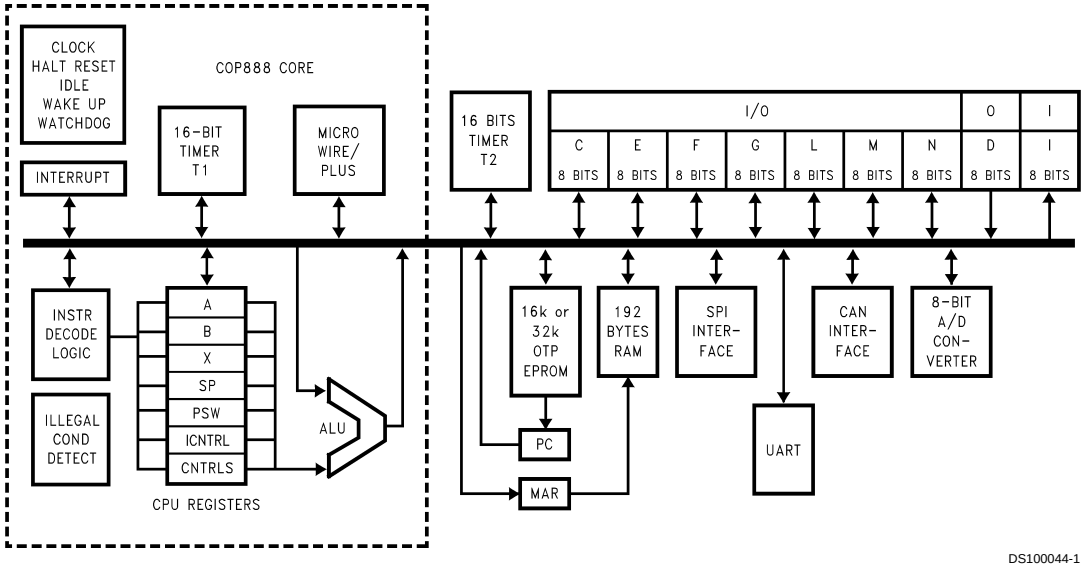
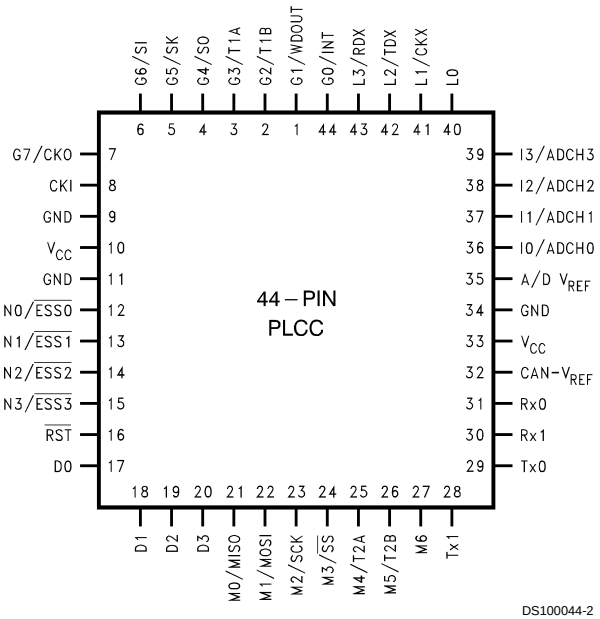


FIGURE 1. Block Diagram

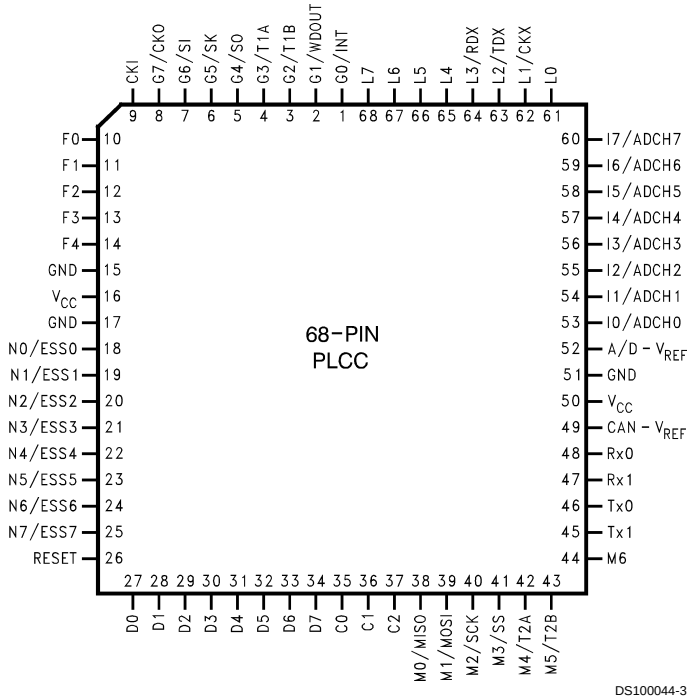
Connection Diagrams

Plastic Chip Carrier



Top View
Order Number COP87L88EBV-XE or COP87L88RBV-XE
See NS Plastic Chip Package Number V44A

Plastic Leaded Chip Carrier



Note:
-X Crystal Oscillator
-E Halt Mode Enabled

Top View
Order Number COP87L89EBV-XE or COP87L89RBV-XE
See NS Plastic Chip Package Number V68A

FIGURE 2. Connection Diagrams

Connection Diagrams (Continued)

Pinouts for 44-Pin and 68-Pin Packages

Port Pin	Type	ALT Function	44-Pin PLCC	68-Pin PLCC
G0	I/O	INT	44	1
G1	I/O	WDOUT	1	2
G2	I/O	T1B	2	3
G3	I/O	T1A	3	4
G4	I/O	SO	4	5
G5	I/O	SK	5	6
G6	I	SI	6	7
G7	I	CKO	7	8
D0	O		17	27
D1	O		18	28
D2	O		19	29
D3	O		20	30
D4	O			31
D5	O			32
D6	O			33
D7	O			34
I0	I	ADCH0	36	53
I1	I	ADCH1	37	54
I2	I	ADCH2	38	55
I3	I	ADCH3	39	56
I4	I	ADCH4		57
I5	I	ADCH5		58
I6	I	ADCH6		59
I7	I	ADCH7		60
L0	I/O	MIWU	40	61
L1	I/O	MIWU;CKX	41	62
L2	I/O	MIWU;TDX	42	63
L3	I/O	MIWU;RDX	43	64
L4	I/O	MIWU		65
L5	I/O	MIWU		66
L6	I/O	MIWU		67
L7	I/O	MIWU		68
M0	I/O	MIWU;MISO	21	38
M1	I/O	MIWU;MOSI	22	39
M2	I/O	MIWU;SCK	23	40
M3	I/O	MIWU;$\overline{SS}$	24	41
M4	I/O	MIWU;T2A	25	42
M5	I/O	MIWU;T2B	26	43
M6	I/O	MIWU	27	44
N0	I/O	ESS0	12	18
N1	I/O	ESS1	13	19
N2	I/O	ESS2	14	20
N3	I/O	ESS3	15	21
N4	I/O	ESS4		22
N5	I/O	ESS5		23
N6	I/O	ESS6		24
N7	I/O	ESS7		25
F0	I/O			10
F1	I/O			11
F2	I/O			12
F3	I/O			13
F4	I/O			14
C0	I/O			35
C1	I/O			36
C2	I/O			37
RX0	I		31	48
RX1	I		30	47
TX0	O		29	46
TX1	O		28	45
$CANV_{REF}$			32	49
CKI			8	9
RESET			16	26
DV_{CC}			10, 33	16, 50
GND			9, 11, 34	15, 17, 51
A/D V_{REF}			35	52

Absolute Maximum Ratings (Note 1)

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

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

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

DC Electrical Characteristics

−40˚C ≤ T_A ≤ +85˚C

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 2)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current	V_{CC} = 5.5V, t_c = 1 µs			16	mA
CKI = 10 MHz (Note 3)					
HALT Current (Notes 4, 5)	V_{CC} = 5.5V, CKI = 0 MHz		< 1		µA
IDLE Current (Note 5)	V_{CC} = 5.5V, t_c = 1 µs			5.5	mA
CKI = 10 MHz					
Input Levels (V_{IH}, V_{IL})					
Reset, CKI					
Logic High		0.8V_{CC}			V
Logic Low				0.2V_{CC}	V
All Other Inputs					
Logic High		0.7V_{CC}			V
Logic Low				0.2V_{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
Port G, L and M Input Hysteresis	(Note 8)		0.05V_{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
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		+5.0	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
Allowable Sink/Source Current per Pin					
D Outputs (sink)				15	mA
Tx0 (Sink) (Note 8)				30	mA
Tx1 (Source) (Note 8)				30	mA
All Other				3	mA
Maximum Input Current	Room Temp			±200	mA
without Latchup (Notes 6, 8)					
RAM Retention Voltage, V_r (Note 7)	500 ns Rise and Fall Time	2.0			V
Input Capacitance	(Note 8)			7	pF
Load Capacitance on D2				1000	pF

Note 2: Maxiumum rate of voltage change must be < 0.5V/ms

DC Electrical Characteristics (Continued)

Note 3: 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 4: The HALT mode will stop CKI from oscillating in the Crystal configurations. Halt test conditions: All inputs tied to V_{CC}; Port C, G, E, F, L, M and N I/Os configured as outputs and programmed low; D outputs programmed high. Parameter refers to HALT mode entered via setting bit 7 of the Port G data register. Part will pull up CKI during HALT in crystal clock mode. Both CAN main comparator and the CAN Wakeup comparator need to be disabled.

Note 5: . HALT and IDLE current specifications assume CAN block comparators are disabled.

Note 6: Pins G6 and $\overline{\text{RESET}}$ are designed with a high voltage input network. These pins allow input voltages greater than V_{CC} and the pins will have sink current to V_{CC} when biased at voltages greater than V_{CC} (the pins do not have source current when biased at a voltage below V_{CC}). The effective resistance to V_{CC} is 750Ω (typical). These two pins will not latch up. The voltage at the pins must be limited to less than 14V.

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

Note 8: Parameter characterized but not tested.

AC Electrical Characteristics

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

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal/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
Output Propagation Delay (t_{PD1}, t_{PD0}) (Note 9)	C_L = 100 pF, R_L = 2.2 kΩ				
SK, SO	$V_{CC} \ge 4.5V$			0.7	µs
All others	$V_{CC} \ge 4.5V$			1	µs
MICROWIRE					
Setup Time (tUWS) (Note 10)		20			ns
Hold Time (tUWH) (Note 10)		56			ns
Output Pop Delay (tUPD)				220	ns
Input Pulse Width					
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 10)		1.0			µs

t_c = Instruction Cycle Time

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 1M 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 bits/s with eight byte messages and a 10 MHz oscillator.

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

Note 9: The output propagation delay is referenced to the end of the instruction cycle where the output change occurs.

Note 10: Parameter not tested.

On-Chip Voltage Reference

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

Parameter	Conditions	Min	Max	Units
Reference Voltage V_{REF}	I_{OUT} < 80 µA, V_{CC} = 5V	$0.5V_{CC}$ −0.12	$0.5V_{CC}$ +0.12	V
Reference Supply Current, I_{DD}	I_{OUT} = 0A, (No Load) V_{CC} = 5V (Note 11)		120	µA

Note 11: Reference supply I_{DD} is supplied for information purposes only, it is not tested.

CAN Comparator DC and AC Characteristics

$4.8V \le V_{CC} \le 5.2V$, $-40^{\circ}C \le T_A \le +85^{\circ}C$

Parameter	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

A/D Converter Specifications

$(4.5V \le V_{CC} \le 5.5V)$ $(V_{SS} - 0.050V) \le$ Any Input $\le (V_{CC} + 0.050V)$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Absolute Accuracy	$V_{REF} = V_{CC}$			±2	LSB
Non-Linearity Deviation from the Best Straight Line				±1	LSB
Differential Non-Linearity				±1	LSB
Common Mode Input Range (Note 14)		GND		V_{CC}	V
DC Common Mode Error				±0.5	LSB
Off Channel Leakage Current			1	2.0	µA
On Channel Leakage Current			1	2.0	µA
A/D Clock Frequency (Note 13)		0.1		1.67	MHz
Conversion Time (Note 12)			17		A/D Clock Cycles
Internal Reference Resistance Turn-On Time (Note 15)				1	µs

Note 12: Conversion Time includes sample and hold time.

Note 13: See Prescaler description.

Note 14: For $V_{IN}(-) = V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip doides are ties to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct — especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 15: Time for internal reference resistance to turn on after coming out of Halt or Idle Mode.

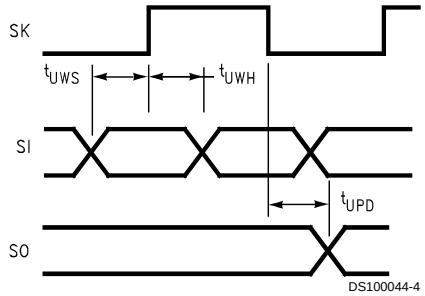


FIGURE 3. MICROWIRE/PLUS Timing Diagram

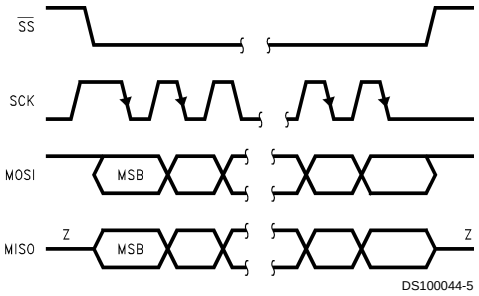
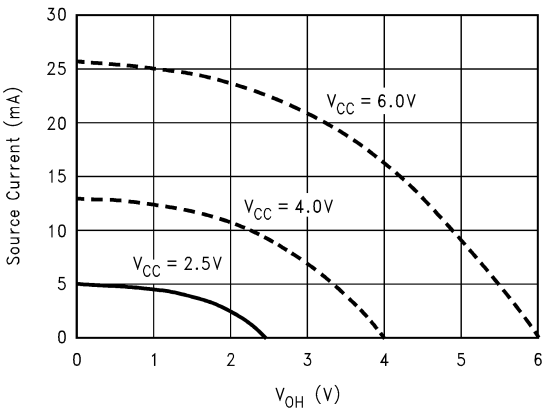


FIGURE 4. SPI Timing Diagram

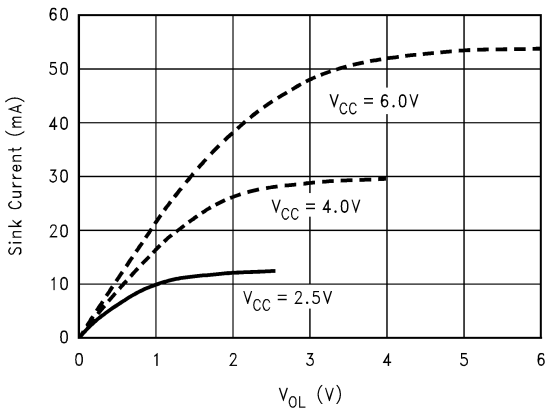
Typical Performance Characteristics ($-55^{\circ}C \le T_A = +125^{\circ}C$)

Port D Source Current



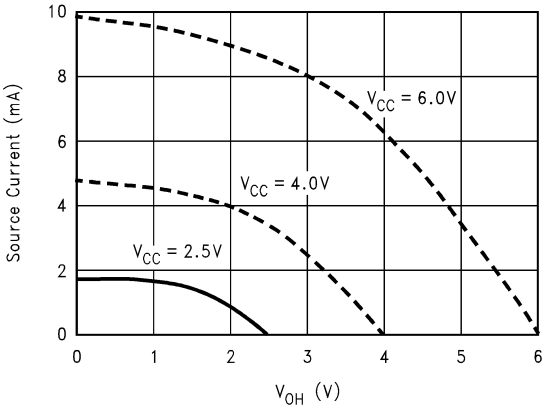
DS100044-57

Port D Sink Current



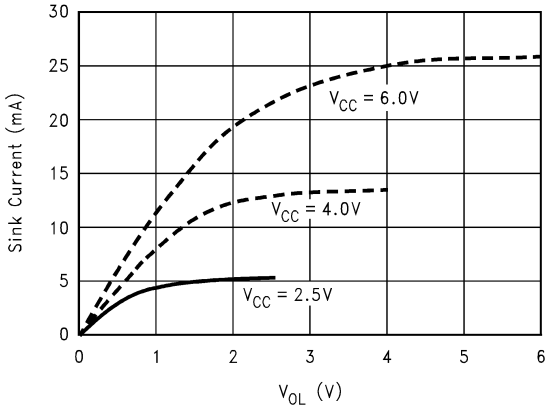
DS100044-58

Ports C/G/L Source Current



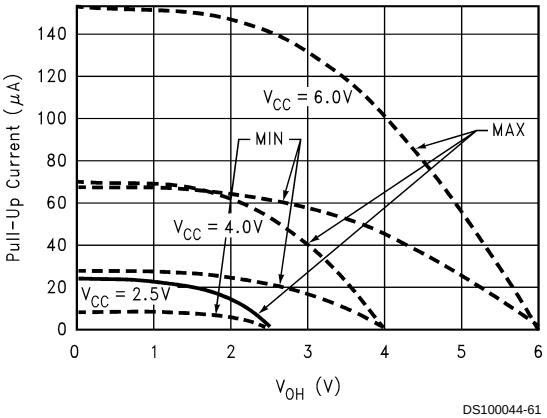
DS100044-59

Ports C/G/L Sink Current



DS100044-60

Ports C/G/L Weak Pull-Up Source Current



DS100044-61

Pin Description

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 seven bidirectional 8-bit I/O ports (C, E, F, G, L, M, N) where each individual bit may be independently configured as an input (Schmitt trigger inputs on all ports), 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 and M are 8-bit I/O ports, they support Multi-Input Wake-up (MIWU) on all eight pins. All L-pins and M-pins have Schmitt triggers on the inputs.

Port L and M only have one (1) interrupt vector.

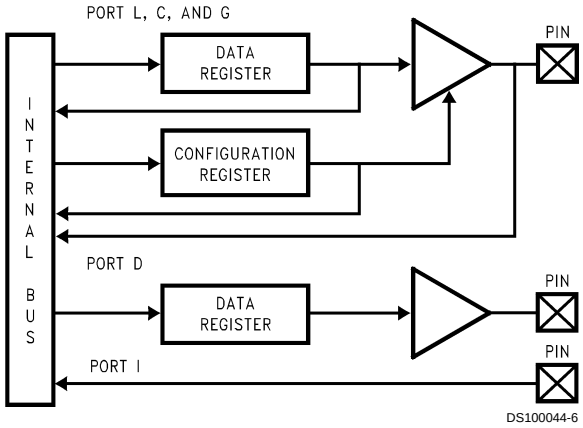


FIGURE 5. I/O Port Configurations

Port L has the following alternate features:

- L7 MIWU
- L6 MIWU
- L5 MIWU
- L4 MIWU
- L3 MIWU or RDX
- L2 MIWU or TDX
- L1 MIWU or CKX
- L0 MIWU

Port G is an 8-bit port with 5 I/O pins (G0–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 wirting a ”1” to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a “1” to bit 6 of the Port G Data Register.

Writing a “1” to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock

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

Port G has the following alternate features:

- G6 SI (MICROWIRE Serial Data Input)
- G5 SK (MICROWIRE Serial Clock)
- G4 SO (MICROWIRE Serial Data Output)
- G3 T1A (Timer I/O)
- G2 (Timer T1 Capture Input)
- G1 Dedicated WATCHDOG output
- G0 INTR (External Interrupt Input)

Port G has the following dedicated function:

- G7 CKO Oscillator dedicated output

Port M is a bidirectional I/O, it may be configured in software as Hi-Z input, weak pull-up, or push-pull output. These pins may be used as general purpose input/output pins or for selected altlernate functions.

Port M pins have optional alternate functions. Each pin (M0–M5) has been assigned an alternate data, configuration, or wakeup source. If the respective alternate function is selected the content of the associated bits in the configuration and/or data register are ignored. If an alternate wakeup source is selected the input level at the respective pin will be ignored for the purpose of triggering a wakeup event, however it will still be possible to read that pin by accessing the input register. The SPI (Serial Peripheral Interface) block, for example, uses four of the Port M pins to automatically reconfigure its MISO (Master Input, Slave Output), MOSI (Master Output, Slave Input), SCK (Serial Clock) and Slave-Select pins as inputs or outputs, depending on whether the interface has been configured as a Master or Slave. When the SPI interface is disabled those pins are available as general purpose I/O pins configurable by user software writing to the associated data and configuration bits. The CAN interface on the device makes use of one of the Port M’s alternate wake-ups, to trigger a wakeup if such a condition has been detected on the CAN’s dedicated receive pins.

