

Microchip Technology Inc.

dsPIC
Digital Signal Controller

Serial Communications using
the dsPIC30F CAN Module

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 1


Welcome to the “Serial Communications using the dsPIC30F CAN Module” web seminar.



Session Agenda

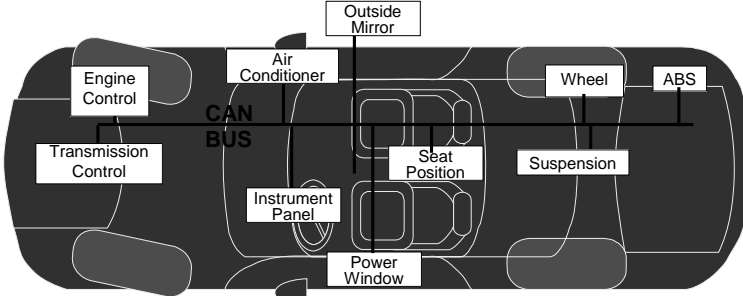
- CAN Protocol Overview
- Message Reception
- Message Transmission
- Bit Timing
- CAN Interrupts

In today's session, we will start by outlining some key features of the Controller Area Network, or CAN, module in the dsPIC30F family of devices. We will then delve deeper into the processes of data transmission and reception through the CAN interface, as well as bit timing considerations. Finally, we will study the interrupt and error management mechanisms built into the CAN module.


MICROCHIP
WebSeminars

CAN Protocol Overview

- The CAN bus is a serial communication protocol
- All nodes are connected together
- All nodes must use the same baud rate
- Each node can transmit or receive any message




© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module3

Controller Area Network, or CAN, is an industry standard serial communications protocol. The specifications for a CAN bus are described in the International Standards Organization specification ISO-11898.

All the nodes communicating on a CAN bus are connected to a common shared connection. Essentially, a CAN bus uses a star network topology. Often, this shared connection is physically implemented as a two wire differential pair for better noise immunity, thereby necessitating the use of a CAN transceiver device in conjunction with the CAN interface on a microcontroller.

All nodes communicating on a particular CAN bus must operate at the same baud rate. Generally, the system designer chooses one of several standard baud rates depending on the message latency requirements of the system. Some typical baud rates used in CAN are 1 Mega Hertz, 500 Kilo Hertz and 125 Kilo Hertz. In a CAN bus, each node can transmit or receive any message, which enables multicasting or broadcasting of messages as well as in-built message arbitration.

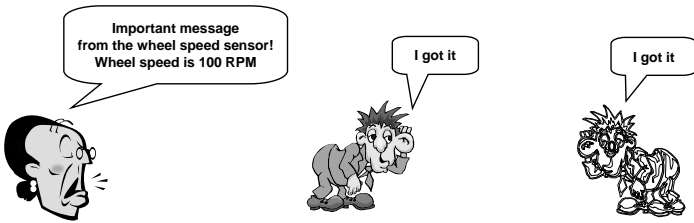
So how does the CAN bus arbitrate and prioritize the transmission of messages on the bus?



MICROCHIP
WebSeminars

CAN Protocol Overview

- Only one transmitter is allowed on the bus at a time
- CAN messages contain “**Identifier**” and “**Data Field**”
- The transmitter sends the message to all receivers
- All receivers will **acknowledge** reception of the message

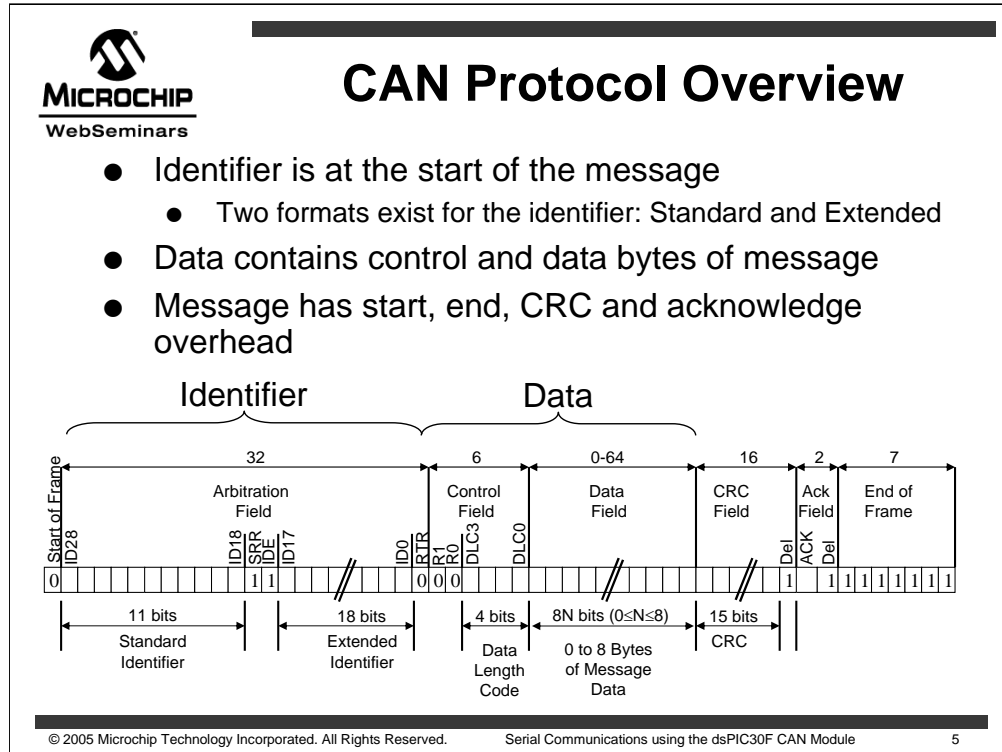


© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module4

On a CAN bus, although many nodes on the bus may have different messages to transmit, only one transmitter is allowed to transmit at a time.

