Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

MCP25625

CAN Controller with Integrated Transceiver

General Features

- Stand-Alone CAN 2.0B Controller with Integrated CAN Transceiver and Serial Peripheral Interface (SPI)
- Up to 1 Mb/s Operation
- Very Low Standby Current (10 µA, typical)
- Up to 10 MHz SPI Clock Speed
- Interfaces Directly with Microcontrollers with 2.7V to 5.5V I/Os
- Available in SSOP-28L and 6x6 QFN-28L
- Temperature Ranges:
  - Extended (E): -40°C to +125°C

CAN Controller Features

- V_{DD}: 2.7 to 5.5V
- Implements CAN 2.0B (ISO11898-1)
- Three Transmit Buffers with Prioritization and Abort Features
- Two Receive Buffers
- Six Filters and Two Masks with Optional Filtering on the First Two Data Bytes
- Supports SPI Modes 0,0 and 1,1
- Specific SPI Commands to Reduce SPI Overhead
- Buffer Full and Request-to-Send Pins are Configurable as General Purpose I/Os
- One Interrupt Output Pin

CAN Transceiver Features

- V_{DDA}: 4.5V to 5.5V
- Implements ISO-11898-2 and ISO-11898-5 Standard Physical Layer Requirements
- CAN Bus Pins are Disconnected when Device is Unpowered:
  - An unpowered node or brown-out event will not load the CAN bus
- Detection of Ground Fault:
  - Permanent Dominant detection on T_{XD}
  - Permanent Dominant detection on bus
- Power-on Reset and Voltage Brown-Out Protection on V_{DDA} Pin
- Protection Against Damage Due to Short-Circuit Conditions (Positive or Negative Battery Voltage)
- Protection Against High-Voltage Transients in Automotive Environments
- Automatic Thermal Shutdown Protection
- Suitable for 12V and 24V Systems
- Meets or Exceeds Stringent Automotive Design Requirements, Including *"Hardware Requirements for LIN, CAN and FlexRay Interfaces in Automotive Applications"*, Version 1.3, May 2012
- High Noise Immunity Due to Differential Bus Implementation
- High-ESD Protection on CANH and CANL, Meets IEC61000-4-2 up to ±8 kV

Description

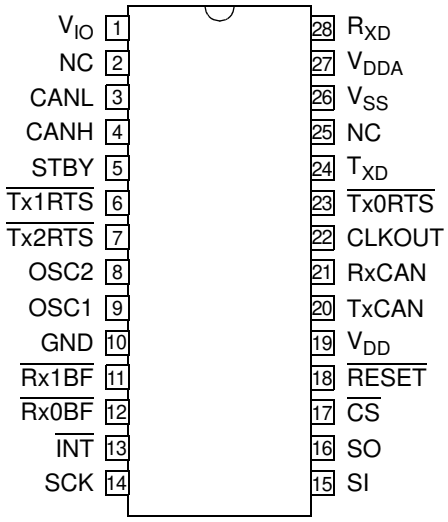
The MCP25625 is a complete, cost-effective and small footprint CAN solution that can be easily added to a microcontroller with an available SPI interface.

The MCP25625 interfaces directly with microcontrollers operating at 2.7V to 5.5V; there are no external level shifters required. In addition, the MCP25625 connects directly to the physical CAN bus, supporting all requirements for CAN high-speed transceivers.