Port M has the following alternate pin functions:

- M7 Multi-input Wakeup or CAN
- M6 Multi-input Wakeup
- M5 Multi-input Wakeup or T2B

Pin Description (Continued)

M4 Multi-input Wakeup or T2A

M3 Multi-input Wakeup or $\overline{SS}$

M2 Multi-input Wakeup or SCK

M1 Multi-input Wakeup or MOSI

M0 Multi-input Wakeup or MISO

Ports C, E, F and N are general-purpose, bidirectional I/O ports.

Any device package that has Port C, E, F, M, N but has fewer than eight pins, contains unbonded, floating pads internally on the chip. For these types of devices, the software should write a 1 to the configuration register bits corresponding to the non-existent port pins. This configures the port bits as outputs, thereby reducing leakage current of the device.

Port N is an 8-bit wide port with alternate function capability used for extending the slave select ($\overline{SS}$) lines of the on SPI interface. The SPI expander block provides mutually exclusive slave select extension signals ($\overline{ESS0}$ to $\overline{ESS7}$) according to the state of the $\overline{SS}$ line and specific contents of the SPI shift register. These slave select extension lines can be routed to the Port N I/O pins by enabling the alternate function of the port in the PORTNX register. If enabled, the internal signal on the $\overline{ESSx}$ line causes the ports state to change exactly like a change to the PORTND register. It is the user's responsibility to switch the port to an output when enabling the alternate function.

Port N has the following alternate pin functions:

N7 $\overline{ESS7}$

N6 $\overline{ESS6}$

N5 $\overline{ESS5}$

N4 $\overline{ESS4}$

N3 $\overline{ESS3}$

N2 $\overline{ESS2}$

N1 $\overline{ESS1}$

N0 $\overline{ESS0}$

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.

ALTERNATE PORT FUNCTIONS

Many general-purpose pins have alternate functions. The software can program each pin to be used either for a general-purpose or for a specific function. The chip hardware determines which of the pins have alternate functions, and what those functions are. This section lists the alternate functions available on each of the pins.

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

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

Port I is an 8-bit Hi-Z input port, and also provides the analog inputs to the A/D converter. If unterminated, Port I pins will draw power only when addressed.

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 8 or 32 kbytes of OTP EPROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the device vector to program memory location 0FF Hex.

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

SECURITY FEATURE

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

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

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated

Functional Description (Continued)

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 192 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, CNTRL, and INCTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The Stack Pointer, SP, is initialized to 06F Hex.

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

SPI:

SPICNTRL: Cleared

SPISTAT: Cleared

STBE Bit: Set

T1CNTRL & T2CNTRL: Cleared

ITMR: Cleared and IDLE timer period is reset to 4k Instr. CLK

ENAD: Cleared

ADDSLT: Random

SIOR: Unaffected after RESET with power already applied.

Random after RESET at power on.

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 6F Hex

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 util eleven consecutive "recessive" (undriven) bits have been received. This is done to ensure that the output drivers are not enamble during an active message on the bus.

CSCAL, CTIM, TCNTL, TEC, REC: CLEARED

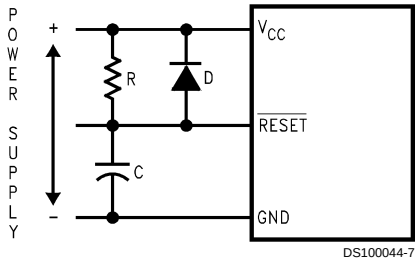
RTSTAT: CLEARED with the exception of the TBE bit which is set to 1

RID, RIDL, TID, TDLC: RANDOM

WATCHDOG: The device comes out of reset with both the WATCHDOG logic and the Clock Monitor detector armed, with the WATCHDOG service window bits set and the Clock Monitor bit set. The WATCHDOG and Clock Monitor circuits are inhibited during reset. The WATCHDOG service window bits being initialized high default to the maximum WATCHDOG service window of 64k t_c clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16 t_c–32 t_c clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The $\overline{\text{RESET}}$ signal goes directly to the HALT latch to restart a halted chip.

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.



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.

CRYSTAL OSCILLATOR

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

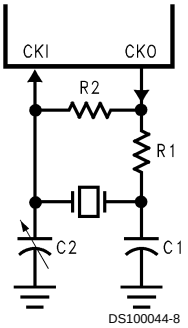


FIGURE 7. Crystal Oscillator Diagram

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

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

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

Control Registers

CNTRL Register (Address X'00EE)

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

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

- T1C3 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C1 Timer T1 mode control bit
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2, T1 Underflow Interrupt Pending Flag in timer mode 3
- MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)

PSW Register (Address X'00EF)

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

The PSW register contains the following select bits:

- HC Half Carry Flag
- C Carry Flag
- T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
- T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
- EXPND External interrupt pending
- BUSY MICROWIRE/PLUS busy shifting flag
- EXEN Enable external interrupt
- GIE Global interrupt enable (enables interrupts)

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

ICNTRL Register (Address X'00E8)

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

The ICNTRL register contains the following bits:

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

T2CNTRL Register (Address X'00C6)

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

The T2CNTRL control register contains the following bits:

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

Timers

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

TIMER T0 (IDLE TIMER)

The device supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

- Exit out of the Idle Mode (See Idle Mode description)
- WATCHDOG logic (See WATCHDOG description)
- Start up delay out of the HALT mode

Figure 8 is a functional block diagram showing the structure of the IDLE Timer and its associated interrupt logic.

Bits 11 through 15 of the ITMR register can be selected for triggering the IDLE Timer interrupt. Each time the selected bit underflows (every 4k, 8k, 16k, 32k or 64k instruction cycles), the IDLE Timer interrupt pending bit T0PND is set, thus generating an interrupt (if enabled), and bit 6 of the Port G data register is reset, thus causing an exit from the IDLE mode if the device is in that mode.

In order for an interrupt to be generated, the IDLE Timer interrupt enable bit T0EN must be set, and the GIE (Global Interrupt Enable) bit must also be set. The T0PND flag and T0EN bit are bits 5 and 4 of the ICNTRL register, respectively. The interrupt can be used for any purpose. Typically, it is used to perform a task upon exit from the IDLE mode. For more information on the IDLE mode, refer to the Power Save Modes section.

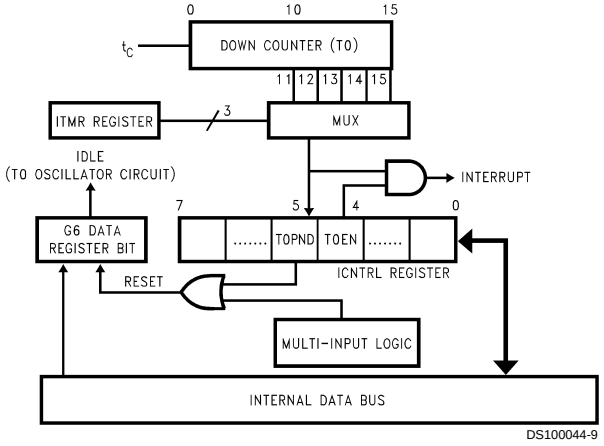


FIGURE 8. Functional Block Diagram for Idle Timer T0

The Idle Timer period is selected by bits 0–2 of the ITMR register Bits 3–7 of the ITMR Register are reserved and should not be used as software flags.

ITMR Register (Address X'0xCF)

Reserved	ITSEL2	ITSEL1	ITSLE0
Bit 7			Bit 0

TABLE 2. Idle Timer Window Length

ITSEL2	ITSEL1	ITSEL0	Idle Timer Period (Instruction Cycles)
0	0	0	4,096
0	0	1	8,192
0	1	0	16,384
0	1	1	32,768
1	X	X	65,536

The ITMR register is cleared on Reset and the Idle Timer period is reset to 4,096 instruction cycles.

Any time the IDLE Timer period is changed there is the possibility of generating a spurious IDLE Timer interrupt by setting the T0PND bit. The user is advised to disable IDLE Timer interrupts prior to changing the value of the ITSEL bits of the ITMR Register and then clear the T0PND bit before attempting to synchronize operation to the IDLE Timer.

TIMER T1 and TIMER T2

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

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the device to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the device to generate a PWM signal with very minimal user intervention.

The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely indepen-

Timers (Continued)

dent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

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

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

Figure 9 shows a block diagram of the timer in PWM mode.

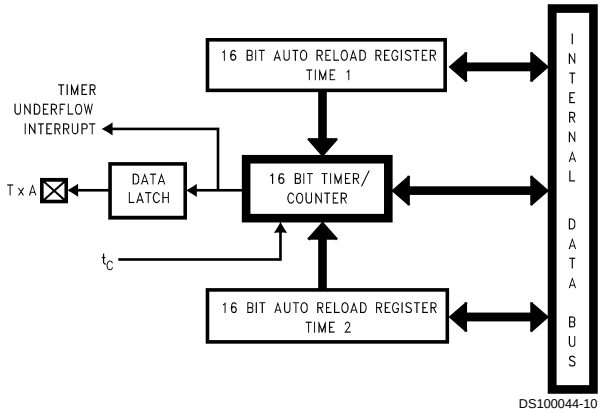


FIGURE 9. Timer in PWM Mode

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

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

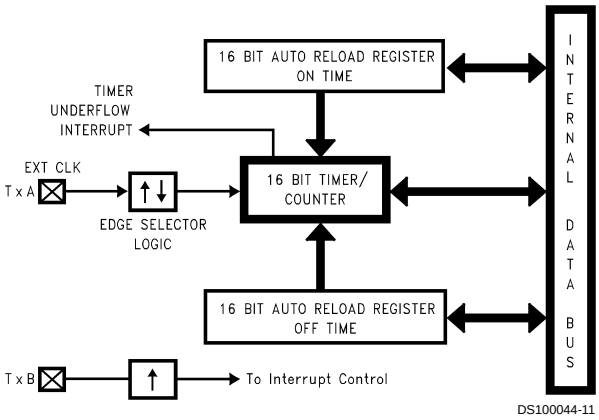
Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

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

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

FIGURE 10. Timer in External Event Counter Mode

Mode 3. Input Capture Mode

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

In this mode, the timer Tx is constantly running at the fixed t_c rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

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

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

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

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

Timers (Continued)

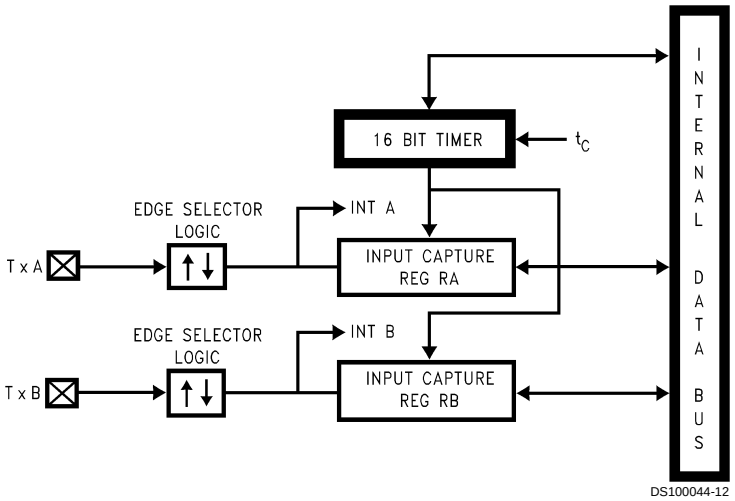


FIGURE 11. Timer in Input Capture Mode

TIMER CONTROL FLAGS

The control bits and their functions are summarized below.

TxC3 Timer mode control

TxC2 Timer mode control

TxC1 Timer mode control

TxC0 Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop

Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)

TxPNDA Timer Interrupt Pending Flag

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

TxPNDB Timer Interrupt Pending Flag

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

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

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

Power Save Modes

The device offer the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The device 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 Vr (Vr = 2.0V) without altering the state of the machine.

CAN HALT/IDLE mode:

In order to reduce the device overall current consumption in HALT/IDLE mode a two step power save mechanism is implemented on the device:

Step 1: Disable main receive comparator. This is done by resetting both the TxEN0 and TxEN1 bits in the CBUS register. Note: These bits should always be reset before entering HALT/IDLE mode to allow proper resynchronization to the CAN bus after exiting HALT/IDLE mode.

Step 2: Disable the CAN wake-up comparators, this is done by resetting bit 7 in the port-m wakeup enable register (MWKEN) a transition on the CAN bus will then not wake the device up.

Note: If both the main receive comparator and the wake-up comparator are disabled the on chip CAN voltage reference is also disabled. The CAN-V_{REF} output is then High-Z

The following table shows the two CAN power save modes and the active CAN transceiver blocks:

Step 1	Step 2	Main-Comp	Wake-Up-Comp	CAN-V_{REF}	V_{REF} Pin
0	0	on	on	on	$V_{CC}/2$
0	1	on	off	on	$V_{CC}/2$
1	0	off	on	on	$V_{CC}/2$
1	1	off	off	off	High-Z

The device supports two different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L & M 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 Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The 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 to 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 activity, except the associated on-board oscillator circuitry, ad the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the device can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the Port L 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 instruciton following the "Enter Idle Mode" instruction.

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

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

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

Note: The following description is for both the Port L and the M port. When the document refers to the registers WKEGD, WKEN or WKPND, the user will have to put either M (for M port) or L (for port) in front of the register, i.e., LWKEN (Port L WKEN), MWKEN (Port M WKEN).

Figures 12, 13 shows the Multi-Input Wakeup logic for the microcontroller. The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular Port L bit (or combi-

Multi-Input Wakeup (Continued)

nation of Port L 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 Port L bit. Setting a particular WKEN bit enables a Wakeup from the associated Port L pin.

The user can select whether the trigger condition on the selected Port L 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 Port L pin. Setting the control bit will select the trigger condition to be a negative edge on that particular Port L pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

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

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

If the Port L bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected Port L 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 assoicated WKPND bits being cleared.

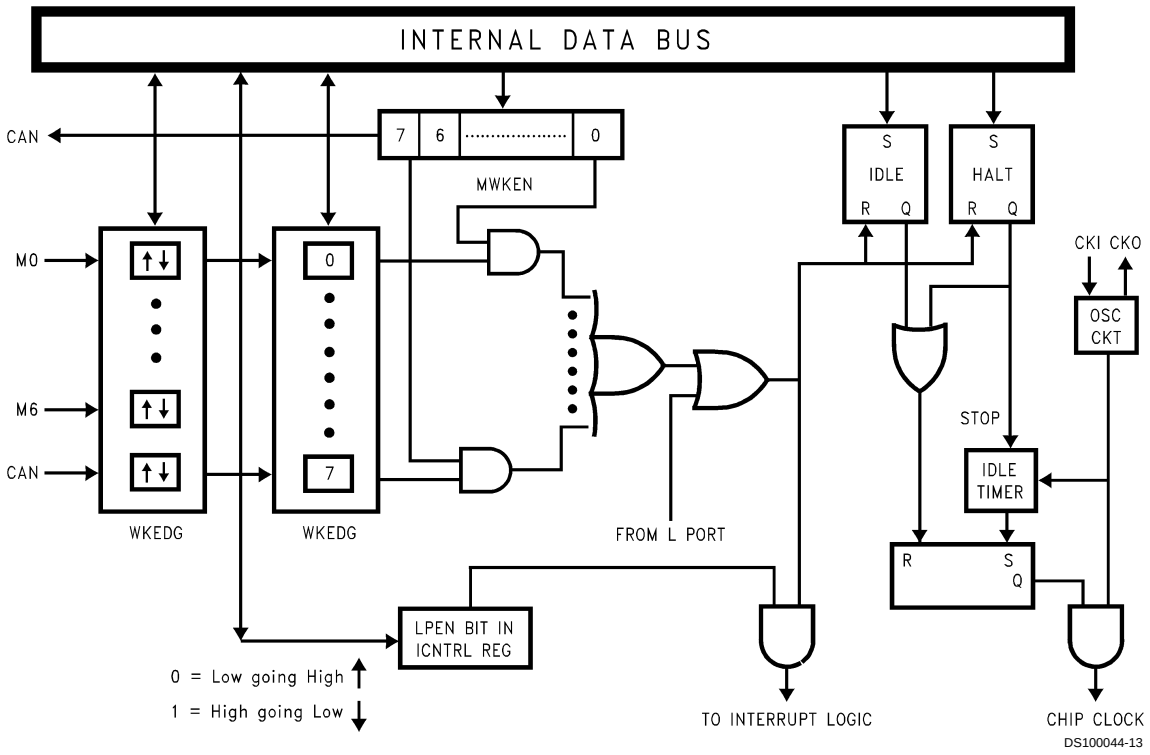


FIGURE 12. Port M Multi-Input Wake-up Logic

This same procedure should be used following reset, since the Port L inputs are left floating as a result of reset. The occurrence of the selected trigger condition for Multi-Input Wakeup is latched to 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 and Port M pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any wakeup bit is both enabled and pending. Consequently, the user 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.

PORT L INTERRUPTS

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

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

Multi-Input Wakeup (Continued)

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 Wakeup 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 Wakeup 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.

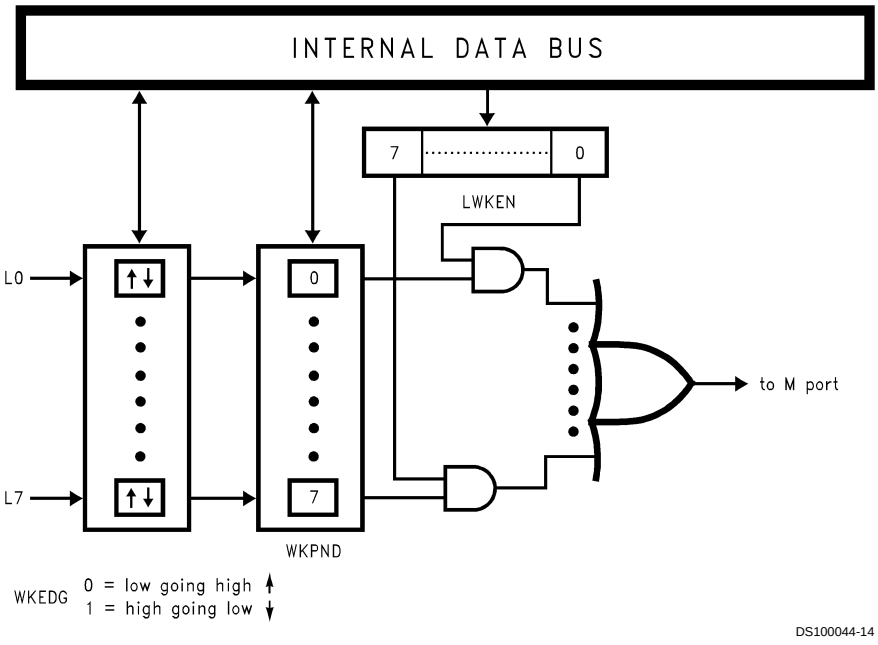


FIGURE 13. Port L Multi-Input Wake-Up Logic

PORT M INTERRUPTS

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

The interrupt from Port M shares logic with the wake up circuitry. The MWKEN register allows interrupts from Port M to be individually enabled or disabled. The MWKEDG register specifies the trigger condition to be either a positive or a negative edge. The MWKPND register latches in the pending trigger conditions.

The LPEN control flag in the ICNTRL register functions as a global interrupt enable for Port M interrupts. Setting the LPEN flag enables interrupts. Note that the GIE bit in the PSW register must also be set to enable these Port L interrupts. A global pending flag is not needed since each pin has a corresponding pending flag in the MWKPND register.

Since Port M is also used for exiting the device from the HALT or IDLE mode, the user can elect to exit the HALT or IDLE mode either with or without the interrupt enabled. If the user elects to disable the interrupt, then the device restarts execution from the point at which it was stopped (first instruction cycle of the instruction following the enter HALT or IDLE mode instruction). In the other case, the device finishes the instruction which was being executed when the part was stopped (the NOP(Note *NO TARGET FOR FNXref NS9529*) instruction following the enter HALT or IDLE mode instruction), and then branches to the interrupt service routine. The device then reverts to normal operation.