Every CAN message contains a numerical Identifier and a Data Field. In the example depicted here, the Identifier denotes that this is an “Important message from the wheel speed sensor”. The data field declares the wheel speed to be 100 RPM.

This message is transmitted on the bus, and all other nodes will be able to see it. In fact, all nodes on the bus, including the one that sent the message, must receive the message and verify that the reception was error-free. They must then acknowledge reception of the message, whether or not a particular node was an intended recipient of the message.



If we look at the bits transmitted in a CAN message, we can see that the message consists of several fields.

The first bit sent is a Start of Frame bit, which indicates the beginning of a message transmission.


This is followed by an identifier, which helps each node on the network determine if the message is intended for it. includes the identifier bits and some control bits. The CAN specification defines two different formats for the identifier. The first format, called the Standard Identifier format, contains an 11 bit standard identifier or SID field and some control bits. The second format, which is the one shown here, is called the Extended Identifier format, and not only contains an 11 bit SID field and control bits, but also an 18 bit Extended Identifier or EID field.

This is followed by the actual data to be transmitted. This data field is preceded by a control field that specifies how many bytes of data are actually present in the message, which may be length from zero to eight bytes.

Following the data, a Cyclic Redundancy Check or CRC field is transmitted to ensure that the message is not corrupted on the bus. This is a key component of the Error Management mechanism of a CAN bus.


The CRC bits are followed by an Acknowledge field that allows all nodes on the bus to acknowledge reception of the message.

Finally, there is an End of Frame sequence that demarcates the end of the message and returns the bus to an idle state.


MICROCHIP
WebSeminars

CAN Protocol Overview

- The receivers will check the Identifier to see if they are interested in the message
- Checking the Identifier is done with message filters
- If a receiver's **filter** matches the identifier, it will store the Data Field of the message



© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module6


Once a node receives a message, the CAN module hardware determines if the message is of interest or not. The node then decides whether to process or discard data.

Remember that the identifier in the message indicates the content of the message. So the CAN module hardware inspects the identifier to see if it matches an identifier on its user-programmed list of acceptable identifiers.

These acceptable identifiers are called Message Filters. If the identifier in the message matches an identifier in any of the message filters, the node will accept the message into its memory buffer. If it does not match, the received message is discarded.

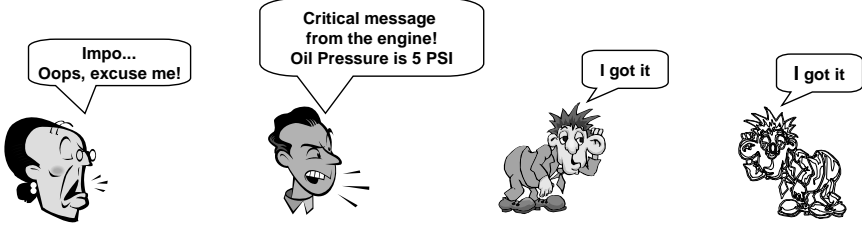
Moreover, the numerical value of the identifier inherently provides a measure of the priority of the message. This allows the receiver to prioritize received messages.

Point-to-point messaging is implemented by having only one node contain a matching filter. On the other hand, multicast messages are implemented by having all nodes contain a matching filter.


MICROCHIP
WebSeminars

Bus Arbitration

- Nodes must wait for a quiet bus before they begin talking
- What if two nodes try to transmit at the same time?
- The contents of the Identifier are used to **Arbitrate** who will talk




© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module7

On a CAN bus, not only can we have multiple receivers, but there may also be multiple transmitters. Essentially any node can send a message to any other node. So, how does the protocol ensure that different messages do not interfere with each other?

First, a node is not allowed to transmit until the bus is in an idle state. If a node is transmitting a message, all other nodes must wait for that node to finish before attempting transmission.

If multiple nodes try to transmit at the same time, the bus has a mechanism to arbitrate which message is more important. The nodes with the less important message will stop transmitting and the most important message will continue transmitting the message.


MICROCHIP
WebSeminars

Bus Arbitration

- Both nodes continue to transmit until mismatch
- A zero on the bus wins over a one on the bus
- Losing node stops transmitting, winner continues

	1		9		6						
Engine Control	0	0	1	1	0	0	1	0	1	1	0
	“Critical Message / Engine = 196”										
Wheel Speed	0	0	1	1	0	0	1	1	E		
	“Important Message / Wheel Speed = 19E”										

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 8

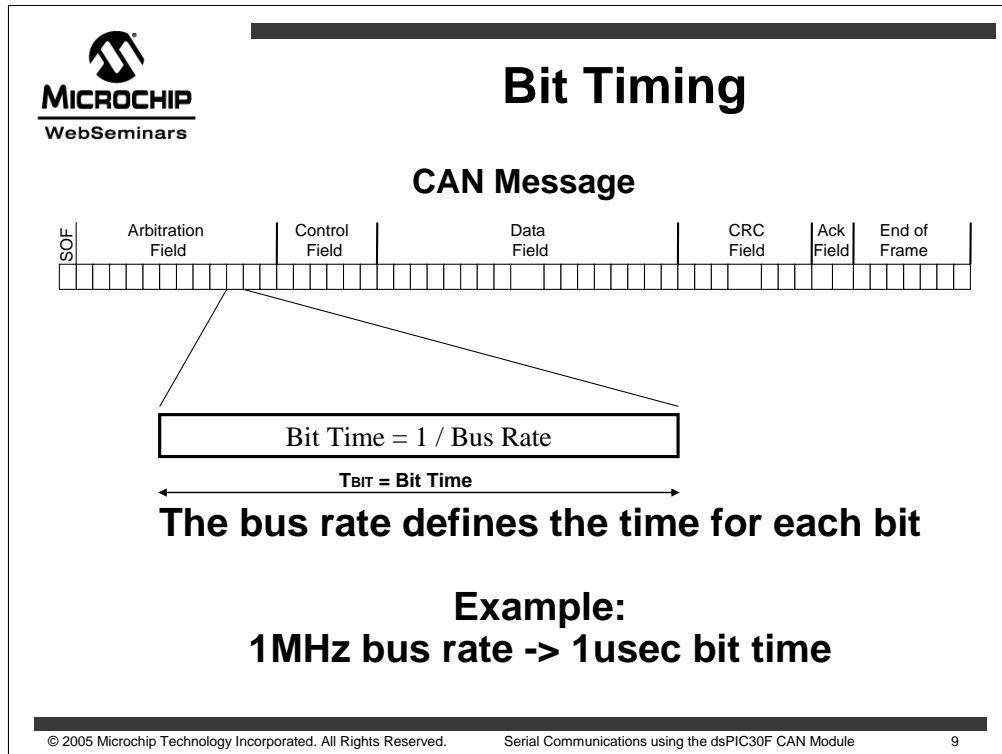
Arbitration is only required when more than 2 nodes attempt to transmit at the same time. As a node transmits each bit, it verifies that it sees the same bit value on the bus that it transmitted.