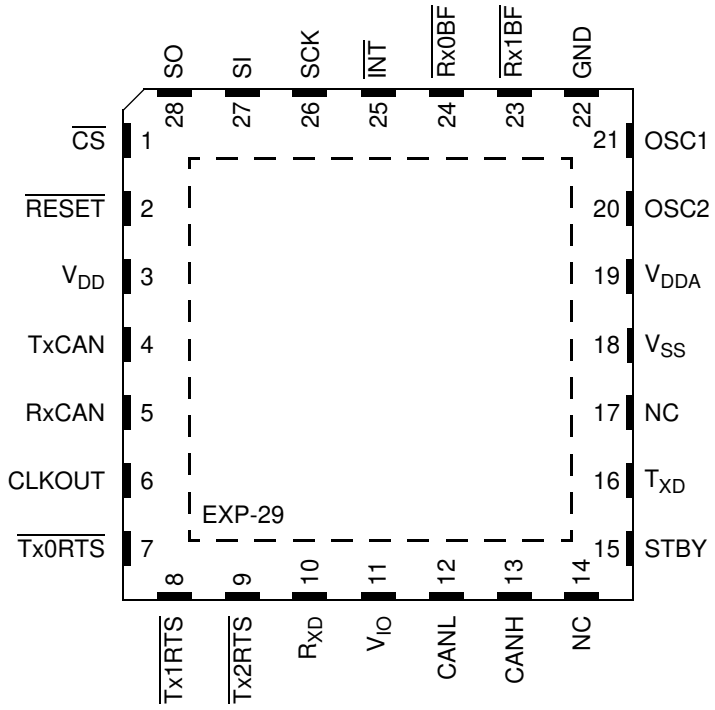
The MCP25625 meets the automotive requirements for high-speed (up to 1 Mb/s), low quiescent current, Electromagnetic Compatibility (EMC) and Electrostatic Discharge (ESD).

Package Types

MCP25625
SSOP



MCP25625
6x6 QFN*



* Includes Exposed Thermal Pad (EP); see Table 1-1.

1.0 DEVICE OVERVIEW

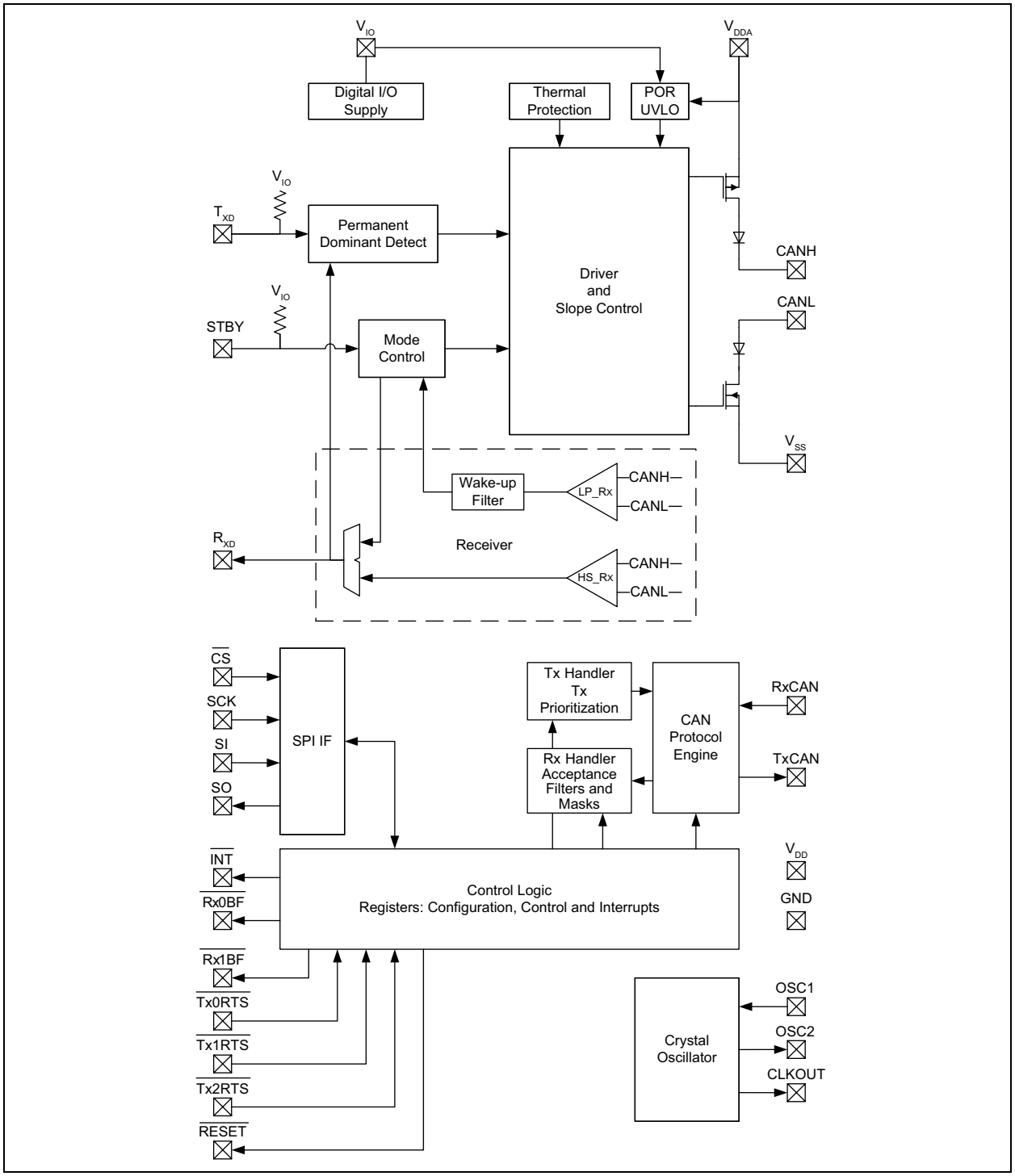
A typical CAN solution consists of a CAN controller that implements the CAN protocol, and a CAN transceiver that serves as the interface to the physical CAN bus. The MCP25625 integrates both the CAN controller and the CAN transceiver. Therefore, it is a complete CAN solution that can be easily added to a microcontroller with an SPI interface.