Note 16: The user must place two NOPs after an enter HALT or IDLE mode instruction.

To prevent erroneous clearing of the SPI receive FIFO when entering HALT/IDLE mode, the user needs to enable the MIWU on port M3. ($\overline{SS}$) by setting bit 3 in the MWKEN register.

CAN RECEIVE WAKEUP

The CAN Receive Wakeup source can be enabled or disabled. There is no specific enable bit for the CAN Wakeup feature. Although the wakeup feature on pins L0..17 and M0..M7 can be programmed to generate an interrupt (Port L or Port M interrupt), no interrupt is generated upon a CAN re-

Multi-Input Wakeup (Continued)

ceive wakeup condition. The CAN block has it's own, dedicated receiver interrupt upon receive buffer full (see CAN Section).

CAN Wake-Up:

The CAN interface can be programmed to wake the device from HALT/IDLE mode. This is done by setting bit 7 in the Port M wake-up enable register (MWKEN). A transition on the bus will cause the bit 7 of the Port M wake-up pending (MWKPND) to be set and thereby waking up the device. The frame on the CAN bus will be lost. The MWEDG (m port wake-up edge) register bit 7 can be programmed high or low (high will wake-up on the first falling edge on Rx0).

Resetting bit 7 in the MWKEN will disable the CAN wake-up.

The following sequence should be executed before entering HALT/IDLE mode:

```
RBIT 7, MWKPND ;clear CAN wake-up pending
LD   A, CBUS
AND  A, #0CF   ;resetTxEN0 and TxEN1
X    A, CBUS   ;disable main receive
               ;comparator
```

After the device woke-up the CBUS bits TxEN0 and/or TxEN1 need be set to allow synchronization on the bus and to enable transmission/reception of CAN frames.

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.

CAN Interface Block (Continued)

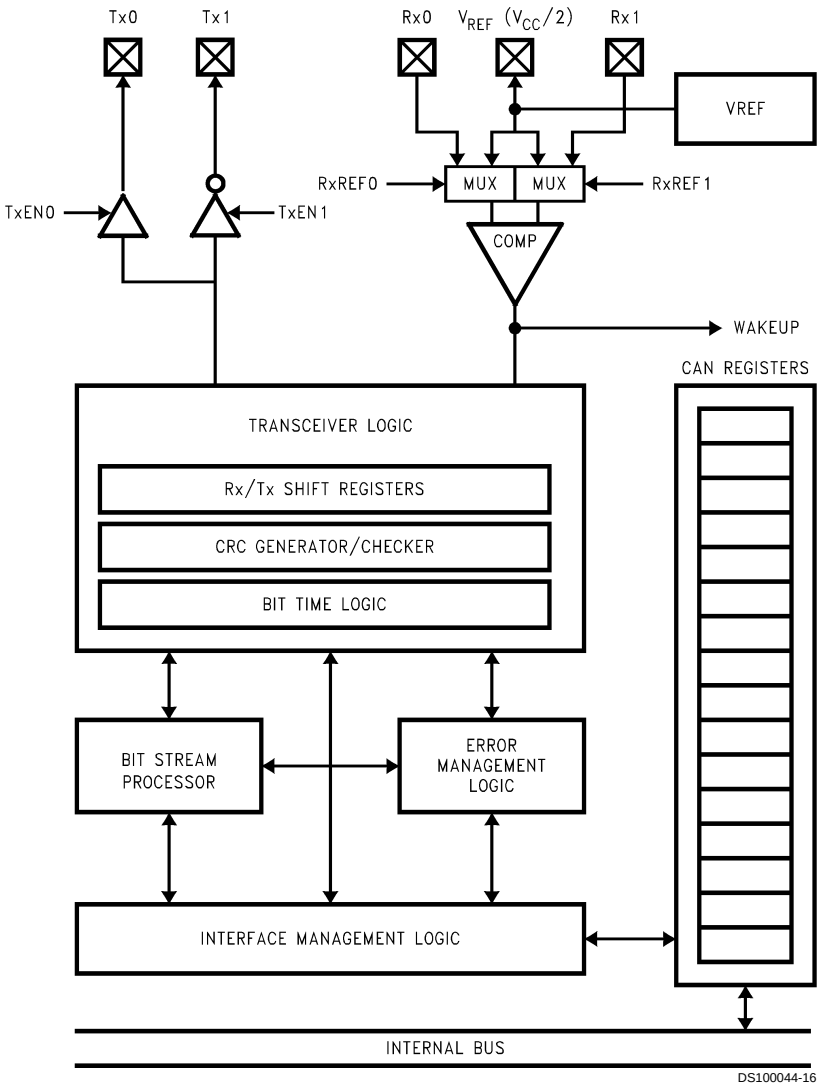


FIGURE 14. CAN Interface Block Diagram

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:

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

Functional Block Description of the CAN Interface (Continued)

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 3 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 3. 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

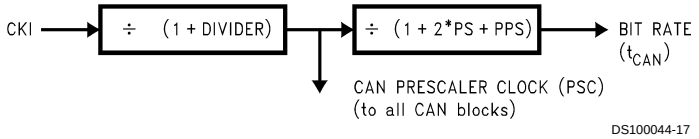


FIGURE 15. Bit Rate Generation

Figure 16 illustrates the minimum time required for t_{LOAD}.