As long as different transmitters are sending the same information, none of them are aware that other nodes are also transmitting data.

In the CAN bus system, the bus is structured such that if one node is transmitting a one and another node is transmitting a zero, the data on the bus will be a zero.

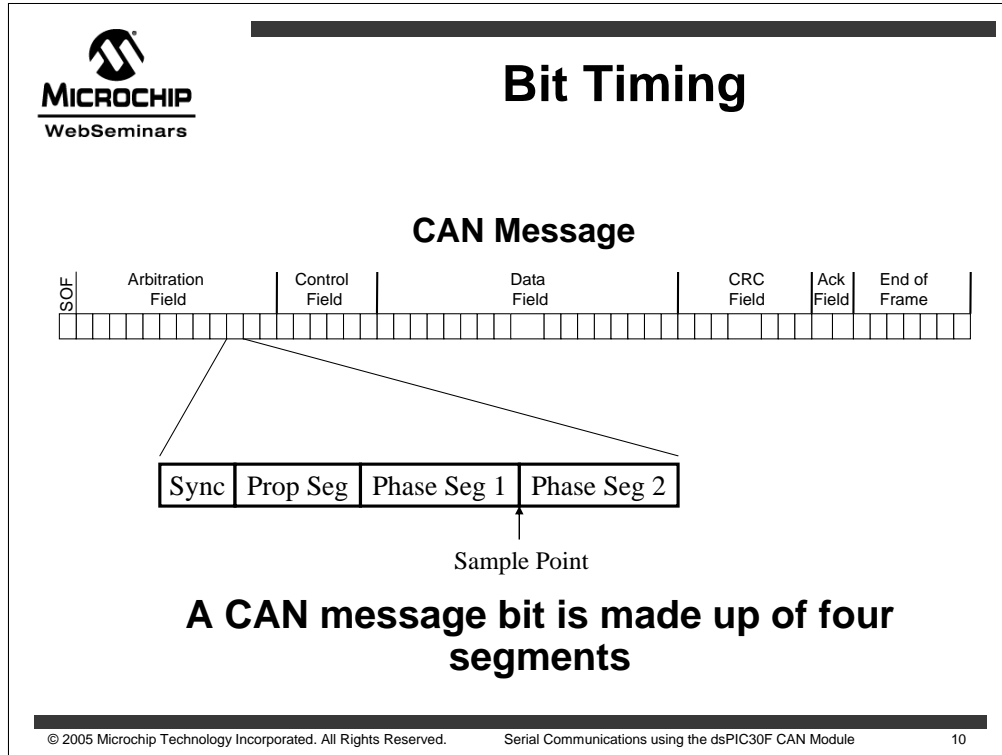
This enables a simple arbitration mechanism. When one node, the wheel speed in this case, transmits a one when the engine transmits a zero, the zero from the engine wins bus arbitration. The wheel speed node will detect that the data received on the bus did not match the data it transmitted, and will then cease to transmit its message.

Thus, it can be inferred that identifiers with a lower numerical value have higher priority, as they contain more zeros at the beginning of the message.




Let us now study the bit timing considerations of a CAN bus and see how to set the module up for a particular bus timing selection.

The CAN bus baud rate is defined as the time required to transmit one bit in the message. For a typical 1MHz baud rate, each bit requires 1 micro second of time. The bit time is abbreviated as T_{BIT} .



Within each bit time, the CAN protocol specifies four time segments: Synchronization Segment, Propagation Segment, Phase Segment 1 and Phase Segment 2.



Bit Timing

● Each Bit Timing Segment is made up of integer units of time called Time Quanta (TQ)

Sync	Prop Seg		Phase Seg 1			Phase Seg 2		
TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ
1TQ	1-8TQ		1-8TQ			1-8TQ		

$T_{BIT} = \text{Bit Time}$

● User configures each segment to a specific number of TQ

● Time allocated to each segment depends on CAN bus timing

● Bit Time can range from 8 to 25 TQ


© 2005 Microchip Technology Incorporated. All Rights Reserved.
Serial Communications using the dsPIC30F CAN Module
11

Each time segment is comprised of a certain number of smaller units of time called time quanta, abbreviated as TQ.

You may wonder, what is the need for all this complexity? Well, on the CAN bus, unlike some synchronous serial protocols such as SPI, there is no clock signal transmitted by a node. Since there is no specific clock signal, each node must use a digital phase locked loop to generate the clock needed by the module. This phase locked loop is part of the module hardware, and uses the time quantum as the basis for clock generation and synchronization.

Each time segment is allocated an integral number of time quanta. The total of all four segments can range from 8TQ to 25TQ, and is user-programmable.

To learn more about how to analyze the bus timing and determine how to allocate the TQ's to each of the time segments, see Microchip's web site for application note AN754: "Understanding Microchip's CAN Module Bit Timing".