1.1 Block Diagram

Figure 1-1 shows the block diagram of the MCP25625. The CAN transceiver is illustrated in the top half of the block diagram, see **Section 6.0 "CAN Transceiver"** for more details.

The CAN controller is depicted at the bottom half of the block diagram, and described in more detail in **Section 3.0 "CAN Controller"**.

FIGURE 1-1: MCP25625 BLOCK DIAGRAM



1.2 Pin Out Description

The descriptions of the pins are listed in Table 1-1.

TABLE 1-1: MCP25625 PIN DESCRIPTION

Pin Name	6x6 QFN	SSOP	Block(1)	Pin Type	Description
V_{IO}	11	1	CAN Transceiver	P	Digital I/O Supply Pin for CAN Transceiver
NC	14	2	—	—	No Connection
CANL	12	3	CAN Transceiver	HV I/O	CAN Low-Level Voltage I/O
CANH	13	4	CAN Transceiver	HV I/O	CAN High-Level Voltage I/O
STBY	15	5	CAN Transceiver	I	Standby Mode Input
$\overline{Tx1RTS}$	8	6	CAN Controller	I	TXB1 Request-to-Send
$\overline{Tx2RTS}$	9	7	CAN Controller	I	TXB2 Request-to-Send
OSC2	20	8	CAN Controller	O	External Oscillator Output
OSC1	21	9	CAN Controller	I	External Oscillator Input
GND	22	10	CAN Controller	P	Ground
$\overline{Rx1BF}$	23	11	CAN Controller	O	RxB1 Interrupt
$\overline{Rx0BF}$	24	12	CAN Controller	O	RxB0 Interrupt
$\overline{INT}$	25	13	CAN Controller	O	Interrupt Output
SCK	26	14	CAN Controller	I	SPI Clock Input
SI	27	15	CAN Controller	I	SPI Data Input
SO	28	16	CAN Controller	O	SPI Data Output
$\overline{CS}$	1	17	CAN Controller	I	SPI Chip Select Input
$\overline{RESET}$	2	18	CAN Controller	I	Reset Input
V_{DD}	3	19	CAN Controller	P	Power for CAN Controller
TxCAN	4	20	CAN Controller	O	Transmit Output to CAN Transceiver
RxCAN	5	21	CAN Controller	I	Receive Input from CAN Transceiver
CLKOUT	6	22	CAN Controller	O	Clock Output/SOF
$\overline{Tx0RTS}$	7	23	CAN Controller	I	TXB0 Request-to-Send
T_{XD}	16	24	CAN Transceiver	I	Transmit Data Input from CAN Controller
NC	17	25	—	—	No Connection
V_{SS}	18	26	CAN Transceiver	P	Ground
V_{DDA}	19	27	CAN Transceiver	P	Power for CAN Transceiver
R_{XD}	10	28	CAN Transceiver	O	Receive Data Output to CAN Controller
EP	29	—	—	—	Exposed Thermal Pad

Legend: P = Power, I = Input, O = Output, HV = High Voltage.