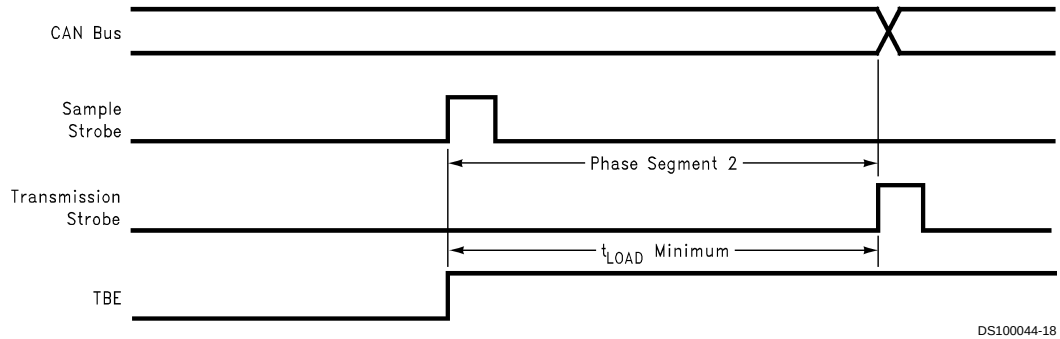


FIGURE 16. TBE Timing

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
 •
 •
 •
```

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.

Functional Block Description of the CAN Interface (Continued)

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 4. 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'00A0)

The Transmit Data Register 1 contains 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'00A1)

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'00A2)

TID3	TID2	TID1	TID0	TDLC3	TDLC2	TDLC1	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'00A3)

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 IDENTIFIERLOW REGISTER (RIDL) (Address X'00A6)

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'00A7)

Reserved	RID10	RID9	RID8	RID7	RID6	RID5	RID4
Bit 7							Bit 0

This register is read/write.

Bit 7 is reserved and should be zero.

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.

CAN PRESCALER REGISTER (CSCAL) (Address X'00A8)

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) (00A9)

PPS2	PPS1	PPS0	PS2	PS1	PS0	Reserved	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.)

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

Functional Block Description of the CAN Interface (Continued)

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: 4 PSC, or the length of PS1/2 (Min: 4 l length of PS1/2).

Bits 1 and 0 are reserved and should be zero.

TABLE 5. 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 28*)

- 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) (00AA)

Re-served	RIAF	TxEN1	TxEN0	RxREF1	RxREF0	Re-served	FMOD
Bit 7							Bit 0

Reserved These bits are 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 6. 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).

TRANSMIT CONTROL/STATUS (TCNTL) (00AB)

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

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

TABLE 7. 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 fewer 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 gen-

Functional Block Description of the CAN Interface (Continued)

erate 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 transmit 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.

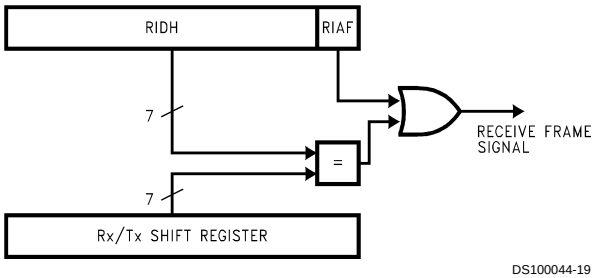


FIGURE 17. Acceptance Filter Block-Diagram

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'00AC)

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 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.

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 automati-

Functional Block Description of the CAN Interface (Continued)

cally 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'00AD)

TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
Bit 7							Bit 0

This register is read/write.

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) (00AE)

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 which fulfills the CAN 2.0, 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.

Basic CAN Concepts (Continued)

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.

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 18*.

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.)

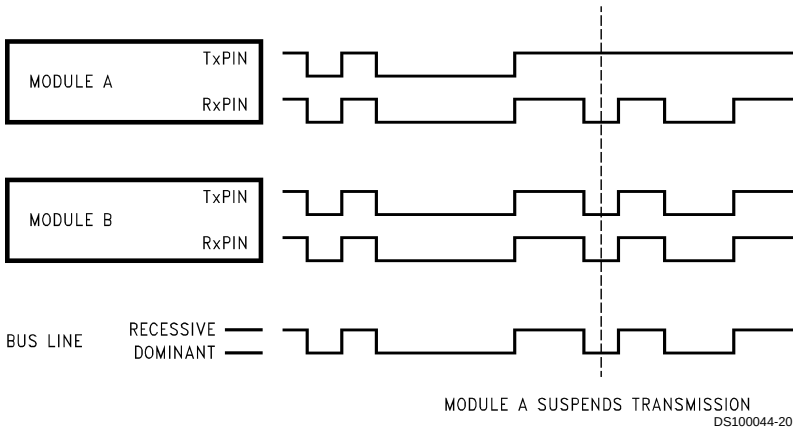


FIGURE 18. CAN Message Arbitration

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 19* and *Figure 20* 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 21* 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
	Note: x = {0,1}	

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.

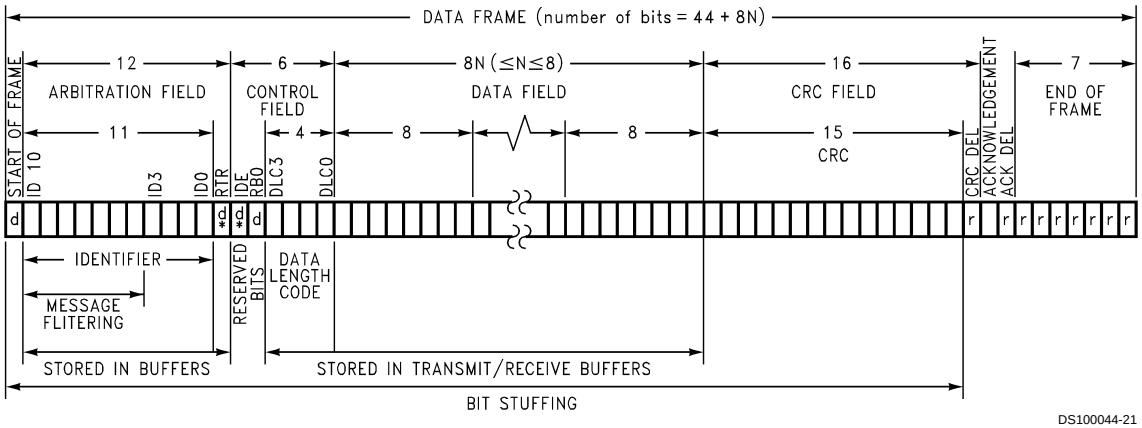
CRC FIELD

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:

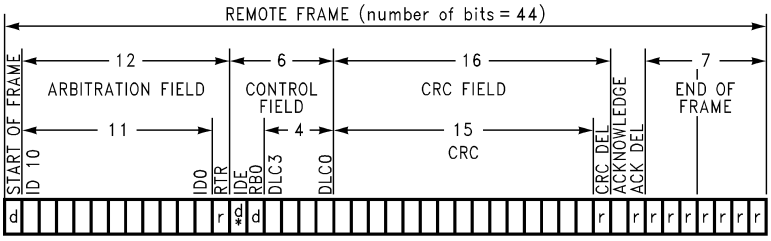
$$\chi^{15} + \chi^{14} + \chi^{10} + \chi^{8} + \chi^{7} + \chi^{4} + \chi^{3} + 1$$

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.

Frame Formats (Continued)



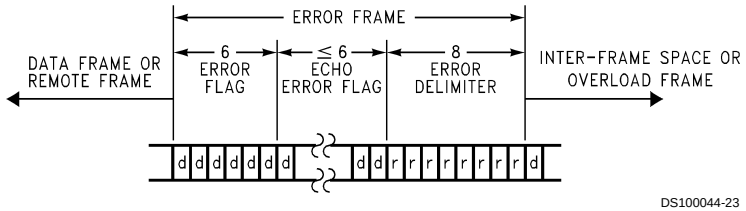
DS100044-21



DS100044-22

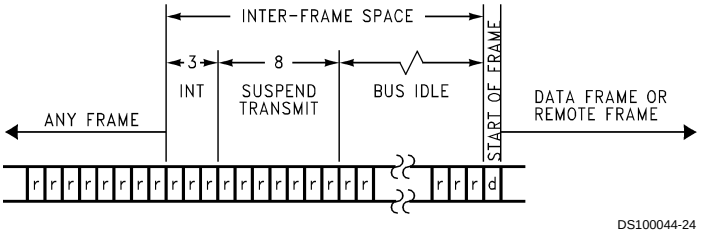
A remote frame is identical to a data frame, except that the RTR bit is “recessive”, and there is no data field.
IDE = Identifier Extension Bit
The IDE bit in the standard format is transmitted “dominant”, whereas in the extended format the IDE bit is “recessive” and the id is expanded to 29 bits.
r = recessive
d = dominant

FIGURE 19. CAN Data Transmission Frames



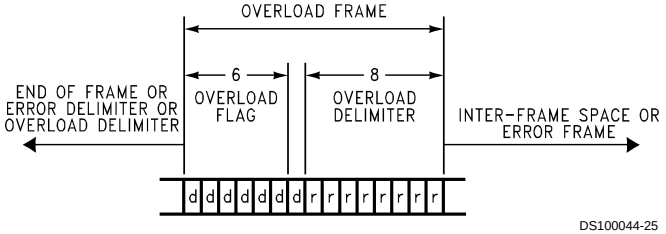
DS100044-23

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



DS100044-24

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



DS100044-25

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

FIGURE 20. CAN Control Frames

Frame Formats (Continued)

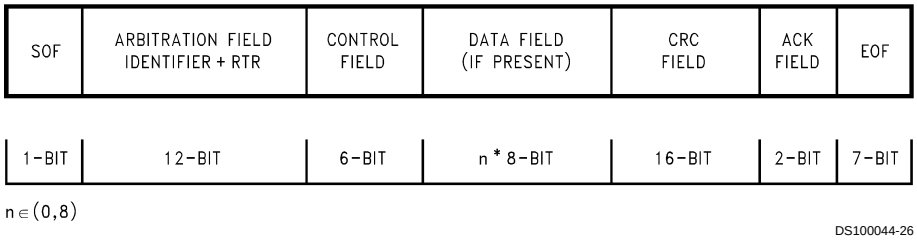


FIGURE 21. CAN Frame Format

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 22* and *Figure 23* 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 24* 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.

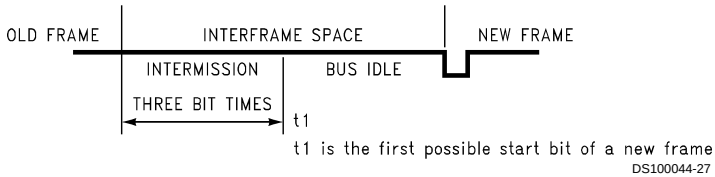
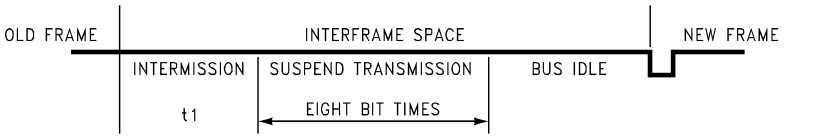


FIGURE 22. Interframe Space for Nodes Which Are Not Error Passive or Have Been Receiver for the Last Frame

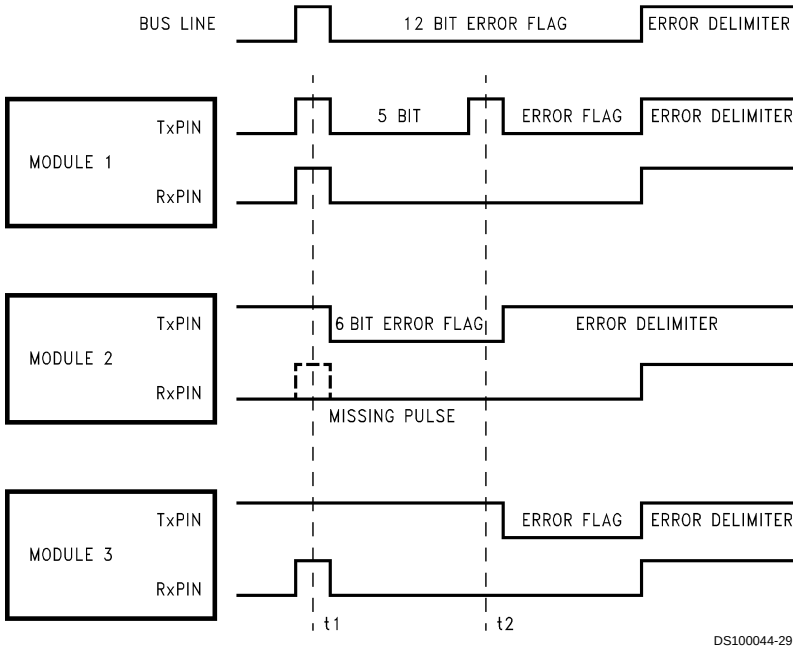
Frame Formats (Continued)



t1 – any module can start transmission except the error passive module which has transmitted the last frame

DS100044-28

FIGURE 23. Interframe Space for Nodes Which Are Error Passive and Have Been Transmitter for the Last Frame

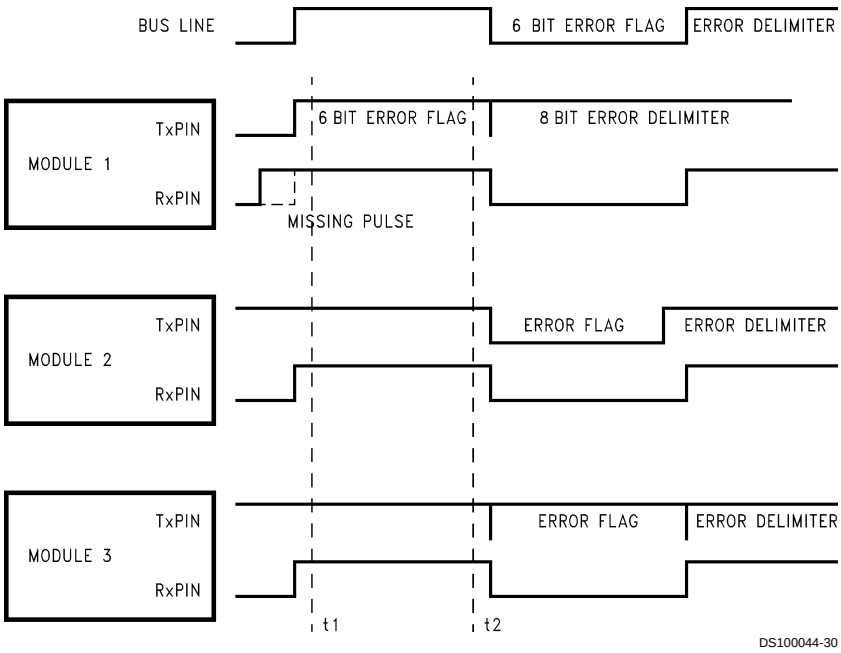


DS100044-29

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 24. Error Frame — Error Active Transmitter

Frame Formats (Continued)



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 25. Error Frame — Error Passive Transmitter

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:

- 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.

Frame Formats (Continued)

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.

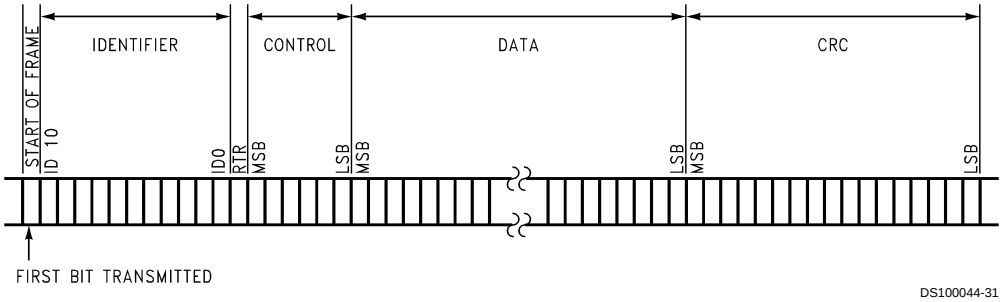


FIGURE 26. Order of Bit Transmission within a CAN Frame

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.

Frame Formats (Continued)

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

TABLE 8. 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 9. 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.

Figure 27 shows the connection of different bus states according to the error counters.

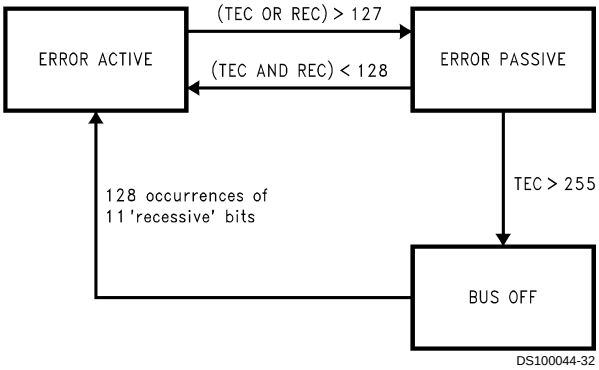


FIGURE 27. CAN Bus States

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:

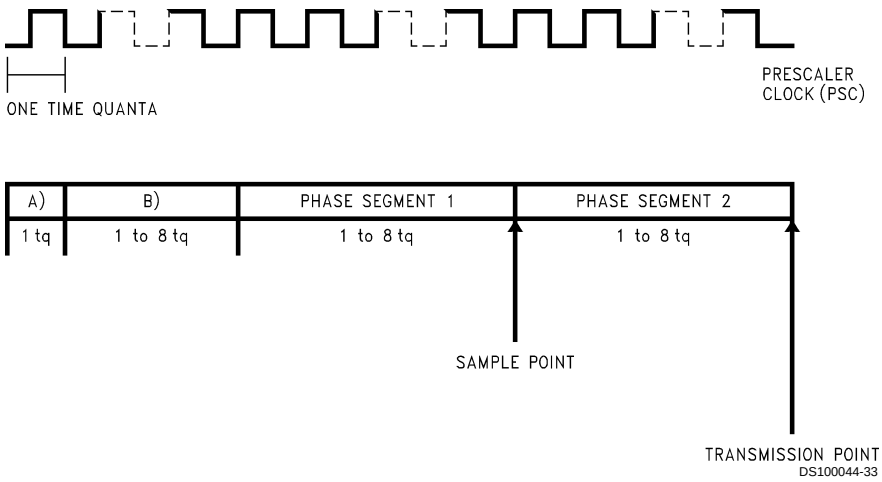
Hard synchronization is done with the falling edge on the bus while the bus is idle, which is then interpreted as the SOF. It restarts the internal logic.

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 29*).

If a falling edge lies in the phase segment 2 (as shown in *Figure 29*) 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.

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.
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).

Frame Formats (Continued)



A) Synchronization segment
B) Propagation segment

FIGURE 28. Bit Timing

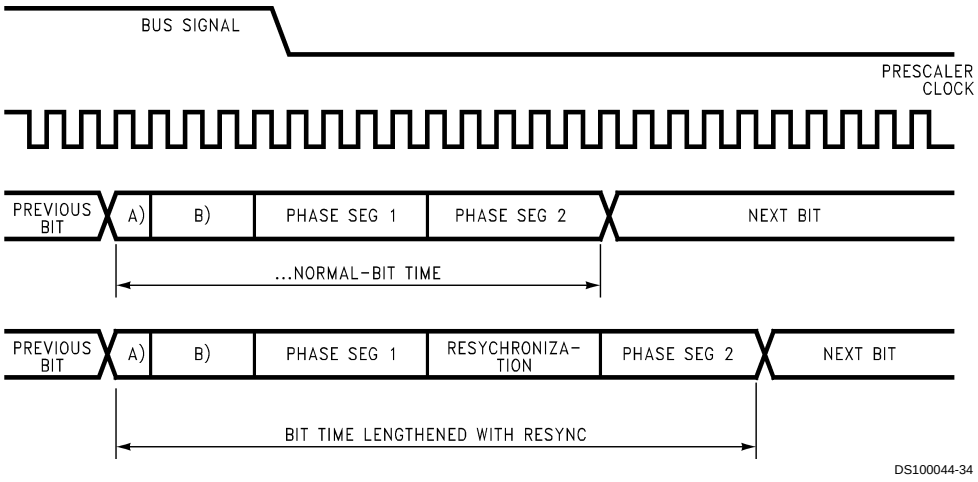


FIGURE 29. Resynchronization 1

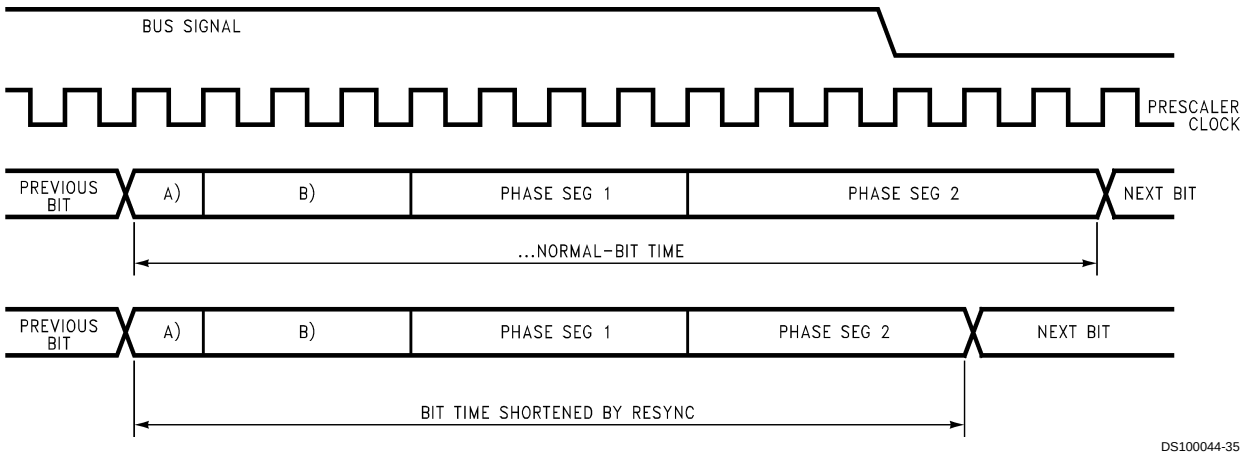


FIGURE 30. Resynchronization 2

Interrupts

INTRODUCTION

Each device supports fourteen vectored interrupts. Interrupt sources include Timer 0, Timer 1, Timer 2, Timer 3, Port L Wakeup, Software Trap, MICROWIRE/PLUS, and External Input.

All interrupts force a branch to location 00FF Hex in program memory. The VIS instruction may be used to vector to the appropriate service routine from location 00FF Hex.

The Software trap has the highest priority while the default VIS has the lowest priority.

Each of the 14 maskable inputs has a fixed arbitration ranking and vector.

Figure 31 shows the Interrupt Block Diagram.

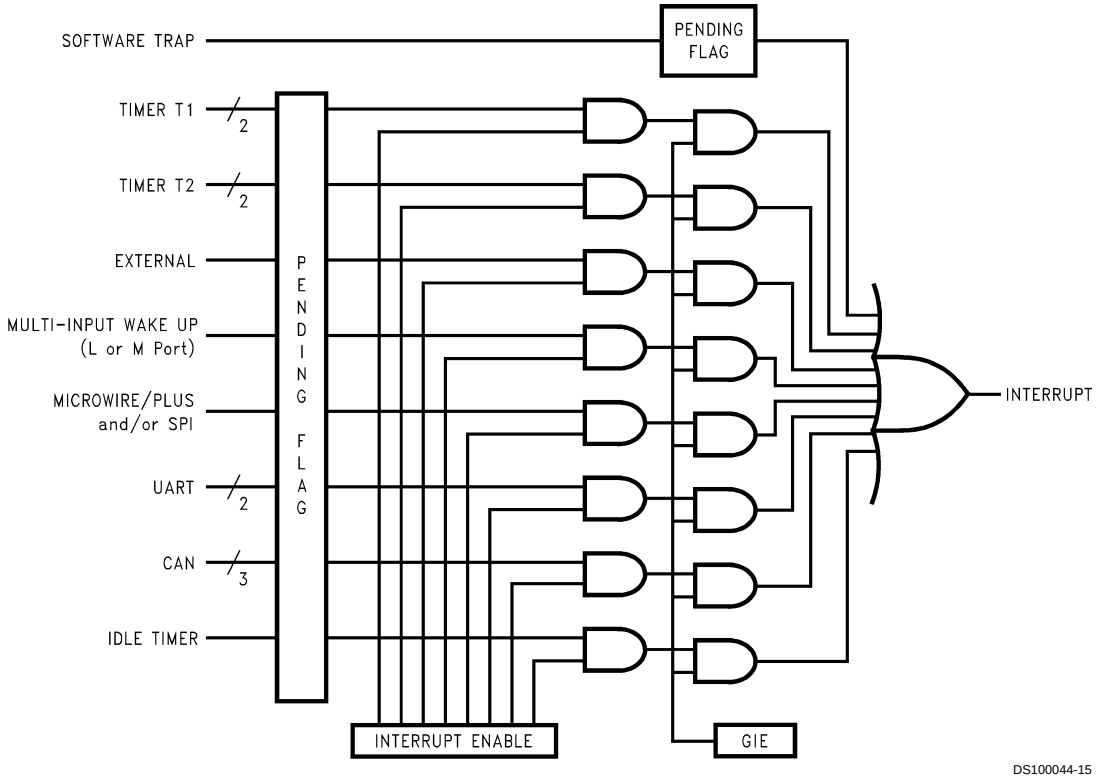


FIGURE 31. Interrupt Block Diagram

Interrupts (Continued)

MASKABLE INTERRUPTS

All interrupts other than the Software Trap are maskable. Each maskable interrupt has an associated enable bit and pending flag bit. The pending bit is set to 1 when the interrupt condition occurs. The state of the interrupt enable bit, combined with the GIE bit determines whether an active pending flag actually triggers an interrupt. All of the maskable interrupt pending and enable bits are contained in mapped control registers, and thus can be controlled by the software.

A maskable interrupt condition triggers an interrupt under the following conditions:

1. The enable bit associated with that interrupt is set.
2. The GIE bit is set.
3. The device is not processing a non-maskable interrupt. (If a non-maskable interrupt is being serviced, a maskable interrupt must wait until that service routine is completed.)

An interrupt is triggered only when all of these conditions are met at the beginning of an instruction. If different maskable interrupts meet these conditions simultaneously, the highest priority interrupt will be serviced first, and the other pending interrupts must wait.

Upon Reset, all pending bits, individual enable bits, and the GIE bit are reset to zero. Thus, a maskable interrupt condition cannot trigger an interrupt until the program enables it by setting both the GIE bit and the individual enable bit. When enabling an interrupt, the user should consider whether or not a previously activated (set) pending bit should be acknowledged. If, at the time an interrupt is enabled, any previous occurrences of the interrupt should be ignored, the associated pending bit must be reset to zero prior to enabling the interrupt. Otherwise, the interrupt may be simply enabled; if the pending bit is already set, it will immediately trigger an interrupt. A maskable interrupt is active if its associated enable and pending bits are set.

An interrupt is an asychronous event which may occur before, during, or after an instruction cycle. Any interrupt which occurs during the execution of an instruction is not acknowledged until the start of the next normally executed instruction is to be skipped, the skip is performed before the pending interrupt is acknowledged.