MICROCHIP
WebSeminars

Bit Timing

- SJW<1:0> specifies 1-4 TQ for sync jump width
- PRSEG<2:0> specifies 1-8 TQ for propagation segment
- SEG1PH<2:0> specifies 1-8 TQ for phase segment 1
- SEG2PHTS specifies options on phase segment 2
 - SEG2PH<2:0> specifies 1-8 TQ for phase segment 2
- SAM specifies options on sampling the bus
- The TQ time is derived from the system clock
 - BRP<5:0> bits define TQ
 - $BRP = (TBIT/n * 2 * FCAN) - 1$, where $n=TQ$ clocks per bit
- CANCKS bit in C1CTRL register selects source of FCAN
 - $FCAN = FCY$, if CANCKS = 1
 - $FCAN = 4 * FCY$, if CANCKS = 0

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 12

Once the number of time quanta used in each time segment is known, they can be used to initialize the CAN timing configuration registers, C1CFG1 and C1CFG2.

For example:


The PRSEG bits specify the number of time quanta in the propagation segment.

The SEG1PH bits specify the number of time quanta in phase segment one.

The SEG2PH bits specify the number of time quanta in phase segment two.

The BRP bits in the C1CFG1 register, in conjunction with the values initialized in the C1CFG2 register, determine the CAN communication clock period, in other words: the bit rate.

The CANCKS bit, when cleared, provides a higher resolution in configuring the CAN bus timing, thereby allowing higher bit rates even with lower device operating speeds.


MICROCHIP
 WebSeminars

Initialization

- Set up CAN interrupt and enable the interrupt
- Set REQOP<2:0>=000 to change to operational mode
- When the OPMODE<2:0> bits reflect operational mode, the module is active
- Application can now run and use CAN module

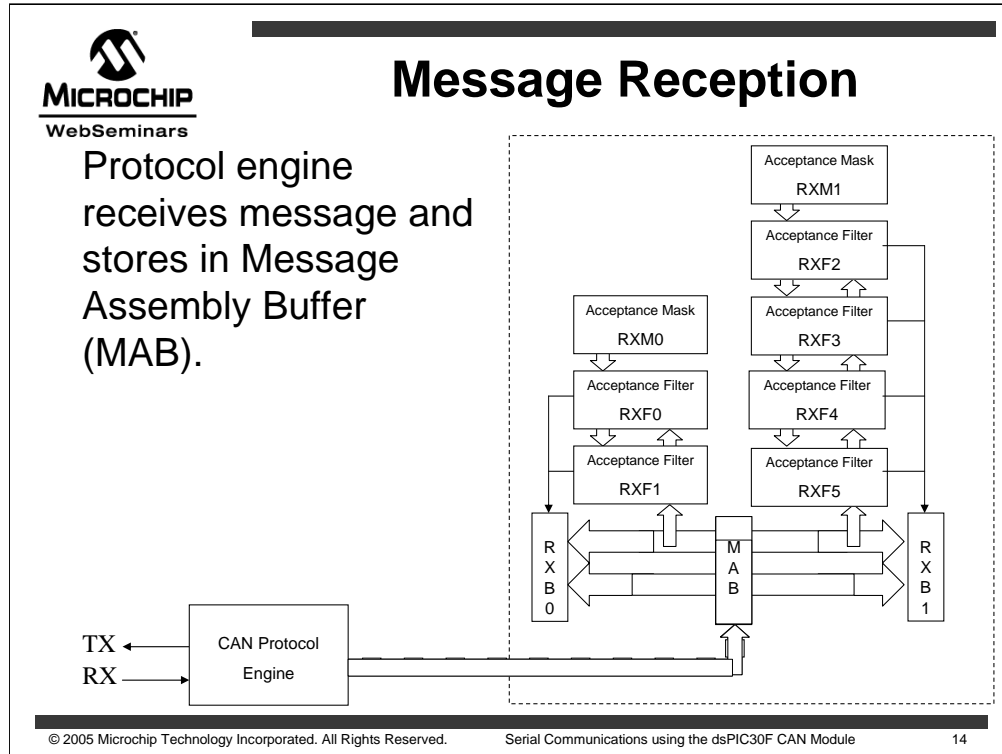
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
CANCAP	-	CANSIDL	ABAT	CANCKS	REQOP2	REQOP1	REQOP0
Bit 15	14	13	12	11	10	9	Bit 8
R-1	R-0	R-0	U-0	R-0	R-0	R-1	U-0
OPMODE2	OPMODE1	OPMODE0	-	ICOD2	ICOD1	ICOD0	-
Bit 7	6	5	4	3	2	1	Bit 0

C1CTRL REGISTER

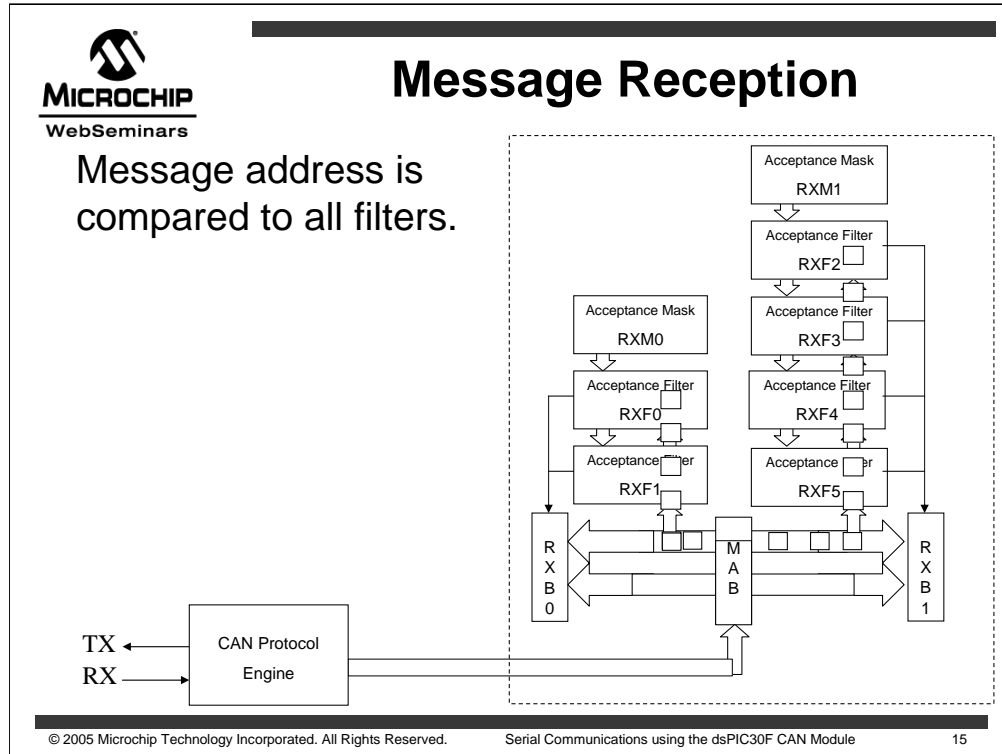
© 2005 Microchip Technology Incorporated. All Rights Reserved.
Serial Communications using the dsPIC30F CAN Module
13