Note 1: See **Section 3.0 "CAN Controller"** and **Section 6.0 "CAN Transceiver"** for further information.

1.3 Typical Application

Figure 1-2 shows an example of a typical application of the MCP25625. In this example, the microcontroller operates at 3.3V.

V_{DDA} supplies the CAN transceiver and must be connected to 5V.

V_{DD}, V_{IO} of the MCP25625 are connected to the V_{DD} of the microcontroller. The digital supply can range from 2.7V to 5.5V. Therefore, the I/O of the MCP25625 is connected directly to the microcontroller, no level shifters are required.

The T_{XD} and R_{XD} pins of the CAN transceiver must be externally connected to the TxCAN and RxCAN pins of the CAN controller.

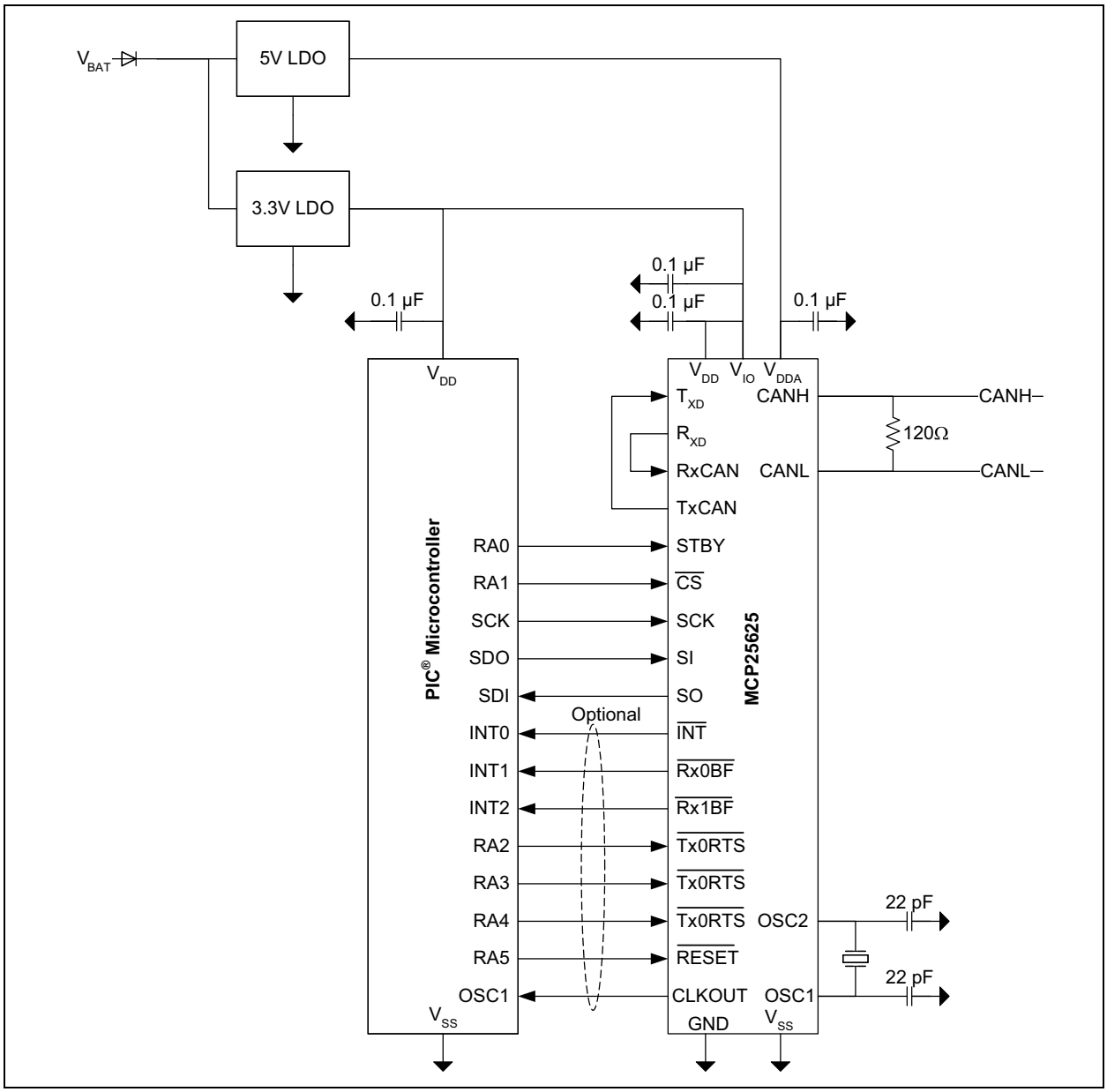
The SPI interface is used to configure and control the CAN controller.