At the start of interrupt acknowledgment, the following actions occur:

1. The GIE bit is automatically reset to zero, preventing any subsequent maskable interrupt from interrupting the current service routine. This feature prevents one maskable interrupt from interrupting another one being serviced.
2. The address of the instruction about to be executed is pushed onto the stack.
3. The program counter (PC) is loaded with 00FF Hex, causing a jump to that program memory location.

The device requires seven instruction cycles to perform the actions listed above.

If the user wishes to allow nested interrupts, the interrupts service routine may set the GIE bit to 1 by writing to the PSW register, and thus allow other maskable interrupts to interrupt the current service routine. If nested interrupts are allowed, caution must be exercised. The user must write the program in such a way as to prevent stack overflow, loss of saved context information, and other unwanted conditions.

The interrupt service routine stored at location 00FF Hex should use the VIS instruction to determine the cause of the interrupt, and jump to the interrupt handling routine corresponding to the highest priority enabled and active interrupt. Alternately, the user may choose to poll all interrupt pending and enable bits to determine the source(s) of the interrupt. If more than one interrupt is active, the user's program must decide which interrupt to service.

Within a specific interrupt service routine, the associated pending bit should be cleared. This is typically done as early as possible in the service routine in order to avoid missing the next occurrence of the same type of interrupt event. Thus, if the same event occurs a second time, even while the first occurrence is still being serviced, the second occurrence will be serviced immediately upon return from the current interrupt routine.

An interrupt service routine typically ends with an RETI instruction. This instruction sets the GIE bit back to 1, pops the address stored on the stack, and restores that address to the program counter. Program execution then proceeds with the next instruction that would have been executed had there been no interrupt. If there are any valid interrupts pending, the highest-priority interrupt is serviced immediately upon return from the previous interrupt.

VIS INSTRUCTION

The general interrupt service routine, which starts at address 00FF Hex, must be capable of handling all types of interrupts. The VIS instruction, together with an interrupt vector table, directs the device to the specific interrupt handling routine based on the cause of the interrupt.

VIS is a single-byte instruction, typically used at the very beginning of the general interrupt service routine at address 00FF Hex, or shortly after that point, just after the code used for context switching. The VIS instruction determines which enabled and pending interrupt has the highest priority, and causes an indirect jump to the address corresponding to that interrupt source. The jump addresses (vectors) for all possible interrupts sources are stored in a vector table.

The vector table may be as long as 32 bytes (maximum of 16 vectors) and resides at the top of the 256-byte block containing the VIS instruction. However, if the VIS instruction is at the very top of a 256-byte block (such as at 00FF Hex), the vector table resides at the top of the next 256-byte block. Thus, if the VIS instruction is located somewhere between 00FF and 01DF Hex (the usual case), the vector table is located between addresses 01E0 and 01FF Hex. If the VIS instruction is located between 01FF and 02DF Hex, then the vector table is located between addresses 02E0 and 02FF Hex, and so on.

Each vector is 15 bits long and points to the beginning of a specific interrupt service routine somewhere in the 32 kbyte memory space. Each vector occupies two bytes of the vector table, with the higher-order byte at the lower address. The vectors are arranged in order of interrupt priority. The vector of the maskable interrupt with the lowest rank is located to 0yE0 (higher-order byte) and 0yE1 (lower-order byte). The next priority interrupt is located at 0yE2 and 0yE3, and so forth in increasing rank. The Software Trap has the highest rank and its vector is always located at 0yFE and 0yFF. The number of interrupts which can become active defines the size of the table.

Table 10 shows the types of interrupts, the interrupt arbitration ranking, and the locations of the corresponding vectors in the vector table.

The vector table should be filled by the user with the memory locations of the specific interrupt service routines. For ex-

Interrupts (Continued)

ample, if the Software Trap routine is located at 0310 Hex, then the vector location 0yFE and -0yFF should contain the data 03 and 10 Hex, respectively. When a Software Trap interrupt occurs and the VIS instruction is executed, the program jumps to the address specified in the vector table.

The interrupt sources in the vector table are listed in order of rank, from highest to lowest priority. If two or more enabled and pending interrupts are detected at the same time, the one with the highest priority is serviced first. Upon return from the interrupt service routine, the next highest-level pending interrupt is serviced.

If the VIS instruction is executed, but no interrupts are enabled and pending, the lowest-priority interrupt vector is used, and a jump is made to the corresponding address in the vector table. This is an unusual occurrence, and may be the result of an error. It can legitimately result from a change in the enable bits or pending flags prior to the execution of the VIS instruction, such as executing a single cycle instruction which clears an enable flag at the same time that the pending flag is set. It can also result, however, from inadvertent execution of the VIS command outside of the context of an interrupt.

The default VIS interrupt vector can be useful for applications in which time critical interrupts can occur during the servicing of another interrupt. Rather than restoring the program context (A, B, X, etc.) and executing the RETI instruction, an interrupt service routine can be terminated by returning to the VIS instruction. In this case, interrupts will be serviced in turn until no further interrupts are pending and the default VIS routine is started. After testing the GIE bit to ensure that execution is not erroneous, the routine should restore the program context and execute the RETI to return to the interrupted program.

This technique can save up to fifty instruction cycles (t_c), or more, (50 µs at 10 MHz oscillator) of latency for pending interrupts with a penalty of fewer than ten instruction cycles if no further interrupts are pending.

To ensure reliable operation, the user should always use the VIS instruction to determine the source of an interrupt. Although it is possible to poll the pending bits to detect the source of an interrupt, this practice is not recommended. The use of polling allows the standard arbitration ranking to be altered, but the reliability of the interrupt system is compromised. The polling routine must individually test the enable and pending bits of each maskable interrupt. If a Software Trap interrupt should occur, it will be serviced last, even though it should have the highest priority. Under certain conditions, a Software Trap could be triggered but not serviced, resulting in an inadvertent "locking out" of all maskable interrupts by the Software Trap pending flag. Problems such as this can be avoided by using VIS instruction.

TABLE 10. Interrupt Vector Table

Arbitration Rank	Interrupt Source	Description	Vector Address
1	Software Trap	INTR Instruction	0yFE–0yFF
2	reserved	NMI	0yFC–0yFD
3	CAN Receive	RBF, RFV set	0yFA–0yFB
4	CAN Error (transmit/receive)	TERR, RERR set	0yF8–0yF9
5	CAN Transmit	TBE set	0yF6–0yF7
6	Pin G0 Edge	External	0yF4–0yF5
7	MICROWIRE/PLUS SPI Interface	BUSY Goes Low SRBF or STBE set	0yF2–0yF3
8	Timer T0	Idle Timer Underflow	0yF0–0yF1
9	UART	receive buffer full	0yEE–0yEF
10	UART	transmit buffer empty	0yEC–0yED
11	Timer T2	T2A/Underflow	0yEA–0yEB
12	Timer T2	T2B	0yE8–0yE9
13	Timer T1	T1A/Underflow	0yE6–0yE7
14	Timer T1	T1B	0yE4–0yE5
15	Port L, Port M; MIWU	Port L Edge or Port M Edge	0yE2–0yE3
16	Default VIS Interrupt	VIS Interrupt	0yE0–0yE1

Note 17: y is a variable which represents the VIS block. VIS and the vector table must be located in the same 256-byte block except if VIS is located at the last address of a block. In this case, the table must be in the next block.

Interrupts (Continued)

VIS Execution

When the VIS instruction is executed it activates the arbitration logic. The arbitration logic generates an even number between E0 and FE (E0, E2, E4, E6 etc...) depending on which active interrupt has the highest arbitration ranking at the time of the 1st cycle of VIS is executed. For example, if the software trap interrupt is active, FE is generated. If the external interrupt is active and the software trap interrupt is not, then FA is generated and so forth. If the only active interrupt is software trap, than E0 is generated. This number replaces the lower byte of the PC. The upper byte of the PC remains unchanged. The new PC is therefore pointing to the vector of the active interrupt with the highest arbitration ranking. This vector is read from program memory and placed into the PC which is now pointed to the 1st instruction of the service routine of the active interrupt with the highest arbitration ranking.

Figure 32 illustrates the different steps performed by the VIS instruction. *Figure 33* shows a flowchart for the VIS instruction.

The non-maskable interrupt pending flag is cleared by the RPND (Reset Non-Maskable Pending Bit) instruction (under certain conditions) and upon RESET.

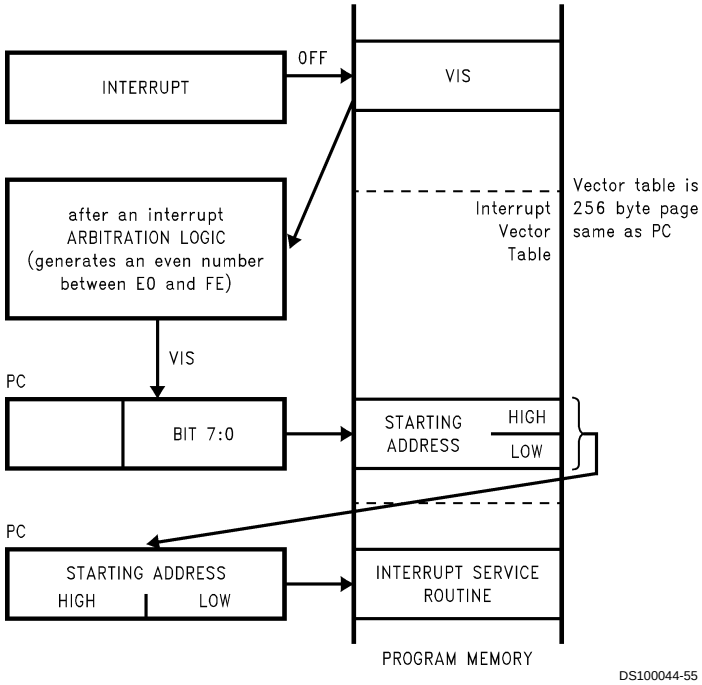


FIGURE 32. VIS Operation

Interrupts (Continued)

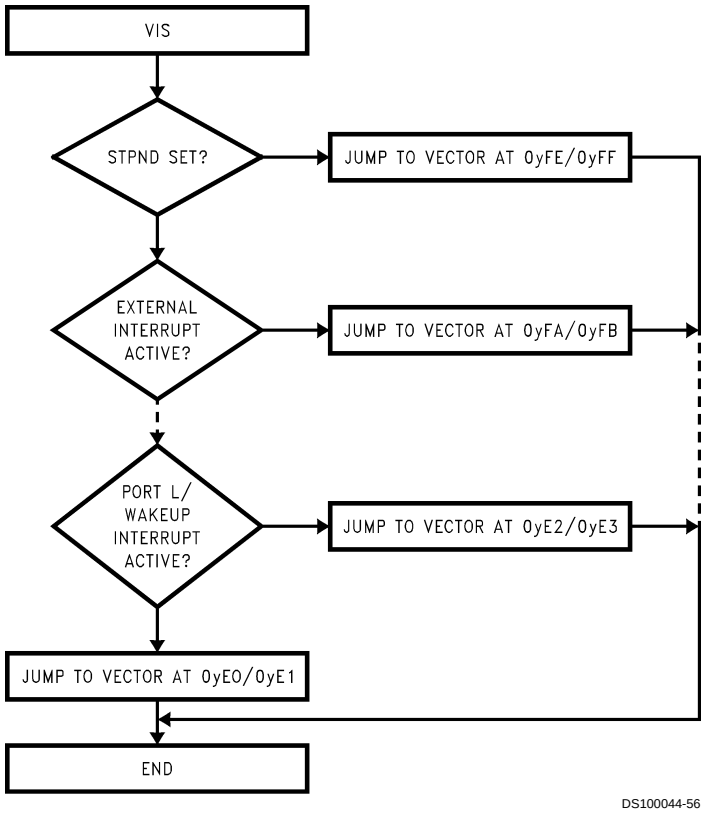


FIGURE 33. VIS Flowchart

Interrupts (Continued)

Programming Example: External Interrupt