On Reset, the CAN module is in the Configuration Mode, and therefore can not send and receive data on the bus until its mode is changed to the Normal Operation mode.

Before we can start using the CAN module, we need to enable the CAN module interrupts. Then, the module can be enabled by requesting the Normal Operation mode by clearing the Request Operating Mode control bits. The Operating Mode status bits automatically get cleared when the module enters its new mode, and then the module is ready for message transmission and reception.



Let us first understand how the CAN module receives a message on the bus. The CAN protocol engine, a key component of the CAN module hardware, does the actual bit reception and error checking. The protocol engine places the received bits of all messages into the message assembly buffer or MAB. When the message has been completely sent by the transmitting node, the complete message is already stored in the MAB of all receiving nodes.

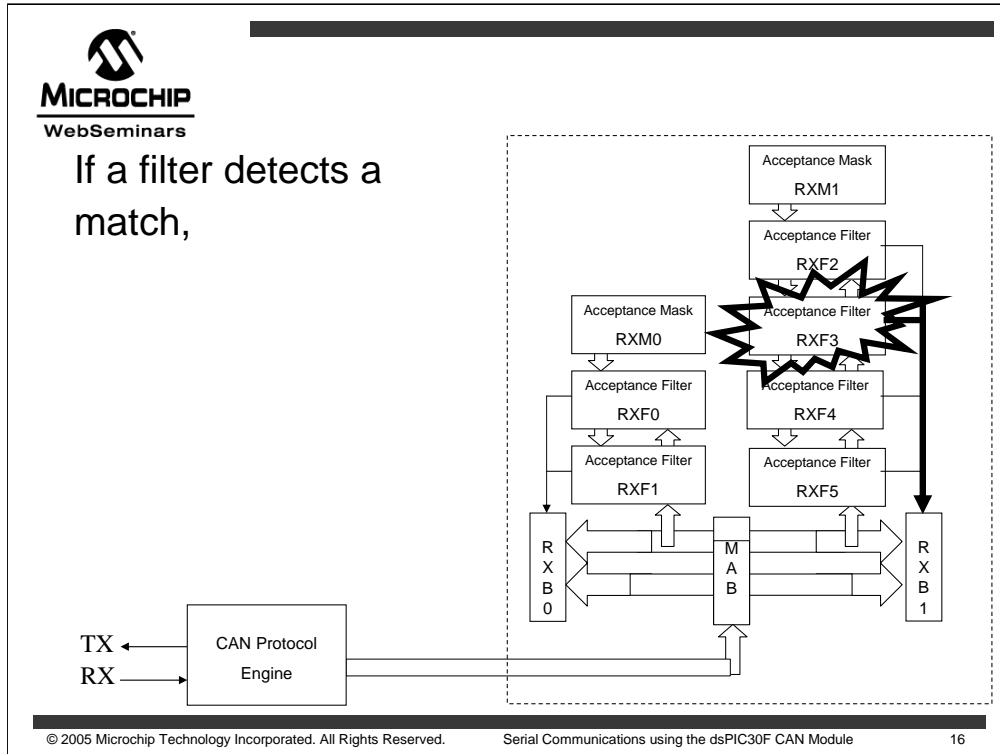


Once the message has been completely shifted into the MAB, the identifier portion of the message is sent to the filters for filter matching.

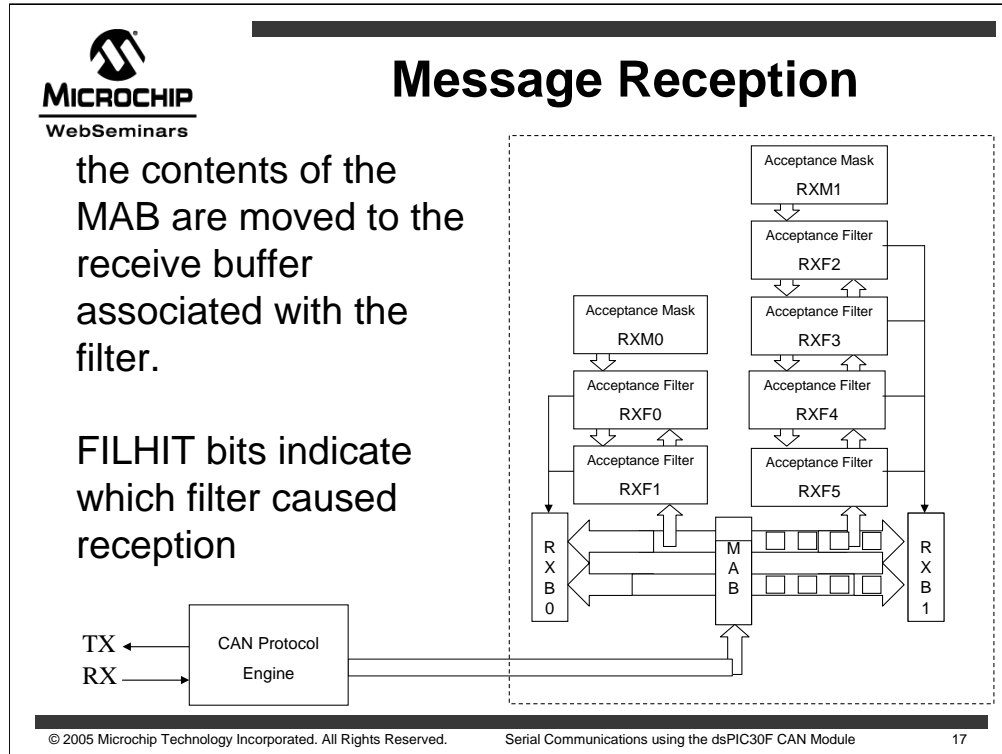
There are two receive buffers, denoted RXB0 and RXB1. The message in the MAB can go to either receive buffer.