The $\overline{INT}$ pin of the MCP25625 signals an interrupt to the microcontroller. Interrupts need to be cleared by the microcontroller through SPI.

The usage of $\overline{RxnBF}$ and $\overline{TxnRTS}$ is optional, since the functions of these pins can be accessed through SPI. The $\overline{RESET}$ pin can optionally be pulled up to the V_{DD} of the MCP25625 using a 10 kΩ resistor.

The CLKOUT pin provides the clock to the microcontroller.

FIGURE 1-2: MCP25625 INTERFACING WITH A 3.3V MICROCONTROLLER



NOTES:

2.0 MODES OF OPERATION

2.1 CAN Controller Modes of Operation

The CAN controller has five modes of operation:

- Configuration mode
- Normal mode
- Sleep mode
- Listen-Only mode
- Loopback mode

The operational mode is selected via the REQOP<2:0> bits in the CANCTRL register (see Register 4-34).

When changing modes, the mode will not actually change until all pending message transmissions are complete. The requested mode must be verified by reading the OPMOD<2:0> bits in the CANSTAT register (see Register 4-35).

2.2 CAN Transceiver Modes of Operation

The CAN transceiver has two modes of operation:

- Normal mode
- Standby mode

Normal mode is selected by applying a low level to the STBY pin. The driver block is operational and can drive the bus pins. The slopes of the output signals on CANH and CANL are optimized to produce minimal Electromagnetic Emissions (EME). The high-speed differential receiver is active.

Standby mode is selected by applying a high level to the STBY pin. In Standby mode, the transmitter and the high-speed part of the receiver are switched off to minimize power consumption. The low-power receiver and the wake-up filter are enabled in order to monitor the bus for activity. The receive pin (R_{XD}) will show a delayed representation of the CAN bus, due to the wake-up filter.

2.3 Configuration Mode

The MCP25625 must be initialized before activation. This is only possible if the device is in Configuration mode. Configuration mode is automatically selected after power-up, a Reset or can be entered from any other mode by setting the REQOPx bits in the CANCTRL register. When Configuration mode is entered, all error counters are cleared. Configuration mode is the only mode where the following registers are modifiable:

- CNF1, CNF2, CNF3
- TXRTSCTRL
- Acceptance Filter registers

2.4 Normal Mode

Normal mode is the standard operating mode of the MCP25625. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the MCP25625 transmits messages over the CAN bus.

Both the CAN controller and the CAN transceiver must be in Normal mode.

2.5 Sleep/Standby Mode

The CAN controller has an internal Sleep mode that is used to minimize the current consumption of the device. The SPI interface remains active for reading even when the MCP25625 is in Sleep mode, allowing access to all registers.

Sleep mode is selected via the REQOPx bits in the CANCTRL register. The OPMODx bits in the CANSTAT register indicate the operation mode. These bits should be read after sending the `SLEEP` command to the MCP25625. The MCP25625 is active and has not yet entered Sleep mode until these bits indicate that Sleep mode has been entered.

When in Sleep mode, the MCP25625 stops its internal oscillator. The MCP25625 will wake-up when bus activity occurs or when the microcontroller sets via the SPI interface. The WAKIF bit in the CANINTF register will “generate” a wake-up attempt (the WAKIE bit in the CANINTE register must also be set in order for the wake-up interrupt to occur).