```
           PSW            =00EF
           CNTRL          =00EE
           RBIT           0,PORTGC
           RBIT           0,PORTGD       ; G0 pin configured Hi-Z
           SBIT           IEDG, CNTRL    ; Ext interrupt polarity; falling edge
           SBIT           EXEN, PSW      ; Enable the external interrupt
           SBIT           GIE, PSW       ; Set the GIE bit
WAIT:      JP             WAIT           ; Wait for external interrupt
           .
           .
           .
           .=0FF                         ; The interrupt causes a
           VIS                           ; branch to address 0FF
                                         ; The VIS causes a branch to
                                         ;interrupt vector table
           .
           .
           .
           .=01FA                        ; Vector table (within 256 byte
           .ADDRW SERVICE                ; of VIS inst.) containing the ext
                                         ; interrupt service routine
           .
           .
INT_EXIT:
           RETI
           .
           .
SERVICE:   RBIT    EXPND, PSW            ; Interrupt Service Routine
                                         ; Reset ext interrupt pend. bit
           .
           .
           .
           JP   INT_EXIT                 ; Return, set the GIE bit
```

Interrupts (Continued)

NON-MASKABLE INTERRUPT

Pending Flag

There is a pending flag bit associated with the non-maskable interrupt, called STPND. This pending flag is not memory-mapped and cannot be accessed directly by the software.

The pending flag is reset to zero when a device Reset occurs. When the non-maskable interrupt occurs, the associated pending bit is set to 1. The interrupt service routine should contain an RPND instruction to reset the pending flag to zero. The RPND instruction always resets the STPND flag.

Software Trap

The Software Trap is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from program memory and placed in the instruction register. This can happen in a variety of ways, usually because of an error condition. Some examples of causes are listed below.

If the program counter incorrectly points to a memory location beyond the available program memory space, the nonexistent or unused memory location returns zeroes which is interpreted as the INTR instruction.

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

A Software Trap can be triggered by a temporary hardware condition such as a brownout or power supply glitch.

The Software Trap has the highest priority of all interrupts. When a Software Trap occurs, the STPND bit is set. The GIE bit is not affected and the pending bit (not accessible by the user) is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. Nothing can interrupt a Software Trap service routine except for another Software Trap. The STPND can be reset only by the RPND instruction or a chip Reset.

The Software Trap indicates an unusual or unknown error condition. Generally, returning to normal execution at the point where the Software Trap occurred cannot be done reliably. Therefore, the Software Trap service routine should reinitialize the stack pointer and perform a recovery procedure that restarts the software at some known point, similar to a device Reset, but not necessarily performing all the same functions as a device Reset. The routine must also execute the RPND instruction to reset the STPND flag. Otherwise, all other interrupts will be locked out. To the extent possible, the interrupt routine should record or indicate the context of the device so that the cause of the Software Trap can be determined.

If the user wishes to return to normal execution from the point at which the Software Trap was triggered, the user must first execute RPND, followed by RETSK rather than RETI or RET. This is because the return address stored on the stack is the address of the INTR instruction that triggered the interrupt. The program must skip that instruction in order to proceed with the next one. Otherwise, an infinite loop of Software Traps and returns will occur.

Programming a return to normal execution requires careful consideration. If the Software Trap routine is interrupted by another Software Trap, the RPND instruction in the service routine for the second Software Trap will reset the STPND flag; upon return to the first Software Trap routine, the STPND flag will have the wrong state. This will allow maskable interrupts to be acknowledged during the servicing of the first Software Trap. To avoid problems such as this, the user program should contain the Software Trap routine to perform a recovery procedure rather than a return to normal execution.

Under normal conditions, the STPND flag is reset by a RPND instruction in the Software Trap service routine. If a programming error or hardware condition (brownout, power supply glitch, etc.) sets the STPND flag without providing a way for it to be cleared, all other interrupts will be locked out. To alleviate this condition, the user can use extra RPND instructions in the main program and in the WATCHDOG service routine (if present). There is no harm in executing extra RPND instructions in these parts of the program.

PORT L INTERRUPTS

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

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

The GIE (Global Interrupt Enable) bit enables the interrupt function.

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

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

INTERRUPT SUMMARY

The device uses the following types of interrupts, listed below in order of priority:

1. The Software Trap non-maskable interrupt, triggered by the INTR (00 opcode) instruction. The Software Trap is acknowledged immediately. This interrupt service routine can be interrupted only by another Software Trap. The Software Trap should end with two RPND instructions followed by a restart procedure.
2. Maskable interrupts, triggered by an on-chip peripheral block or an external device connected to the device. Under ordinary conditions, a maskable interrupt will not interrupt any other interrupt routine in progress. A maskable interrupt routine in progress can be interrupted by the non-maskable interrupt request. A maskable interrupt routine should end with an RETI instruction or, prior to restoring context, should return to execute the VIS instruction. This is particularly useful when exiting long interrupt service routiness if the time between interrupts is short. In this case the RETI instruction would only be executed when the default VIS routine is reached.

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

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 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 address 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 34* shows a block diagram of the MICROWIRE/PLUS logic.

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

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. *Table 11* 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 35* 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.

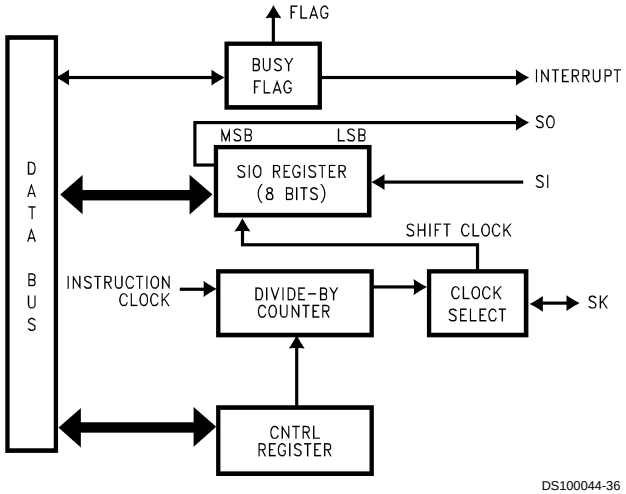


FIGURE 34. MICROWIRE/PLUS Block Diagram

MICROWIRE/PLUS (Continued)

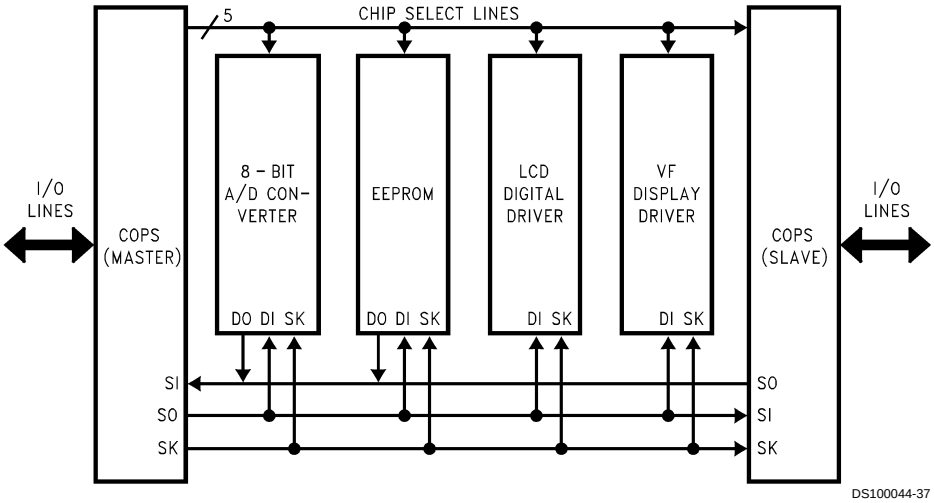


FIGURE 35. MICROWIRE/PLUS Application

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 configuraiton register. *Table 11* summarizes the bit settings required for Master or Slave mode of operation.

TABLE 11. MICROWIRE/PLUS Master Mode Clock Selection

SL1	SL0	SK
0	0	2 X t_c
0	1	4 X t_c
1	x	8 X t_c

Where t_c is the instruction cycle clock

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

The user must set the BUSY flag immediately upon entering the Salve 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 12. MICROWIRE/PLUS Mode Selection

This table assumes that the control flag MSEL is set.

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

Serial Peripheral Interface

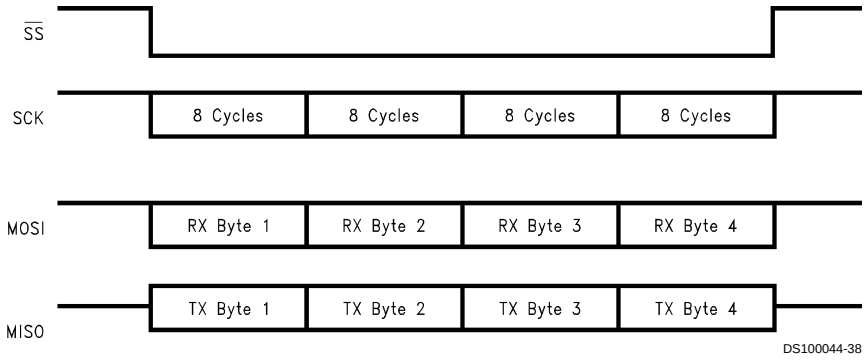


FIGURE 36. SPI Transmission Example

The Serial Peripheral Interface (SPI) is used in master-slave bus systems. It is a synchronous bidirectional serial communication interface with two data lines MISO and MOSI (**M**aster **I**n **S**lave **O**ut, **M**aster **O**ut **S**lave **I**n). A serial clock and a slave select ($\overline{SS}$) signal are always generated by the SPI Master. The interface receives/transmits protocol frames with up to 12 bytes length within a frame, where a frame is defined as the time between a falling edge and a rising edge of $\overline{SS}$.

THEORY OF OPERATION

Figure 38 shows a block diagram illustrating the basic operation of the SPI circuit. In the SPI interface, data is transmitted/received in packets of 8 bits length which are shifted into/out of a shift register with the active edge of the shift clock SCK. Two 12 byte FIFOs, which serve as a receive and a transmit buffer, allow a maximum message length of 12 x 8 bits in both transmit and receive direction without CPU intervention. With CPU intervention, many more bytes can be received. Two registers, the SPI Control Register (SPICNTL) and the SPI Status Register (SPISTAT), are used to control the SPI interface via the internal COP bus. Several different operation modes, such as master or slave operation, are possible.

An $\overline{SS}$-Expander allows the generation of up to 8 signals on the N-port, which can be used as additional $\overline{SS}$-signals ($\overline{ESS}$[7:0]) or as host programmable general purpose signals. The $\overline{SS}$-Expander is programmed with the content of the first MOSI-byte (i.e., the content of the 1st byte [7:0] appears at $\overline{ESS}$[7:0]) (N-port[7:0]), respectively), if the $\overline{ESS}$ programming mode is selected. The $\overline{ESS}$ programming mode is selected by the condition MOSI = L at the falling edge of $\overline{SS}$.

Use of the $\overline{ESS}$ expander requires the setup of four conditions by the user.

1. Set the SESSEN bit of SPICNTL.
2. Set PORTNX to select which bits are used for $\overline{SS}$ expansion.
3. Configure the PORTNC register to enable the desired $\overline{SS}$ expansion bits as outputs.
4. Have an $\overline{ESS}$ condition (MOSI = low at the falling edge of $\overline{SS}$).

Loop Back Mode

Setting the SLOOP bit enables the Loop Back mode, which can be used for test purposes. If the Loop Back mode is selected, TX FIFO data are communicated to the RX FIFO via the SPI Register. In the slave mode, MISO output is internally connected to the MOSI input. In the master mode, the MOSI output is internally connected to the MISO input.

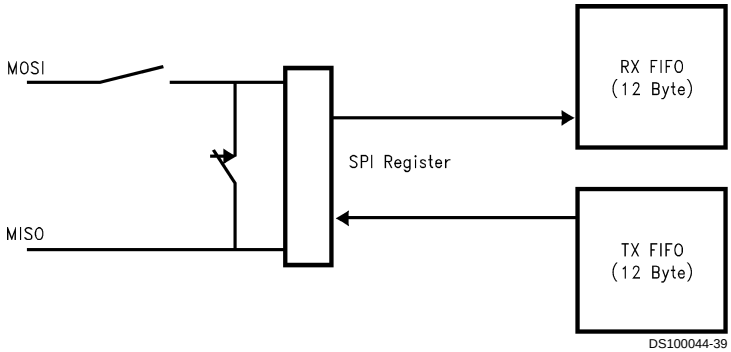


FIGURE 37. Loop Back Mode Block Diagram

Serial Peripheral Interface (Continued)

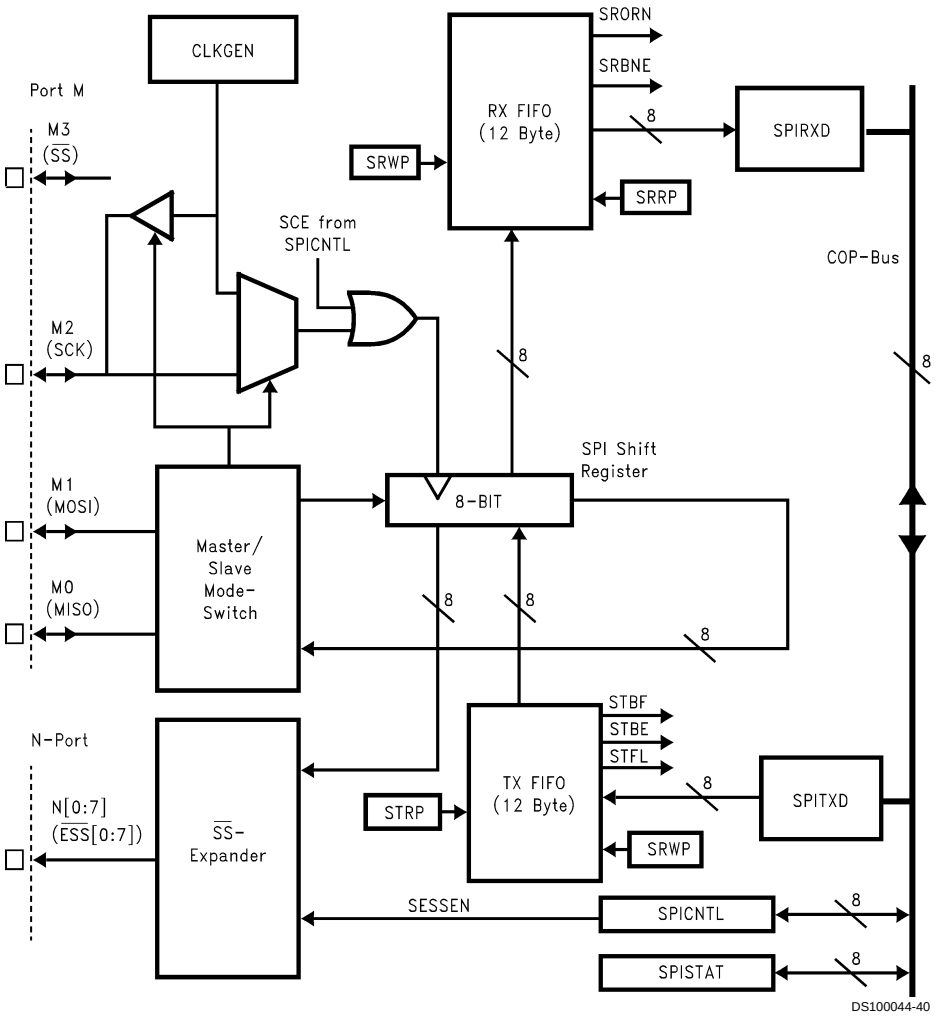


FIGURE 38. SPI Block Diagram

The SPIU Control Register

TABLE 13. SPI Control (SPICNTL) (0098)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SRIE	STIE	SESSEN	SPIMOD[1:0]		SCE	SPIEN	SLOOP
0	0	0	0	0	0	0	0

B7	SRIE	**S**PI **R**eceive **I**nterrupt **E**nable 0—disable receive interrupt 0—enable receive interrupt
B6	STIE	**S**PI **T**ransmit buffer **I**nterrupt **E**nable 0—disable transmit buffer interrupt 0—enable transmit buffer interrupt
B5	SESSEN	**S**PI **$\overline{SS}$** Expander ($\overline{ESS}$) enable 0—The detection of the $\overline{ESS}$ programming mode is disabled, i.e., the value of MOSI at the falling edge of $\overline{SS}$ is "don't care". 1—$\overline{ESS}$ programming mode detection is enabled, i.e., if the condition "MOSI = 0 at the falling edge of $\overline{SS}$" occurs, the $\overline{SS}$-Expander is selected and bits [7:0] of the first transmitted byte determine the state of the N-port ($\overline{ESS}$[7:0]). $\overline{ESS}$[7:0] will go 1 at the positive edge of $\overline{SS}$.

Serial Peripheral Interface (Continued)

B[4:3]	SPIMOD[1:0]	**SPI** operation **mode** select SPIMOD[1:0] 0 0: Slave mode, —SCK is SPI clock input —MISO is SPI data output —MOSI is SPI data input —$\overline{SS}$ is slave select input 1 0: Standard Master mode, —SCK is SPI clock output (CKI/40) —MISO is SPI data input —MOSI is SPI data output —$\overline{SS}$ is slave select output In the Master mode, 3 different SPI clock frequencies are available: 0 1: $f_{SCK} = 1/(t_c) = CKI/10$ 1 0: $f_{SCK} = 1/(4\ t_c) = CKI/40$ 1 1: $f_{SCK} = 1/(16\ t_c) = CKI/160$
B2	SCE	**S**PI **a**ctive clock **e**dge select 0: data are shifted out on the falling edge of SCK and are shifted in on the rising edge of SCK 1: data are shifted out on the rising edge of SCK and are shifted in on the falling edge of SCK
B1	SPIEN	**SPI en**able Enables the SPI interface and the alternate functions of the MISO, MOSI, SCK and $\overline{SS}$ pins. 0: disable SPI 1: enable SPI, all Port M $\overline{ESS}$ signals are set to 1
B0	SLOOP	**S**PI **loop** back mode 0: disable loop back mode 1: enable loop back mode, MISO and MOSI are internally connected (see *Figure 39*)

PROGRAMMING THE SPI EXPANDER

If the $\overline{SS}$ Expander is enabled by setting SESSEN = 1 in the SPI Control Register (SPICNTL), the N-port will be programmed with the content of the first MOSI-byte (i.e., the content of the 1st byte [7:0] appears at N-port[7:0] after complete reception of the first byte), if the $\overline{ESS}$ programming mode is detected. If any bytes follow after the 1st MOSI byte, all data will be ignored by the SPI.

The $\overline{ESS}$ programming mode is detected by the $\overline{ESS}$ control logic, which decodes the condition "MOSI = L at the falling edge of $\overline{SS}$. For further details, see *Figure 39*.

The selected N-port bits will be set to 1 after the positive edge of $\overline{SS}$.

Single N-port bits may be enabled for use as $\overline{SS}$ expansion, or disabled to allow for general purpose I/O, by the respective bits in the PORTNX register.

Serial Peripheral Interface (Continued)

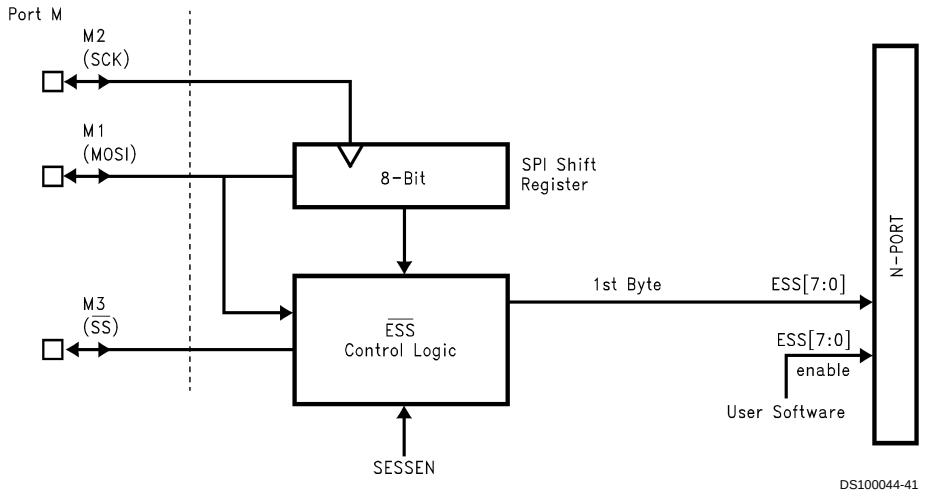
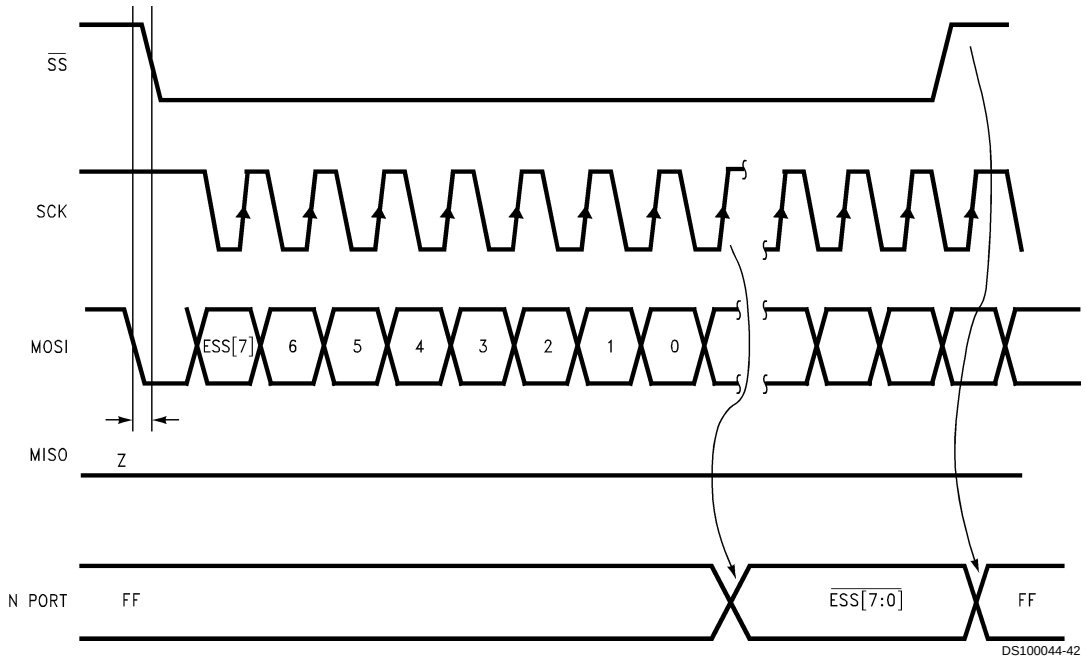


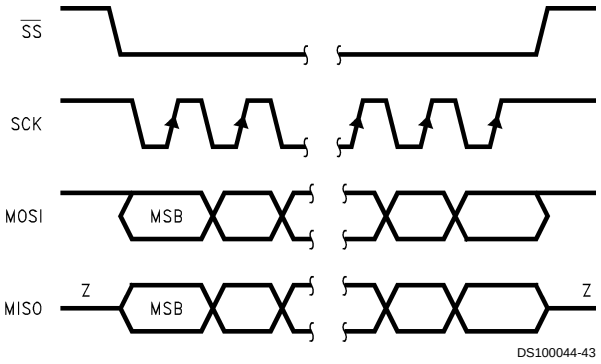
FIGURE 39. Programming the SPI Expander



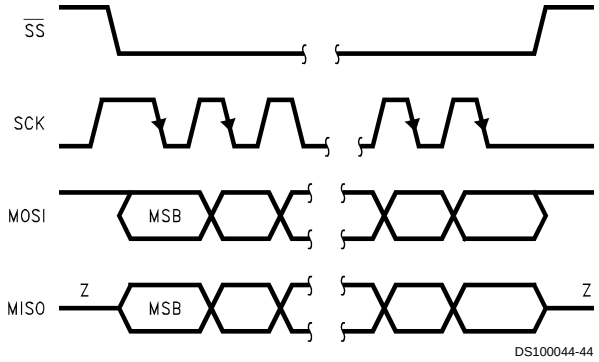
SESSEN = 1, SCE = 0. If MOSI = 0 at the falling edge of $\overline{SS}$, the $\overline{ESS}$ programming mode is detected and all N-port alternate functions are enabled.

FIGURE 40. Programming the $\overline{SS}$ Expander

SPI Status Register

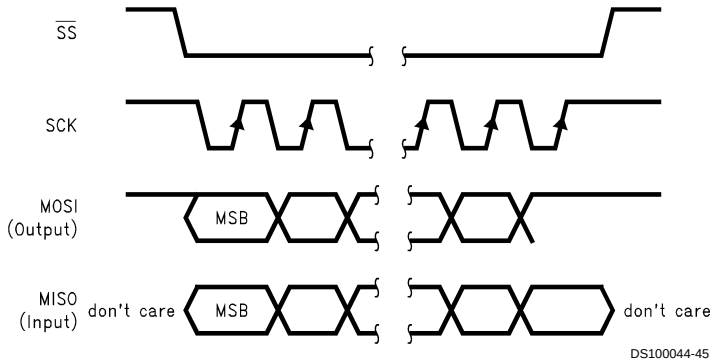


a) Slave mode; rising SCK edge is active edge. (SPIMOD[1,0] = [0,0], SCE = 0)

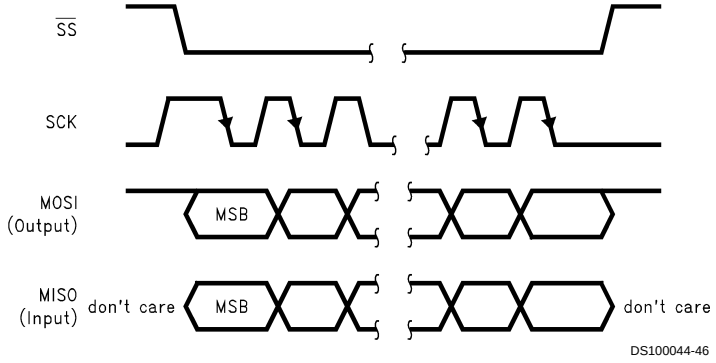


b) Slave mode; falling SCK edge is active edge. (SPIMOD[1,0] = [0,0], SCE = 1)

FIGURE 41. Slave Mode Communication



a) Master mode; rising SCK edge is active edge. (SPIMOD[1,0] = [1,0], SCE = 0)



b) Master mode; falling SCK edge is active edge. (SPIMOD[1,0] = [1,0], SCE = 1)

FIGURE 42. Master Mode Communication

SPI Status Register (Continued)

TABLE 14. SPI Status Register (SPISTAT) (0099)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SRORN	SRBNE	STBF	STBE	STFL	SESSDET	x	x
0	0	1	1	0	0		0

The SPI Status Register is a read only register.

B7	SRORN	**S**PI **r**eceiver **o**verrun. This bit is set on the attempt to overwrite valid data in the RX FIFO by the SPI interface. (The condition to detect this is: SRWP = SRRP & COP has not read the data at SRRP and attempting to write to the RX FIFO by the SPI interface.) This bit can generate a receive interrupt if the receive interrupt is enabled (SRIE = 1). (Notes 18, 19, 20)
B6	SRBNE	**S**PI **R**eceive **b**uffer **n**ot **e**mpty This bit is set with a write to the SPI RX FIFO resulting in SRWP ! = SRRP (caution at rollover!). This bit is reset with the read of the SPIRXD register resulting in SRWP to be equal to SRRP.
B5	STBF	This bit can generate a receive interrupt if enabled with the RIE bit. **S**PI **T**ransmit **b**uffer **f**ull This bit is set after a write operation to the SPITXD register (from the COP side), which results in STRP = STWP. It gets reset as soon as the STRP gets incremented - by the SPI if reading data out of the TX FIFO.
B4	STBE	**S**PI **t**ransmit **b**uffer **e**mpty This bit is set after the last bit of the a read from the SPITXD register, which results in STRP = STWP. It gets reset as soon as the STWP gets incremented - by the COP if writing data into the TX FIFO. It is set on reset.
B3	STFL	**S**PI **T**ransmit **b**uffer **f**lush This bit indicates that the contents of the transmit buffer got discharged by the $\overline{SS}$ signal becoming high before all data in the transmit buffer could be transmitted. This bit gets set if the $\overline{SS}$ signal gets high and 1.STRP ! = STWP **or** 2.STRP = STWP and the current byte has not been completely transmitted from the SPI shift register These conditions will reset STRP and STWP to 0. These are virtual pointers and cannot be viewed. (Note 21)
B2	SESSDET	**S**PI $\overline{\textbf{SS}}$ **E**xpander **d**etection This bit indicates the detection of a $\overline{SS}$ expand condition (MOSI = 0 at the falling edge of $\overline{SS}$) immediately after the N-port has been programmed (8th SCK bit, 8 µs at SCK = 1 MHz). This bit is reset at the rising edge of $\overline{SS}$. 1: $\overline{SS}$ expand condition detected. 0: normal communication. (Note 22)
B1		Reserved
B0		Reserved

Note 18: At this condition the write operation will not be executed and all data get lost.

Note 19: The SRORN bit stays set until the reset condition.

This bit is reset with a dummy write to the SPISTAT register. (As the register is read only a dummy write does not have any effect on any other bits in this register.)

As a result of the SRORN condition, the SRWP becomes frozen (i.e., does not change until the SRORN bit is reset) and the SPI will not store any new data in the RX FIFO.

Note 20: With the SRRP being still available, the user can read the data in the RX FIFO before resetting the SRORN bit.

Note 21: STRP = STWP & STBE = 1 will generate an interrupt.

This bit gets reset with a write to the SPITXD register.

Note 22: The SPI master must hold $\overline{SS}$ = 0 long enough to allow the device to read SESSDET. Otherwise the SESSDET information will get lost.

SPI Status Register (Continued)

SPI SYNCHRONIZATION

After the SPI is enabled (SPIEN = 1), the SPI internal receive and transmit shift clock is kept disabled until $\overline{SS}$ becomes inactive. This includes $\overline{SS}$ being active at the time SPIEN is set, i.e., no receive/transmit is possible until $\overline{SS}$ becomes inactive after enabling the SPI.

HALT/IDLE MODE

If the device enters the HALT/IDLE mode, both RX and TX FIFOs get reset (Flushed). If the device is exiting HALT/IDLE mode, and SPI synchronization takes place as described above. SPIRXD and SPITXD have the same state as after Reset, SPISTAT bits after HALT/IDLE mode are:

SRORN: unchanged

SRBNE: 0

STBF: 0

STBE: 1

STFL: 1

SESSDET: x (depending on $\overline{SS}$ and MOSI line)

TRANSMISSION START IN MASTER MODE