There are six receive filters. Two of the filters, RXF0 and RXF1, are associated with buffer RXB0. The other four filters, RXF2 through RXF5, are associated with buffer RXB1. The identifier field from the received message is sent to all six filters.


There are two masks. Mask RXM0 is associated with buffer RXB0. Mask RXM1 is associated with buffer RXB1.



Suppose a filter value, in conjunction with the corresponding mask value, matches the identifier bits from the received message.



In this case, the module will accept that message into the receive buffer associated with that filter. For example, if the identifier bits specified by the user in RXF2 matched the identifier bits in the received message, the message is received in RXB1.



Receive Buffers

MICROCHIP

WebSeminars

File Name	Addr	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1RX1SID	0x80	-	-	-	SID<10:6>				SID<5:0>				SRR	RXIDE			
C1RX1EID	0x82	-	-	-	EID<17:14>				EID<13:6>								
C1RX1DLC	0x84			EID<5:0>				RXRTRRXRB1				-	-	-	RXRBO	DLC<3:0>	
C1RX1D1	0x86			Receive Buffer 1 Byte 1				Receive Buffer 1 Byte 0									
C1RX1D2	0x88			Receive Buffer 1 Byte 3				Receive Buffer 1 Byte 2									
C1RX1D3	0x8A			Receive Buffer 1 Byte 5				Receive Buffer 1 Byte 4									
C1RX1D4	0x8C			Receive Buffer 1 Byte 7				Receive Buffer 1 Byte 6									
C1RX1CON	0x8E	-	-	-	-	-	-	-	-	RX FUL	-	-	-	RX RTRRO	FILHIT<2:0>		

- A group of 8 SFR's make up a receive buffer
- Receive buffer contains identifier and data
- C1RX1CON is a control and status register for the buffer

© 2005 Microchip Technology Incorporated. All Rights Reserved.

Serial Communications using the dsPIC30F CAN Module


18

Each receive buffer consists of a block of 8 special function registers. The first 3 words of the buffer contain the identifier field of the received message. The next 4 words of the buffer contain the data field. The last word is a control and status register for the buffer.

When a message is accepted into the buffer, the module sets the Receive Buffer Full bit RXFUL, and generates an interrupt. The application software can then clear the RXFUL bit to indicate that it has completed reading the buffer and the buffer is now available to receive another message.

The RXRTRRO status bit indicates that the received message is a Remote Transfer Request, which is a type of message sent by another node to request data from this node.

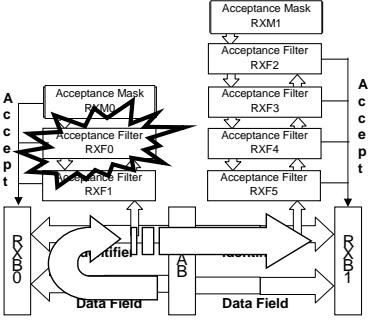
The least significant 3 bits in the C1RX1CON register are status bits which indicate which filter matched the received message identifier and caused the acceptance of the message. This helps the user software to determine the contents of the identifier.



MICROCHIP
WebSeminars

Receive Buffers

- DBEN=1 enables double buffering of the RXB0 receive buffer
- If RXB0 is full when it receives another message, the new message will be sent to RXB1 if it is available



=1 =1

R/C-0	U-0	U-0	U-0	R-0	R/W-0	R-0	R-0
RXFUL	-	-	-	RXRTRRO	DBEN	JTOFF	FILHITO
Bit 7	6	5	4	3	2	1	Bit 0

C1RX0CON REGISTER

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 19

Receive Buffer 0 provides the capability of optionally having a received message stored in buffer 1 in case buffer 0 is full when the message is received.

The DBEN bit is used to enable this feature. Let us assume that Receive Buffer 0 has already received a message, and that the application software has neither read the contents of the buffer nor cleared the RXFUL bit.

Suppose another message is now received on the bus, and the identifier matches the Filter 0 value. Normally, this would cause a receive buffer overflow error, and the incoming message would be lost.

However, if the DBEN bit is set, then instead of generating an overflow error, the contents of the Message Acceptance Buffer will be automatically transferred to Receive Buffer 1, assuming that Receive Buffer 1 is empty at that time.



Receive Masks and Filters

- The Mask determines which Filter bits are applied to the associated bit in the message
- The enabled Filter bits must match the bit in the message to accept the message

Mask Bit n	Filter Bit n	Message Identifier Bit n	Accept or Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Let us now look at the Receive Filters and Masks in a little more detail. Whenever a message is received on the CAN bus, the identifier in the received message is compared bit by bit with the filter and mask register values.

As mentioned before, a mask register is associated with a particular Receive Buffer, and therefore with multiple filters.

For each bit of the identifier, the corresponding Mask bit determines if that particular bit value is tested by the module. If the mask bit is zero, that bit will not be checked, meaning that essentially that bit is always accepted by the filter, even if its value did not match the corresponding bit in any of the Receive Filters assigned to that buffer. If the mask bit is one, the identifier bit will be checked and it must match the equivalent bit in the Filter register.