The CAN transceiver must be in Standby mode in order to take advantage of the low standby current of the transceiver. After a wake-up, the microcontroller must put the transceiver back into Normal mode using the STBY pin.

2.5.1 WAKE-UP FUNCTIONS

The CAN transceiver will monitor the CAN bus for activity. The wake-up filter inside the transceiver is enabled to avoid a wake-up due to noise. In case there is activity on the CAN bus, the R_{XD} pin will go low. The CAN bus wake-up function requires both CAN transceiver supply voltages to be in a valid range: V_{DDA} and V_{IO}.

The CAN controller will detect a falling edge on the RxCAN pin and interrupt the microcontroller if the wake-up interrupt is enabled.

Since the internal oscillator is shut down while in Sleep mode, it will take some amount of time for the oscillator to start-up and the device to enable itself to receive messages. This Oscillator Start-up Timer (OST) is defined as 128 T_{OSC}.

The device will ignore the message that caused the wake-up from Sleep mode, as well as any messages that occur while the device is “waking up”. The device will wake-up in Listen-Only mode.

The microcontroller must set both the CAN controller and CAN transceiver to Normal mode before the MCP25625 will be able to communicate on the bus.

2.6 Listen-Only Mode

Listen-Only mode provides a means for the MCP25625 to receive all messages (including messages with errors) by configuring the RXM<1:0> bits in the RXBxCTRL register. This mode can be used for bus monitor applications or for detecting the baud rate in “hot plugging” situations.

For Auto-Baud Detection (ABD), it is necessary that at least two other nodes are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received.

Listen-Only mode is a silent mode, meaning no messages will be transmitted while in this mode (including error flags or Acknowledge signals). In Listen-Only mode, both valid and invalid messages will be received regardless of filters and masks or RXM<1:0> bits in the RXBxCTRL registers. The error counters are reset and deactivated in this state. The Listen-Only mode is activated by setting the REQOPx bits in the CANCTRL register.

2.7 Loopback Mode

Loopback mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing.

In this mode, the ACK bit is ignored and the device will allow incoming messages from itself, just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state (including error flags or Acknowledge signals). The TxCAN pin will be in a Recessive state.

The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the REQOPx bits in the CANCTRL register.

3.0 CAN CONTROLLER

The CAN controller implements the CAN protocol Version 2.0B. It is compatible with the ISO 11898-1 standard.

Figure 3-1 illustrates the block diagram of the CAN controller. The CAN controller consists of the following major blocks:

- CAN protocol engine
- TX handler
- RX handler
- SPI interface
- Control logic with registers and interrupt logic
- I/O pins
- Crystal oscillator

3.1 CAN Module

The CAN protocol engine, together with the TX and RX handlers, provide all the functions required to receive and transmit messages on the CAN bus. Messages are transmitted by first loading the appropriate message buffers and control registers. Transmission is initiated by using control register bits via the SPI interface or by using the transmit enable pins. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against the user-defined filters to see if it should be moved into one of the two receive buffers.

3.2 Control Logic

The control logic block controls the setup and operation of the MCP25625 and contains the registers.