The transmission of data in the Master mode is started if the user controlled $\overline{SS}$ signal is switched active. No SCK will be generated in Master mode and thus no data is transmitted if the $\overline{SS}$ signal is kept high, i.e., $\overline{SS}$ must be switched low to generate SCK. Resetting the $\overline{SS}$ signal in the Master mode will immediately stop the transmission and flush the transmit FIFO. Thus, the user must only reset the $\overline{SS}$ if:

1. TBE is set or
2. SCK is high (SCE = 0) or low (SCE = 1)

TX AND RX FIFO

If the SPI is disabled (SPIEN = 0), all SPI FIFO related pointers are reset and kept at zero until the SPI is enabled again. Also, the Read/Write operation to both SPITXD and SPIRXD will not cause the pointers to change, if SPIEN is set, Read operations from the RXFIFO and Write operation to TXFIFO will increment the respective Read/Write pointers.

SPIRXD SPI Receive Data Register

SPIRXD is at address location “009A”. It is a read/write register.

This register holds the receive data at the current SRRP location: a COP read operation from this register to the accumulator will read the RX FIFO at the SRRP location and increment SRRP afterwards. A write to this register (by the controllers SW) will write to the RX FIFO at the current SRRP location. The SRRP is not changed.

Note: During breakpoint the SRRP is not incremented.

A write to this register from the SPI interface side will write to the current SRWP location and increment SRWP afterwards.

SPITXD SPI Transmit Data Register

SPITXD is at address location “009B”. It is a read/write register.

This register holds the transmit data at the current STWP location: a write from the controller to this register will write to the STWP location and increment the STWP afterwards. A read from the controller to this register will read the TX FIFO at the current STWP location. The pointer is not changed.

Writing data into this register will start a transmission of data in the master mode.

Note: No read modify write instructions should be used on this register.

Reading this register from the SPI side will read the byte at the current STRP location and afterwards increment STRP.

SPI RX FIFO

The SPI RX FIFO is a 12 byte first in first out buffer. SPI RX FIFO data are read from the controller by reading the SPIRXD register. A pointer (SRRP) controls the controller read location. Data is written to this register by the SPI interface. The write location is controlled by the SRWP. SRWP is incremented after data is stored to the FIFO SRWP is never decremented SRWP has a roll-over 10 → 11 → 0 → 1 → 2 → etc. It is a circularly linked list.

SRRP is incremented after data is read from the FIFO SRRP is never decremented SRRP has a roll-over 10 → 11 → 0 → 1 → 2 → etc. It is a circularly linked list.

Both pointers are cleared at reset.

The following bits indicate the status of the RX FIFO: SRBNE = (SRWP != SRRP) and !SRORN .SRORN is set at (SRWP = SRRP) and after a write from the SPI side, reset at write to SPISTAT.

Special conditions: if .SRORN is set, no writes to the RX FIFO are allowed from the SPI side. SRWP is frozen. Resetting. SRORN (after it was set) clears both SRWP and SRRP. To prevent erroneous clearing of the Receive FIFO when entering HALT/IDLE mode, the user needs to enable the MIWU or port M3 ($\overline{SS}$) by setting bit 3 in MWKEN register.

SPI TX FIFO

The SPI TX FIFO is a 12 byte first in first out buffer. Data is written to the FIFO by the controller executing a write instruction to the SPITXD register. A pointer (STWP) controls the controller write location. Data is read from this register by the SPI interface. The read location is controlled by the STRP. STRP is incremented after data is read from the FIFO STRP is never decremented STRP has a roll-over 10 → 11 → 0 → 1 → 2 → etc. It is a circularly linked list.

STWP is incremented after data is written to the FIFO STWP is never decremented STWP has a roll-over 10 → 11 → 0 → 1 → 2 → etc. It is a circularly linked list.

Both pointers are cleared at reset.

The following bits indicate the status of the TX FIFO: STBF = set at (STRP = STWP) after a write from the controller reset at ((STRP != STWP) I STBE) after a read from the SPI STBE = (STRP = STWP) after a read from the SPI.

Special conditions: If the $\overline{SS}$ signal becomes high before data the last bit of the last byte in the TX FIFO is transmitted both STRP and STWP will be set to 0. The STFL bit will be set. (STBE will be set as well.)

Note: The SRRP, SRWP, STRP and STWP registers are not available to the user. Their operation description is included for clarity and to enhance the user’s understanding.

A/D Converter

The device contains an 8-channel, multiplexed input, successive approximation, Analog-to Digital converter. The device’s VCC and GND pins are used for voltage reference.

A/D Converter (Continued)

OPERATING MODES

The A/D converter supports ratiometric measurements. It supports both Single Ended and Differential modes of operation.

Four specific analog channel selection modes are supported. These are as follows:

Allow any specific channel to be selected at one time. The A/D converter performs the specific conversion requested and stops.

Allow any specific channel to be scanned continuously. In other words, the user specifies the channel and the A/D converter scans it continuously. At any arbitrary time the user can immediately read the result of the last conversion. The user must wait for only the first conversion to complete.

Allow any differential channel pair to be selected at one time. The A/D converter performs the specific differential conversion requested and stops.

Allow any differential channel pair to be scanned continuously. In other words, the user specifies the differential channel pair and the A/D converter scans it continuously. At any arbitrary time the user can immediately read the result of the last differential conversion. The user must wait for only the first conversion to complete.

The A/D converter is supported by two memory mapped registers, the result register and the mode control register. When the device is reset, the mode control register (ENAD) is cleared, the A/D is powered down and the A/D result register has unknown data.

A/D Control Register

The ENAD control register contains 3 bits for channel selection, 2 bits for prescaler selection, 2 bits for mode selection and a Busy bit. An A/D conversion is initiated by setting the ADBSY bit in the ENAD control register. The result of the conversion is available to the user in the A/D result register, ADRSLT, when ADBSY is cleared by the hardware on completion of the conversion.

ENAD (Address 0xCB)

CHANNEL SELECT			MODE SELECT		PRESCALER SELECT		BUSY
ADCH2	ADCH1	ADCH0	ADMOD1	ADMOD0	PSC1	PSC0	ADBSY
Bit 7							Bit 0

CHANNEL SELECT

This 3-bit field selects one of eight channels to be the V_{IN+}. The mode selection determines the V_{IN-} input.

Single Ended mode:

Bit 7	Bit 6	Bit 5	Channel No.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Differential mode:

Bit 7	Bit 6	Bit 5	Channel Pairs (+, −)
0	0	0	0, 1
0	0	1	1, 0
0	1	0	2, 3
0	1	1	3, 2
1	0	0	4, 5
1	0	1	5, 4
1	1	0	6, 7
1	1	1	7, 6

MODE SELECT

This 2-bit field is used to select the mode of operation (single conversion, continuous conversions, differential, single ended) as shown in the following table.

Bit 4	Bit 3	Mode
0	0	Single Ended mode, single conversion
0	1	Single Ended mode, continuous scan of a single channel into the result register
1	0	Differential mode, single conversion
1	1	Differential mode, continuous scan of a channel pair into the result register

PRESCALER SELECT

This 2-bit field is used to select one of the four prescaler clocks for the A/D converter. The following table shows the various prescaler options.

A/D Converter Clock Prescale

Bit 2	Bit 1	Clock Select
0	0	Divide by 2
0	1	Divide by 4
1	0	Divide by 6
1	1	Divide by 12

BUSY BIT

The ADBSY bit of the ENAD register is used to control starting and stopping of the A/D conversion. When ADBSY is cleared, the prescale logic is disabled and the A/D clock is turned off. Setting the ADBSY bit starts the A/D clock and initiates a conversion based on the mode select value currently in the ENAD register. Normal completion of an A/D conversion clears the ADBSY bit and turns off the A/D converter.

The ADBSY bit remains a one during continuous conversion. The user can stop continuous conversion by writing a zero to the ADBSY bit.

If the user wishes to restart a conversion which is already in progress, this can be accomplished only by writing a zero to the ADBSY bit to stop the current conversion and then by writing a one to ADBSY to start a new conversion. This can be done in two consecutive instructions.

ADC Operation

The A/D converter interface works as follows. Setting the ADBSY bit in the A/D control register ENAD initiates an A/D conversion. The conversion sequence starts at the beginning of the write to ENAD operation which sets ADBSY, thus powering up the A/D. At the first falling edge of the converter clock following the write operation, the sample signal turns

A/D Converter (Continued)

on for seven clock cycles. If the A/D is in single conversion mode, the conversion complete signal from the A/D will generate a power down for the A/D converter and will clear the ADBSY bit in the ENAD register at the next instruction cycle boundary. If the A/D is in continuous mode, the conversion complete signal will restart the conversion sequence by deselecting the A/D for one converter clock cycle before starting the next sample. The A/D 8-bit result is immediately loaded into the A/D result register (ADRSLT) upon completion. Internal logic prevents transient data (resulting from the A/D writing a new result over an old one) being read from ADRSLT.

Inadvertent changes to the ENAD register during conversion are prevented by the control logic of the A/D. Any attempt to write any bit of the ENAD Register except ADBSY, while ADBSY is a one, is ignored. ADBSY must be cleared either by completion of an A/D conversion or by the user before the prescaler, conversion mode or channel select values can be changed. After stopping the current conversion, the user can load different values for the prescaler, conversion mode or channel select and start a new conversion in one instruction.

It is important for the user to realize that, when used in differential mode, only the positive input to the A/D converter is sampled and held. The negative input is constantly connected and should be held stable for the duration of the conversion. Failure to maintain a stable negative input will result in incorrect conversion.

PRESCALER

The A/D Converter (A/D) contains a prescaler option that allows four different clock selections. The A/D clock frequency is equal to CKI divided by the prescaler value. Note that the prescaler value must be chosen such that the A/D clock falls within the specified range. The maximum A/D frequency is 1.67 MHz. This equates to a 600 ns A/D clock cycle.

The A/D converter takes 17 A/D clock cycles to complete a conversion. Thus the minimum A/D conversion time for the device is 10.2 µs when a prescaler of 6 has been selected. The 17 A/D clock cycles needed for conversion consist of 1 cycle at the beginning for reset, 7 cycles for sampling, 8 cycles for converting, and 1 cycle for loading the result into the A/D result register (ADRSLT). This A/D result register is a read-only register. The user cannot write into ADRSLT.

The ADBSY flag provides an A/D clock inhibit function, which saves power by powering down the A/D when it is not in use.

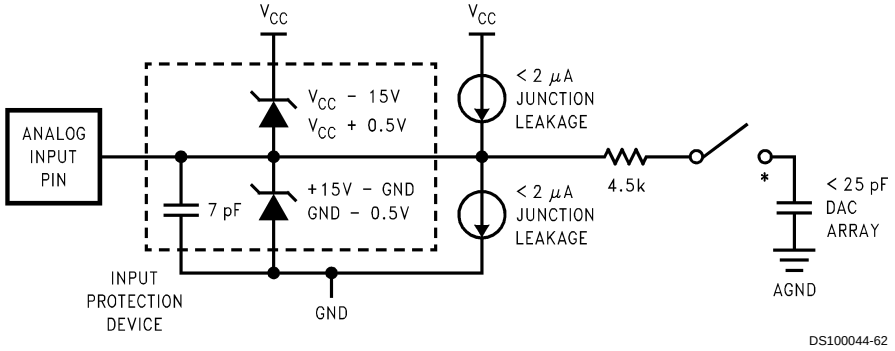
Note: The A/D converter is also powered down when the device is in either the HALT or IDLE modes. If the A/D is running when the device enters the HALT or IDLE modes, the A/D powers down and then restarts the conversion with a corrupted sampled voltage (and thus an invalid result) when the device comes out of the HALT or IDLE modes.

Analog Input and Source Resistance Considerations

Figure 43 shows the A/D pin model in single ended mode. The differential mode has a similar A/D pin model. The leads to the analog inputs should be kept as short as possible. Both noise and digital clock coupling to an A/D input can cause conversion errors. The clock lead should be kept away from the analog input line to reduce coupling. The A/D channel input pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signals due to output buffer leakage current.

Source impedances greater than 3 kΩ on the analog input lines will adversely affect the internal RC charging time during input sampling. As shown in*Figure 43*, the analog switch to the DAC array is closed only during the 7 A/D cycle sample time. Large source impedances on the analog inputs may result in the DAC array not being charged to the correct voltage levels, causing scale errors.

If large source resistance is necessary, the recommended solution is to slow down the A/D clock speed in proportion to the source resistance. The A/D converter may be operated at the maximum speed for R_S less than 3 kΩ. For R_S greater than 3 kΩ, A/D clock speed needs to be reduced. For example, with R_S = 6 kΩ, the A/D converter may be operated at half the maximum speed. A/D converter clock speed may be slowed down by either increasing the A/D prescaler divide-by or decreasing the CKI clock frquency. The A/D minimum clock speed is 100 kHz.



*The analog switch is closed only during the sample time.

FIGURE 43. A/D Pin Model (Single Ended Mode)

USART

The device contains a full-duplex software programmable USART. The USART *Figure 44* consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a USART control and status register (ENU), a USART receive control and status register (ENUR), a USART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framing, data overrun and parity errors while the USART is receiving.

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

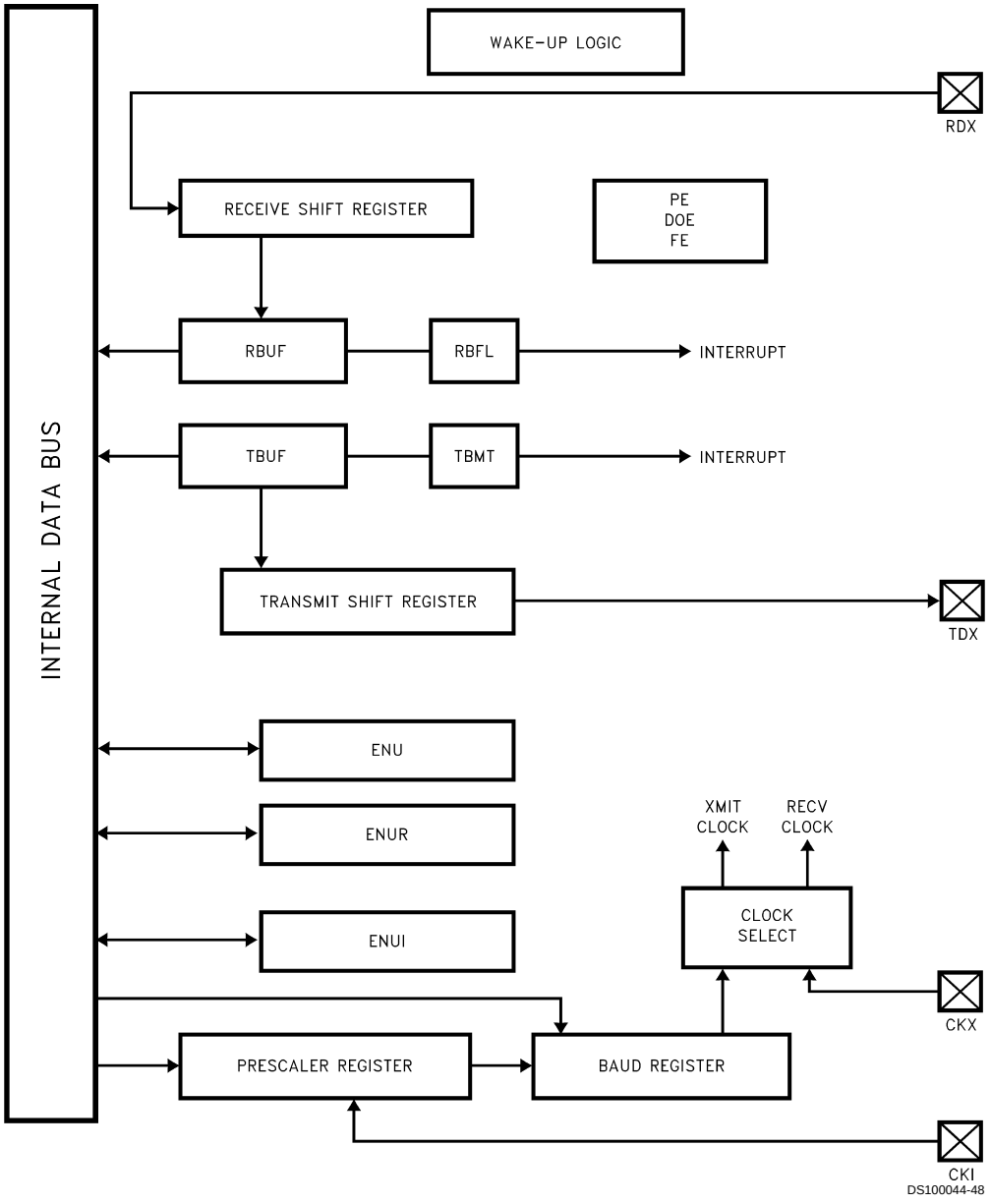


FIGURE 44. USART Block Diagram

USART (Continued)

USART CONTROL AND STATUS REGISTERS

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

DESCRIPTION OF USART REGISTER BITS

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

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

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

PEN = 0 Parity disabled.

PEN = 1 Parity enabled.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

DOE	FE	PE	Reserved	RBIT9	ATTN	XMTG	RCVG
Bit 7							Bit 0

Note 23: Bit is reserved for future use. User must set to zero.

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

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

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

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

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

FE = 1 Indicates the occurrence of a Framing Error.

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

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

PE = 1 Indicates the occurrence of a Parity Error.

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

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

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

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

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

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

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

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

STP2 = 0 One Stop bit transmitted.

STP2 = 1 Two Stop bits transmitted.

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

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

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

SSEL = 0 Asynchronous Mode.

SSEL = 1 Synchronous Mode.

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

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

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

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

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

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

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

USART (Continued)

ERI = 0 Interrupt from the receiver is disabled.

ERI = 1 Interrupt from the receiver is enabled.

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

ETI = 0 Interrupt from the transmitter is disabled.

ETI = 1 Interrupt from the transmitter is enabled.

Associated I/O Pins

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

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

USART Operation

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

ASYNCHRONOUS MODE

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

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

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

SYNCHRONOUS MODE

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

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

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

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

FRAMING FORMATS

The USART supports several serial framing formats (*Figure 45*) . The format is selected using control bits in the ENU, ENUR and ENUI registers.

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

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

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

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

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

USART Operation (Continued)

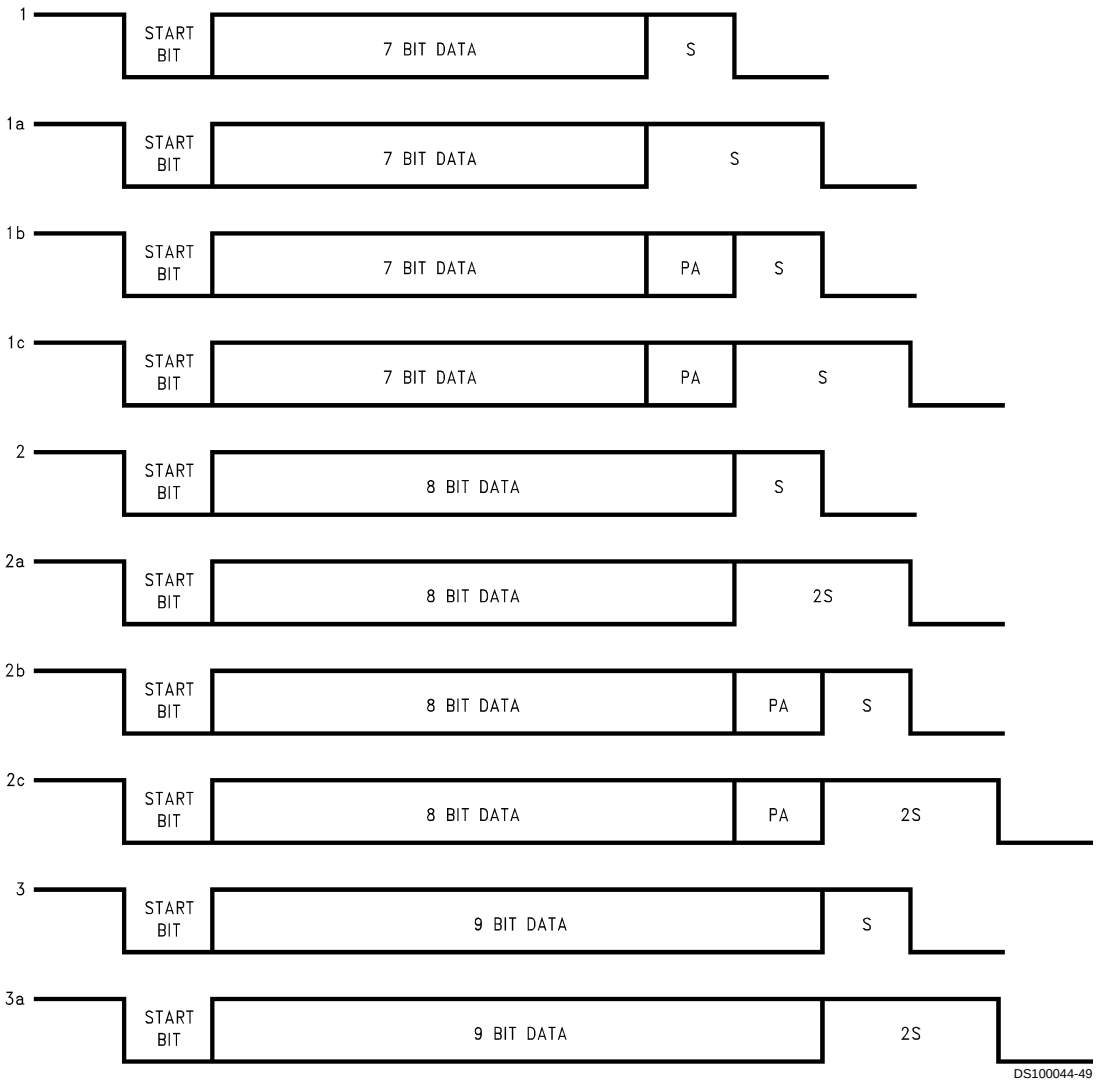


FIGURE 45. Framing Formats

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

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

USART INTERRUPTS

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

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

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

Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the USART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1–16 (increments of 0.5) prescaler and an 11-bit binary counter. (*Figure 46*) The divide factors are specified through two read/

Baud Clock Generation (Continued)

write registers shown in *Figure 47*. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in *Table 15*, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the USART power down mode where the USART clock is turned off for power saving purpose. The user must also turn the USART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in *Table 15*. There are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (*Table 16*). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

TABLE 15. Prescaler Factors

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

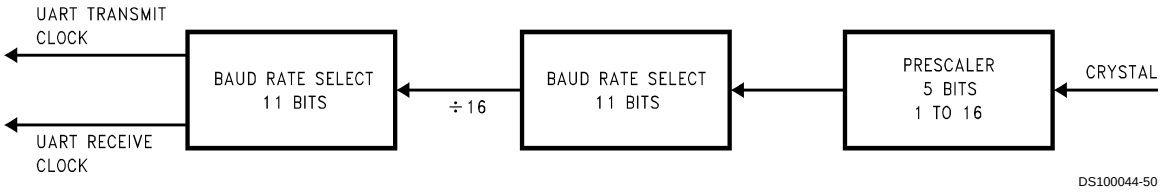


FIGURE 46. USART BAUD Clock Generation

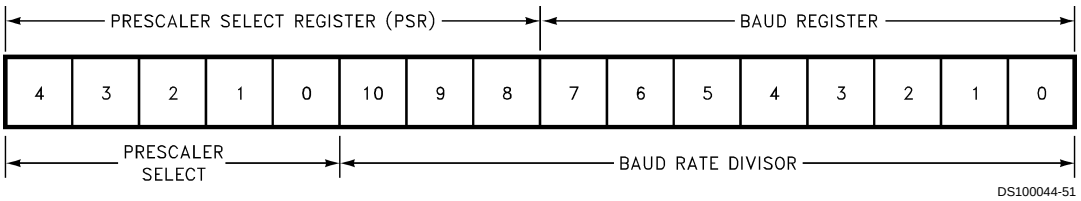


FIGURE 47. USART BAUD Clock Divisor Registers

Baud Clock Generation (Continued)

TABLE 16. Baud Rate Divisors (1.8432 MHz Prescaler Output)

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