If every bit of the received identifier either matched the corresponding filter bit or was ignored, the message is automatically transferred to the appropriate Receive Buffer.



Receive Masks and Filters

- A group of 3 SFRs make up a receive mask
- Each CAN module has 2 sets of mask SFR's
- MIDE bit specifies if identifier type is checked

- A group of 3 SFRs make up a receive filter
- Each module has 6 sets of filter SFR's
- EXIDE bit specifies if standard or extended identifiers will be accepted

© 2005 Microchip Technology Incorporated. All Rights Reserved.

Serial Communications using the dsPIC30F CAN Module


21

The Receive Filter consists of 3 special function registers that are similar in structure to the Mask registers.

Each Filter contains a special control bit called Extended Identifier Enable, or EXIDE, which specifies whether that particular filter will accept standard or extended identifiers. This provides the user filter-level control over the acceptance of different message types.

Each Mask contains a bit called MIDE which specifies if the type of identifier will be checked by the module, that is, it determines whether the EXIDE bit has any effect. Basically, the MIDE bit provides the user buffer-level control over what type of messages are accepted by the filters assigned to that buffer.

If the MIDE bit is zero, then the EXIDE bit is effectively ignored, and both standard and extended identifiers may match the filter. If MIDE is one, then the EXIDE bit in each individual filter selects between standard and extended identifiers.


MICROCHIP
WebSeminars

Message Transmission

Transmit
Buffer
TXB0

Transmit
Buffer
TXB1

Transmit
Buffer
TXB2

Transmit
Byte
Sequencer

- **Messages are placed in one of three transmit message buffers and then sent to the bus by the protocol engine**


CAN Protocol
Engine

TX ←→ RX

© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module22

We have learnt about the reception of CAN messages. Now let us look at the process of transmitting CAN messages.

The CAN module has three transmit buffers. At any time, only one of the buffers is allowed to transmit data. Whenever the module initiates a transmission from a particular buffer, the bits of that buffer are sent to the CAN protocol engine for transmission on the bus and error checking.



Message Transmission

WebSeminars

- Determine the identifier field of the message
- TXIDE bit defines if identifier is standard (11-bit) or extended (29-bit)
- Load the identifier registers
- Move the message data into the buffer

File Name	Addr	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
C1TXOSID	0x62	SID<10:6>						-	-	-	SID<5:0>						SRR	TXIDE	
C1TXOEID	0x62	EID<17:14>						-	-	-	EID<13:6>								
C1TXODLC	0x64	EID<5:0>						TXRTR TXRB1 TXRBO			DLC<3:0>						-	-	-
C1TXOD1	0x66	Transmit Buffer 0 Byte 1						Transmit Buffer 0 Byte 0											
C1TXOD2	0x68	Transmit Buffer 0 Byte 3						Transmit Buffer 0 Byte 2											
C1TXOD3	0x6A	Transmit Buffer 0 Byte 5						Transmit Buffer 0 Byte 4											
C1TXOD4	0x6C	Transmit Buffer 0 Byte 7						Transmit Buffer 0 Byte 6											
C1TXOCON	0x6E	-	-	-	-	-	-	-	-	-	-	TX ABAT	TX LARB	TX ERR	TX REQ	-	TXPRI<1:0>		

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 23

Generally, the user application determines when a message needs to be sent.

For example, if an engine node in a car determines that the oil pressure is too low, it may send a message notifying this fact to other car modules. The application will assemble the identifier information and move the identifier into the first 3 words of the transmit buffer.

The TXIDE bit in the transmit buffer control register specifies if the identifier of the transmitted message is a standard or extended identifier.

In the example of an engine node, the data part of the message may contain the actual oil pressure detected. The application can move up to 8 bytes of data into the data field of the transmit buffer. The number of data bytes in the message is coded into the Data Length Code, or DLC, field of the identifier.

A transmitter can also send a special message type called a remote transfer request or RTR. This is a message asking another node to send data back using another Data Frame. A message can be transmitted with the TXRTR bit set but containing no data bytes.



Message Transmission

- Setting TXREQ bit (C1TX0CON register) requests a transmission
- Buffer used by module after TXREQ set
- If the protocol engine is busy sending/receiving another message, this message is queued
- Module clears TXREQ bit when the transmission completes
- Module generates a CAN interrupt when transmission completes

© 2005 Microchip Technology Incorporated. All Rights Reserved.


Serial Communications using the dsPIC30F CAN Module

24

Once the message is assembled in the transmit buffer, the application can request transmission of the message by setting the Transmit Request, or TXREQ, bit. Note that this is only a request to send the message and may not trigger an immediate data transmission, because the bus may be busy with another message at the time the TXREQ bit is set. The module will wait until the bus is idle before attempting transmission.

Once the TXREQ is set, the application should not attempt to access the transmit buffer because it may be in use by the module.

When the module completes transmitting the message, it automatically clears the TXREQ bit and generates a CAN interrupt. At this time, the application software may choose to initiate another transmission.



Message Transmission


- When the protocol engine has a bus opening, it will chose which of the queued transmit buffers to send
- The TXPRI bits (C1TX0CON register) select the priority of the three transmit buffers
 - 11=highest (first); 00=lowest (last)
 - If buffers have same TXPRI values, the higher number buffer goes next

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 25

Since there are three transmit buffers, it is possible for two transmit buffers, or even all three, to have their Transmit Request bits set. The transmission requests are then automatically queued by the module.

When the bus becomes free, the module needs to decide the sequence in which the messages would be transmitted. This is done on the basis of a user-specified priority level for each buffer. The Transmit Priority bits in the transmit buffer control register can be set up for one of 4 priority levels. A message with a Transmit Priority of '11' has the highest transmission priority, whereas a message with Transmit Priority of '00' has the lowest transmission priority. This feature provides an additional degree of flexibility to the user application in prioritizing messages, above and beyond the bus arbitration methodology specified by the CAN protocol.