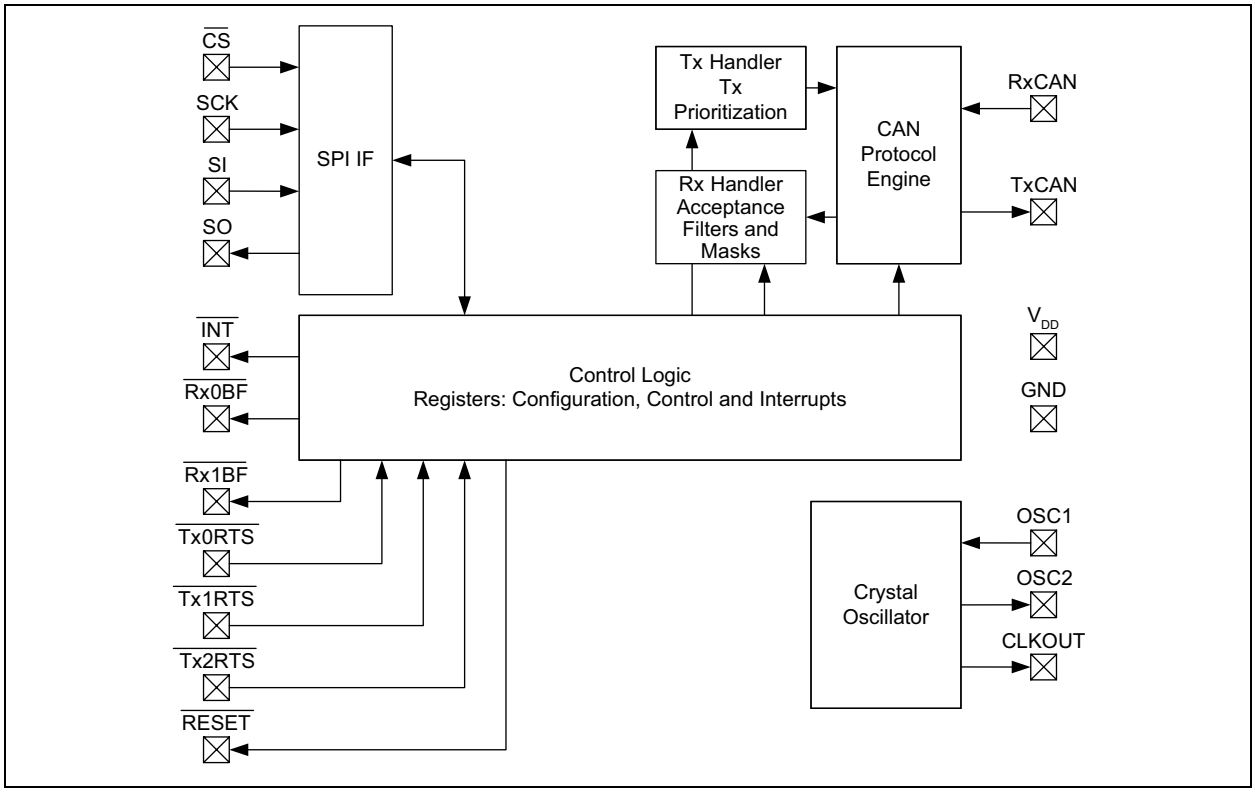
Interrupt pins are provided to allow greater system flexibility. There is one multipurpose interrupt pin (as well as specific interrupt pins) for each of the receive registers that can be used to indicate a valid message has been received and loaded into one of the receive buffers. Use of the specific interrupt pins is optional. The general purpose interrupt pin, as well as Status registers (accessed via the SPI interface), can also be used to determine when a valid message has been received.

Additionally, there are three pins available to initiate immediate transmission of a message that has been loaded into one of the three transmit registers. Use of these pins is optional, as initiating message transmissions can also be accomplished by utilizing control registers accessed via the SPI interface.

3.3 SPI Protocol Block

The microcontroller interfaces to the device via the SPI interface. Registers can be accessed using the SPI `READ` and `WRITE` commands. Specialized SPI commands reduce the SPI overhead.