Note: The entries in *Table 16* assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

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

4.608/1.8432 = 2.5

The 2.5 entry is available in *Table 15*. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (*Table 16*) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in *Table 16* is 5.

N − 1 = 5 (N − 1 is the value from *Table 16*)

N = 6 (N is the Baud Rate Divisor)

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

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

The actual Baud Rate may be found from:

BR = Fc/(16 X N X P)

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is the Baud Rate Divisior (*Table 16*).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (*Table 15*)

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

Example:

Asynchronous Mode:

Crystal Frequency = 5 MHz

Desired baud rate = 9600

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

N x P = (5 X 10^6)/(16 x 9600) = 32.552

Now 32.552 is divided by each Prescaler Factor (*Table 15*) to obtain a value closet to an integer. This factor happens to be 6.5 (P = 6.5).

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

The programmed value (from *Table 16*) should be 4 (N −1).

Using the above values calculated for N and P:

BR = (5 x 10^6)/(16 x 5 x 6.5) = 9615.384

% error = (9615.385 − 9600)/9600 * 100 = 0.16

Effect of HALT/IDLE

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

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

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

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

Diagnostic

Bits CHL0 and CHL1 in the ENU register provide a loopback feature for diagnostic testing of the USART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Register is “looped back” into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the USART.

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

Attention Mode

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

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

While in ATTENTION mode, the USART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the USART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if USART Receiver interrupts are enabled. The ATTN bit is also cleared

Attention Mode (Continued)

automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

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

WATCHDOG

The device contains a WATCHDOG and clock monitor. The WATCHDOG is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WATCHDOG consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WATCHDOG consists of writing a specific value to a WATCHDOG Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. *Table 18* shows the WDSVR register.

TABLE 17. WATCHDOG Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
Bit 7							Bit 0

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table 18 shows the four possible combinations of lower and upper limits for the WATCHDOG service window. This flexibility in choosing the WATCHDOG service window prevents any undue burden on the user software.

TABLE 18. WATCHDOG Service Window Select

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

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

Clock Monitor

The Clock Monitor aboard the device can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WATCHDOG Operation

The WATCHDOG and Clock Monitor are disabled during reset. The device comes out of reset with the WATCHDOG armed, the WATCHDOG Window Select (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WATCHDOG service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WATCHDOG service window and match the WATCHDOG key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WATCHDOG service window value and the key data (bits 7 through 1) in the WDSVR Register. *Table 19* shows the sequence of events that can occur.

The user must service the WATCHDOG at least once before the upper limit of the service window expires. The WATCHDOG may not be serviced more than once in every lower limit of the service window. The user may service the WATCHDOG as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WATCHDOG service.

The WATCHDOG has an output pin associated with it. This is the WDOUT pin, on pin 1 of the Port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WATCHDOG, the logic will pull the WDOUT (G1) pin low for an additonal 16 t_c–32 t_c cycle after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the device will stop forcing the WDOUT output low.

The WATCHDOG service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

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

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_c–32 t_c clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

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

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

WATCHDOG Operation (Continued)

WATCHDOG AND CLOCK MONITOR SUMMARY

The following salient points regarding the COP888 WATCHDOG and CLOCK MONITOR should be noted:

- Both the WATCHDOG and Clock Monitor detector circuits are inhibited during RESET.
- Following RESET, the WATCHDOG and CLOCK MONITOR are both enabled, with the WATCHDOG having the maximum service window selected.
- The WATCHDOG service window and Clock Monitor enable/disable option can only be changed once, during the initial WATCHDOG service following RESET.
- The initial WATCHDOG service must match the key data value in the WATCHDOG Service register WDSVR in order to avoid a WATCHDOG error.
- Subsequent WATCHDOG services must match all three data fields in WDSVR in order to avoid WATCHDOG errors.
- The correct key data value cannot be read from the WATCHDOG Service register WDSVR. Any attempt to read this key data value of 01100 from WDSVR will read as key data value of all 0's.
- The WATCHDOG detector circuit is inhibited during both the HALT and IDLE modes.
- The Clock Monitor detector circuit is active during both the HALT and IDLE modes. Consequently, the device inadvertently entering the HALT mode will be detected as a Clock Monitor error (provided that the Clock Monitor enable option has been selected by the program).
- With the single-pin R/C oscillator mask option selected and the CLKDLY bit reset, the WATCHDOG service window will resume following HALT mode from where it left off before entering the HALT mode.
- With the crystal oscillator mask option selected, or with the single-pin R/C oscillator mask option selected and the CLKDLY bit set, the WATCHDOG service window will be set to its selected value from WDSVR following HALT. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following HALT, but must be serviced within the selected window to avoid a WATCHDOG error.
- The IDLE timer T0 is not initialized with RESET.
- The user can sync in to the IDLE counter cycle with an IDLE counter (T0) interrupt or by monitoring the T0PND flag. The T0PND flag is set whenever the thirteenth bit of the IDLE counter toggles (every 4096 instruction cycles). The user is responsible for resetting the T0PND flag.
- A hardware WATCHDOG service occurs just as the device exits the IDLE mode. Consequently, the WATCHDOG should not be serviced for at least 2048 instruction cycles following IDLE, but must be serviced within the selected window to avoid a WATCHDOG error.
- Following RESET, the initial WATCHDOG service (where the service window and the CLOCK MONITOR enable/disable must be selected) may be programmed anywhere within the maximum service window (65,536 instruction cycles) initialized by RESET. Note that this initial WATCHDOG service may be programmed within the initial 2048 instruction cycles without causing a WATCHDOG error.

TABLE 19. WATCHDOG Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WATCHDOG Output
Mismatch	Don't Care	Don't Care	Error: Generate WATCHDOG Output
Don't Care	Don't Care	Mismatch	Error: Generate WATCHDOG Output

Memory Map

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

Address	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads as All Ones)
0080	PORTMD, Port M Data Register
0081	PORTMC, Port M Configuration Register
0082	PORTMP, Port M Input Pins (Read Only)
0083	Reserved for Port M
0084	MMIWU Edge Select Register (MWKEDG)
0085	MMIWU Enable Register (MWKEN)
0086	MMIWU Pending Register (MWKPND)
0087	Reserved for MMIWU
0088	PORTND, Port N Data Register
0089	PORTNC, Port N Configuration Register
008A	PORTNP, Port N Input Pins (Read Only)
008B	PORTNX, Port N Alternate Function Enable
008C to 008F	Unused RAM Address Space (Reads Undefined Data)
0090	PORTED, Port E Data Register
0091	PORTEC, Port E Configuration Register
0092	PORTEP, Port E Input Pins (Read Only)
0093	Reserved for Port E
0094	PORTFD, Port F Data Register
0095	PORTFC, Port F Configuration Register
0096	PORTFP, Port F Input Pins (Read Only)
0097	Reserved for Port F
0098	SPICNTL, SPI Control Register
0099	SPISTAT, SPI Status Register
009A	SPIRXD, SPI Current Receive Data (Read Only)
009B	SPITXD, SPI Transmit Data
009C to 009F	Reserved
00A0	TXD1, Transmit 1 Data
00A1	TXD2, Transmit 2 Data
00A2	TDLC, Transmit Data Length Code and Identifier Low
00A3	TID, Transmit Identifier High
00A4	RXD1, Receive Data 1
00A5	RXD2, Receive Data 2
00A6	RIDL, Receive Data Length Code
00A7	RID, Receive Identify HIgh
00A8	CSCAL, CAN Prescaler
00A9	CTIM, Bus Timing Register
00AA	CBUS, Bus Control Register
00AB	TCNTL, Transmit/Receive Control Register
00AC	RTSTAT Receive/Transmit Status Register
00AD	TEC, Transmit Error Count Register
00AE	REC, Receive Error Count Register
00AF	PLATST, CAN Bit Stream Processor Test Register
00B8	UART Transmit Buffer (TBUF)
00B9	UART Receive Buffer (RBUF)
00BA	UART Control Status (ENU)
00BB	UART Receive Control Status (ENUR)
00BC	UART Interrupt and Clock (ENUI)
00BD	UART Baud Register (BAUD)
00BE	UART Prescaler Register (PSR)
00BF	Reserved for UART
00C0	Timer T2 Lower Byte (TMR2LO)
00C1	Timer T2 Upper Byte (TMR2HI)
00C2	Timer T2 Autoload Register T2RA Lower Byte (T2RALO)
00C3	Timer T2 Autoload Register T2RA Upper Byte (T2RAHI)
00C4	Timer T2 Autoload Register T2RB Lower Byte (T2RBLO)
00C5	Timer T2 Autoload Register T2RB Upper Byte (T2RBHI)
00C6	Timer T2 Control Register (T2CNTRL)
00C7	WATCHDOG Service Register (Reg:WDSVR)
00C8	LMIWU Edge Select Register (LWKEDG)
00C9	LMIWU Enable Register (LWKEN)
00CA	LLMIWU Pending Register (LWKPND)

Memory Map (Continued)

Address	Contents
00CB	A/D Converter Control Register (Reg:ENAD)
00CC	A/D Converter Result Register (Reg:ADRSLT)
00CD to 00CE	Reserved
00CF	IDLE Timer Control Register (Reg:ITMR)
00D0	PORTLD, Port L Data Register
00D1	PORTLC, Port L Configuration Register
00D2	PORTLP, Port L Input Pins (Read Only)
00D3	Reserved for Port L
00D4	PORTGD, Port G Data Register
00D5	PORTGC, Port G Configuration Register
00D6	PORTGP, Port G Input Pins (Read Only)
00D7	Port I Input Pins (Read Only)
00D8	Port CD, Port C Data Register
00D9	Port CC, Port C Configuration Register
00DA	Port CP, Port C Input Pins (Read Only)
00DB	Reserved for Port C
00DC	Port D
00DD to 00DF	Reserved for Port D
00E0 to 00E5	Reserved for EE Control Registers
00E6	Timer T1 Autoload Register T1RB Lower Byte (T1BRLO)
00E7	Timer T1 Autoload Register T1RB Upper Byte (T1BRHI)
00E8	ICNTRL Register
00E9	MICROWIRE/PLUS Shift Register (SOIR)
00EA	Timer T1 Lower Byte (TMR1LO)
00EB	Timer T1 Upper Byte (TMR1HI)
00EC	Timer T1 Autoload Register T1RA Lower Byte (T1RALO)
00ED	Timer T1 Autoload Register T1RA Upper Byte (T1RAHI)
00EE	CNTRL, Control Register
00EF	PSW, Processor Status Word Register
00F0 to 00FB	On-Chip RAM Mapped as Registers
00FC	X Register
00FD	SP Register
00FE	B Register
00FF	S Register
0100 to 013F	On-Chip RAM Bytes (64 Bytes)
Reading memory locations 0070H–007FH will return all ones. Reading unused memory locations 00xxH–00xxH will return undefined data. Reading memory locations from other Segments (i.e. segment 2, segment 3, ...etc.) will return undefined data.	

Addressing Modes

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

OPERAND ADDRESSING MODES

Register Indirect

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

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

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

Direct

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

Immediate

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

Short Immediate

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

Indirect

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

TRANSFER OF CONTROL ADDRESSING MODES

Relative

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

Addressing Modes (Continued)

Absolute

The 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←X 1)
LD	A, [B]	LoaD A with Memory [B]	A←[B], (B←B 1)
LD	A,[X]	LoaD A with Memory [X]	A←[X], (X←X 1)
LD	[B],Imm	LoaD Memory [B] Immed.	[B]←Imm, (B←B 1)
CLR	A	CLeaR A	A←0
INC	A	INCrement A	A←A + 1
DEC	A	DECrement A	A←A − 1
LAID		Load A InDirect from ROM	A←ROM (PU,A)
DCOR	A	Decimal CORrect A	A←BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C→A7→…→A0→C
RLC	A	Rotate A Left thru C	C←A7←…←A0←C
SWAP	A	SWAP nibbles of A	A7…A4↔A3…A0
SC		Set C	C←1, HC←1
RC		Reset C	C←0, HC←0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP←SP + 1, A←[SP]
PUSH	A	PUSH A onto the stack	[SP]←A, SP←SP − 1
VIS		Vector to Interrupt Service Routine	PU←[VU], PL←[VL]
JMPL	Addr.	Jump absolute Long	PC←ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9…0←i (i = 12 bits)
JP	Disp.	Jump relative short	PC←PC + r (r is −31 to +32, except 1)

Instruction Set (Continued)

INSTRUCTION SET (Continued)

JSRL	Addr.	Jump SubRoutine Long	[SP]←PL, [SP−1]←PU,SP−2, PC←ii
JSR	Addr.	Jump SubRoutine	[SP]←PL, [SP−1]←PU,SP−2, PC9…0←i
JID		Jump InDirect	PL←ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL←[SP], PU←[SP−1]
RETSK		RETurn and SKip	SP + 2, PL←[SP],PU←[SP−1]
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 Set (Continued)

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, (Note *NO TARGET FOR FNXref NS13180*)	1/1	1/3	2/3		1/2	1/3	
LD A, (Note 24)	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				

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

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

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

Development Tools Support

OVERVIEW

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

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

SUMMARY OF TOOLS

COP8 Evaluation Tools

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

COP8 Integrated Software/Hardware Design Development Kits

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

COP8 Development Languages and Environments

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

COP8 Productivity Enhancement Tools

- **WCOP8 IDE:** Very Low cost IDE (Integrated Development Environment) from KKD. Supports COP8C, COP8-NSASM, COP8-MLSIM, DriveWay COP8, and MetaLink debugger under a common Windows Project Management environment. Code development, debug, and emulation tools can be launched from the project window framework.
- **DriveWay-COP8:** Low cost COP8 Peripherals Code Generation tool from Aisys Corporation. Automatically generates tested and documented C or Assembly source code modules containing I/O drivers and interrupt handlers for each on-chip peripheral. Application specific code can be inserted for customization using the integrated editor. (Compatible with COP8-NSASM, COP8C, and WCOP8 IDE.)
- **COP8-UTILS:** Free set of COP8 assembly code examples, device drivers, and utilities to speed up code development.
- **COP8-MLSIM:** Free Instruction Level Simulator tool for Windows. For testing and debugging software instructions only (No I/O or interrupt support).

Development Tools Support

(Continued)

COP8 Real-Time Emulation Tools

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

COP8 Device Programmer Support

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

TOOLS ORDERING NUMBERS FOR THE COP87L88EB/RB FAMILY DEVICES

Vendor	Tools	Order Number	Cost	Notes
National	COP8-NSEVAL	COP8-NSEVAL	Free	Web site download
	COP8-NSASM	COP8-NSASM	Free	Included in EPU and DM. Web site download
	COP8-MLSIM	COP8-MLSIM	Free	Included in EPU and DM. Web site download
	COP8-NSDEV	COP8-NSDEV	VL	Included in EPU and DM. Order CD from website
	COP8-EPU	Not available for this device		
	COP8-DM	Contact MetaLink		
	Development Devices	COP87L88EB/RB	VL	16k or 32k OTP devices
	OTP Programming Adapters	EDI -44+68PL/40D-ZAL-W-COP888EB	L	For programming 44/68 PLCC on any programmer. Contact EDI
	IM-COP8	Contact MetaLink		
MetaLink	COP8-EPU	Not available for this device		
	COP8-DM	DM4-COP8-888EB (10 MHz), plus PS-10, plus DM-COP8/xxx (ie. 44P)	M	Included p/s (PS-10), target cable of choice (PLCC; i.e. DM-COP8/40P), EDI 44/68 PLCC OTP adapter
	OTP Programming Adapters	EDI -44/68PL/40D-ZAL-W-COP888EB	L	For programming 44/68 PLCC
	IM-COP8	IM-COP8-AD-464 (-220) (10 MHz maximum)	H	Base unit 10 MHz; -220 = 220V; add probe card (required) and target adapter (if needed); included software and manuals
	IM Probe Card	PC-888EB44P5-AD-10	M	10 MHz 44 PLCC probe card; 2.5V to 6.0V
		PC-888EB68P5-AD-10	M	10 MHz 68 PLCC probe card; 2.5V to 6.0V
ICU	COP8-EVAL	Not available for this device		
KKD	WCOP8-IDE	WCOP8-IDE	VL	Included in EPU and DM
IAR	EWCOP8-xx	See summary above	L - H	Included all software and manuals
Byte Craft	COP8C	COP8C	M	Included all software and manuals
Aisys	DriveWay COP8	DriveWay COP8	L	Included all software and manuals
OTP Programmers		Contact vendors	L - H	For approved programmer listings and vendor information, go to our OTP support page at: www.national.com/cop8
Cost: Free; VL =< $100; L = $100 - $300; M = $300 - $1k; H = $1k - $3k; VH = $3k - $5k				

Development Tools Support (Continued)

WHERE TO GET TOOLS

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

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

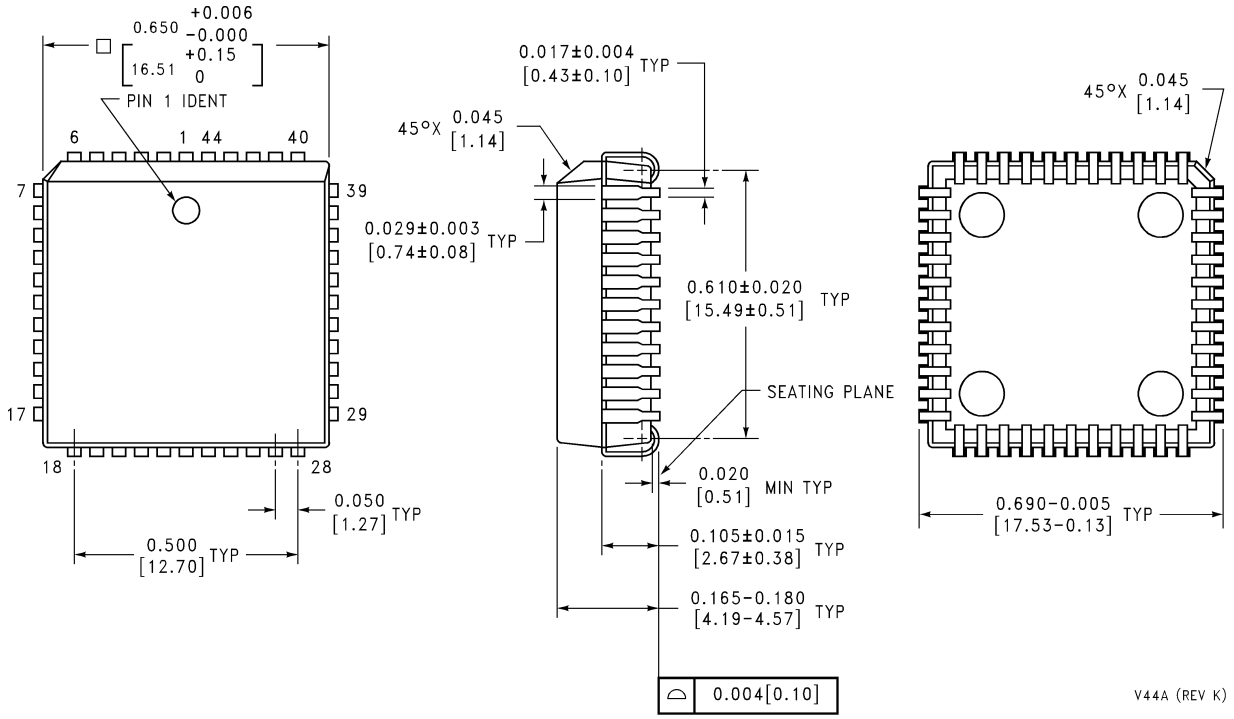
The following companies have approved COP8 programmers in a variety of configurations. Contact your local office or distributor. You can link to their web sites and get the latest listing of approved programmers from National's COP8 OTP Support page at: www.national.com/cop8.

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

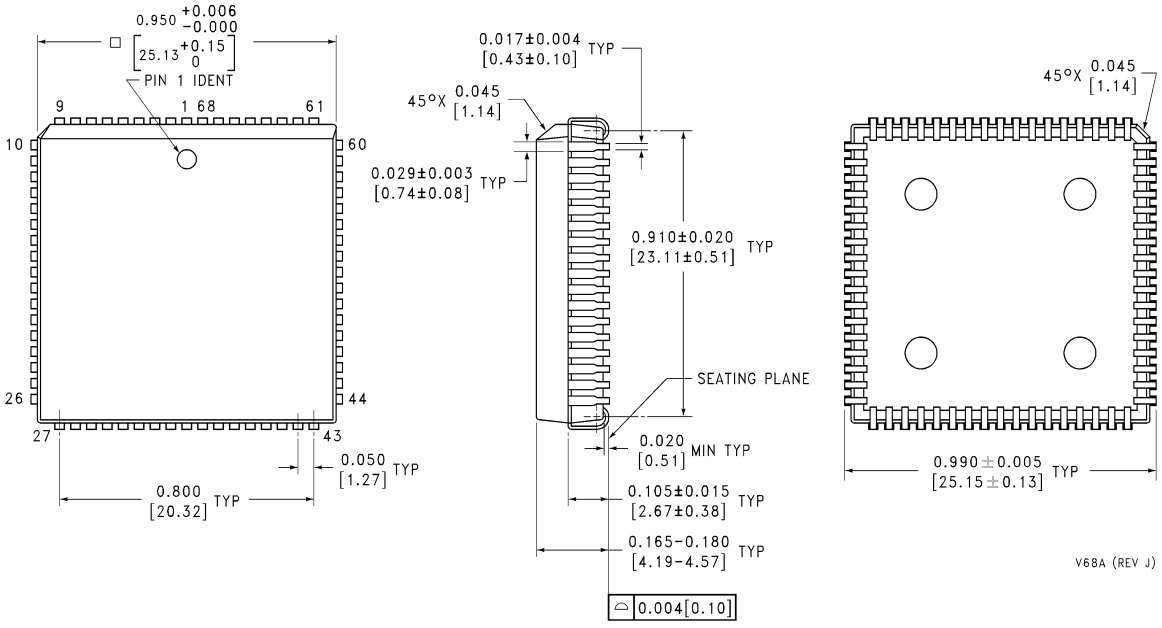
Customer Support

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

Physical Dimensions inches (millimeters) unless otherwise noted



44-Lead Molded Plastic Leaded Chip Carrier
Order Number COP87L88EBV-XE or COP87L88RBV-XE
NS Plastic Chip Package Number V44A



68-Lead Molded Plastic Leaded Chip Carrier
Order Number COP87L89EBV-XE or COP87L89RBV-XE
NS Plastic Chip Package Number V68A

Notes

LIFE SUPPORT POLICY

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

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

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

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 87 90

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

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Email: nsj.crc@jksmtp.nsc.com
Fax: 81-3-5639-7507