If we have two enqueued buffers that have the same Transmit Priority, the buffer with the higher index will be transmitted first. For example, if TXB2 and TXB0 both have high priorities, TXB2 will be transmitted first.



CAN Interrupts

MICROCHIP
WebSeminars

- The module generates one interrupt
- Interrupt can originate from one of 8 sources
- Each source has a flag bit in the C1INTF register and an enable bit in the C1INTE register

R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
RXOOVR	RX1OVR	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
Bit 15	14	13	12	11	10	9	Bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IVRIF	WAKIF	ERRIF	TX2IF	TX1IF	TXOIF	RX1IF	RXOIF
Bit 7	6	5	4	3	2	1	Bit 0

C1INTF REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
-	-	-	-	-	-	-	-
Bit 15	14	13	12	11	10	9	Bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IVRIE	WAKIE	ERRIE	TX2IE	TX1IE	TXOIE	RX1IE	RXOIE
Bit 7	6	5	4	3	2	1	Bit 0


C1INTE REGISTER

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 26

Each CAN module generates one interrupt and has one interrupt vector. However, this interrupt can represent many different sources of interrupt events occurring in the CAN module.

There are 8 primary events that can generate a CAN interrupt. Each of these events has its own interrupt enable bit and an interrupt flag bit. A CAN interrupt is generated if any of the individual CAN event interrupt enable bits are set. A certain CAN event can cause an interrupt only if the corresponding event's interrupt enable bit is set.

The CAN interrupt flag is set if any of the individual event flags are set. A certain CAN event flag is set even if the corresponding event interrupt enable bit is not set. Thus, the individual CAN event interrupt enable and flag bits behave just like the other interrupt enable and flag bits in dsPIC30F devices.



CAN Interrupts

- 1 interrupt for each receive buffer - RX0IF, RX1IF
- 1 interrupt for each transmit buffer - TX0IF, TX1IF, TX2IF
- 1 interrupt for wake up - WAKIF
- 1 interrupt for invalid received message - IVRIF
- 1 interrupt that groups 8 error interrupts together - ERRIF

© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 27

Let us take a look at the 8 primary sources of CAN interrupts.


Each of the two receive buffers can generate a CAN interrupt when a message is received in the buffer.

Similarly, each of the three transmit buffers can cause an interrupt when a message queued for transmission has successfully completed transmission.

The wake up interrupt occurs when the processor is in sleep or disable mode and the CAN module detects CAN bus activity.

If the receiver receives an invalid message, the receiver generates an invalid message interrupt.

Finally, there is an error interrupt. This interrupt occurs if any one of eight possible error events occurs, which are discussed in the next slide. The Error Interrupt bit is effectively a logical-OR of 8 distinct error conditions, each having its own status flag. The individual error flags help the user application determine the sources of system error, potentially enhancing system robustness.



CAN Interrupts

- Eight error conditions can generate an interrupt
 - Receiver Buffer 0 Overflowed - RX0OVR
 - Receiver Buffer 1 Overflowed - RX1OVR
 - Transmitter has bus error warning - TXWAR
 - Transmitter has gone error passive - TXBP
 - Transmitter has gone bus off - TXBO
 - Receiver has bus error warning - RXWAR
 - Receiver has gone error passive - RXBP
 - Transmitter or receiver bus error warning - EWARN


© 2005 Microchip Technology Incorporated. All Rights Reserved. Serial Communications using the dsPIC30F CAN Module 28

The 8 possible error events are encoded into the most significant byte of the C1INTF interrupt flag register. The error event bits are set when the error occurs and when the bit is set, it causes an error interrupt event to occur and the Error Interrupt flag is set. The error event bits are cleared only by resolving the error condition.

The error events also include receiver buffer overflows on either receive buffer one or zero.

The CAN module maintains a count of both transmission and reception errors. When either the receiver or transmitter error counters reaches the warning count greater than 96, the error warning bits are set. If the counters reach more than 128, the error passive bits are set. If the transmit counter reaches more than 255, the bus off bit is set. The EWARN bit is a combination of the RXWARN and TXWARN bits.

Please refer to the dsPIC30F Family Reference Manual for more details about the error management mechanisms built into the CAN module.



Key Support Documents


Device Selection Reference	Document #
● General Purpose and Sensor Family Data Sheet	DS70083
● Motor Control and Power Conv. Data Sheet	DS70082
● dsPIC30F Family Overview	DS70043

Base Design Reference	Document #
● dsPIC30F Family Reference Manual	DS70046
● dsPIC30F Programmer's Reference Manual	DS70030
● MPLAB® C30 C Compiler User's Guide	DS51284
● MPLAB ASM30, LINK30 & Utilities User's Guide	DS51317
● dsPIC® Language Tools Libraries User's Guide	DS51456

© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module 29

For more information, here are references to some important documents that contain a wealth of information about the dsPIC30F family of devices.

The Family Reference Manual contains detailed information about the architecture and peripherals, whereas the Programmer's Reference Manual contains a thorough description of the instruction set.



MICROCHIP
WebSeminars

Key Support Documents

<u>Device Specific Reference</u>	<u>Document #</u>
● dsPIC30F3014/4013 Data Sheet	DS70138
● dsPIC30F4011/4012 Data Sheet	DS70135
● dsPIC30F5011/5013 Data Sheet	DS70116
● dsPIC30F6010 Data Sheet	DS70119
● dsPIC30F6011/6012/6013/6014 Data Sheet	DS70117

Microchip Web Site: www.microchip.com

© 2005 Microchip Technology Incorporated. All Rights Reserved.Serial Communications using the dsPIC30F CAN Module30

For device-specific information such as pinout diagrams, packaging and electrical characteristics, the device datasheets listed here are the best source of information.

All these documents can be obtained from the Microchip web site shown, by clicking on the “dsPIC® Digital Signal Controllers” or “Technical Documentation” link.