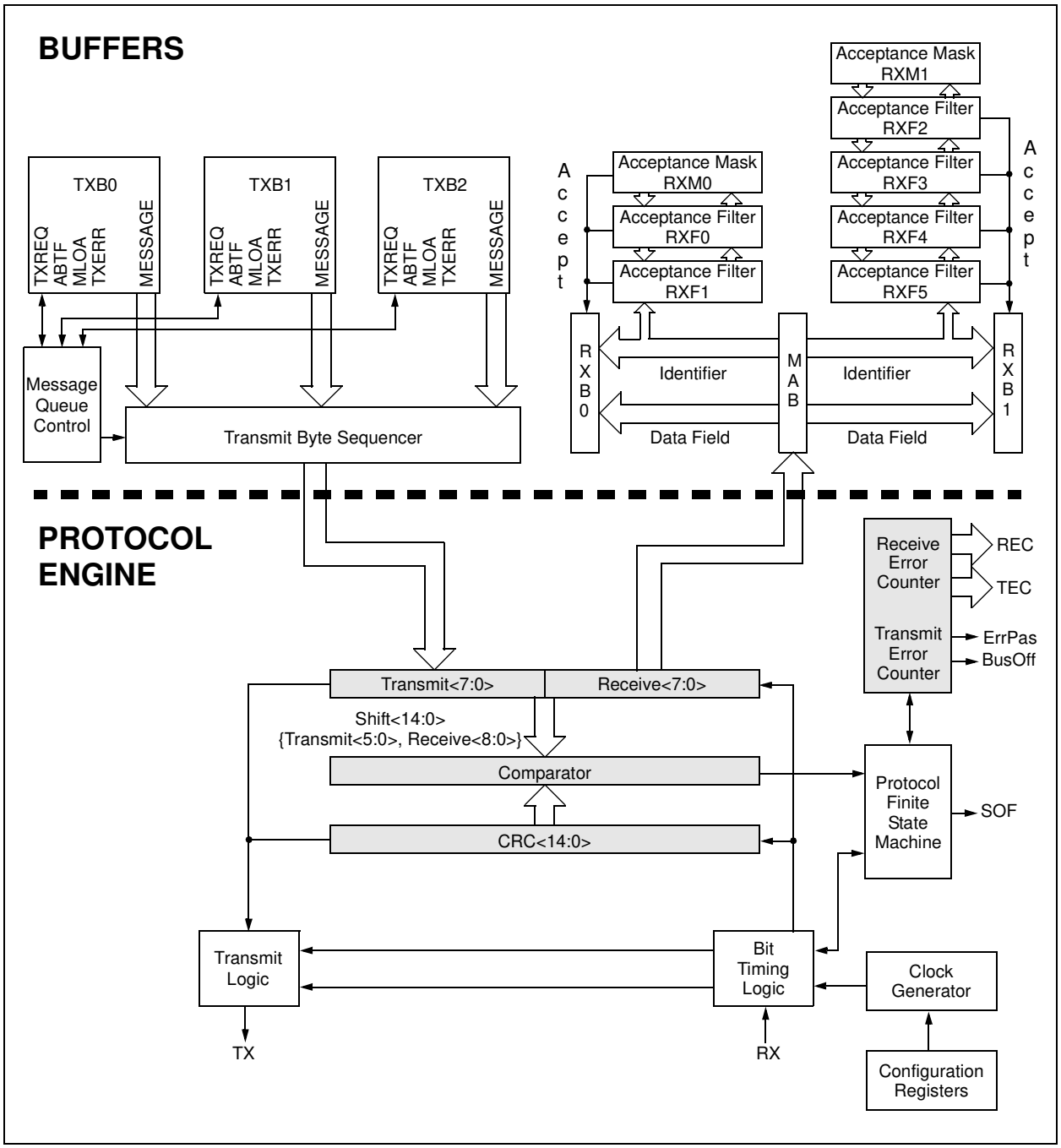
FIGURE 3-1: CAN CONTROLLER BLOCK DIAGRAM



3.4 CAN Buffers and Filters

Figure 3-2 shows the CAN buffers and filters in more detail. The MCP25625 has three transmit and two receive buffers, two acceptance masks (one per receive buffer) and a total of six acceptance filters.

FIGURE 3-2: CAN BUFFERS AND PROTOCOL ENGINE



3.5 CAN Protocol Engine

The CAN protocol engine combines several functional blocks, shown in Figure 3-3 and described below.

3.5.1 PROTOCOL FINITE STATE MACHINE

The heart of the engine is the Finite State Machine (FSM). The FSM is a sequencer that controls the sequential data stream between the TX/RX Shift register, the CRC register and the bus line. The FSM also controls the Error Management Logic (EML) and the parallel data stream between the TX/RX Shift registers and the buffers. The FSM ensures that the processes of reception, arbitration, transmission and error signaling are performed according to the CAN protocol. The automatic retransmission of messages on the bus line is also handled by the FSM.

3.5.2 CYCLIC REDUNDANCY CHECK

The Cyclic Redundancy Check register generates the Cyclic Redundancy Check (CRC) code, which is transmitted after either the control field (for messages with 0 data bytes) or the data field and is used to check the CRC field of incoming messages.

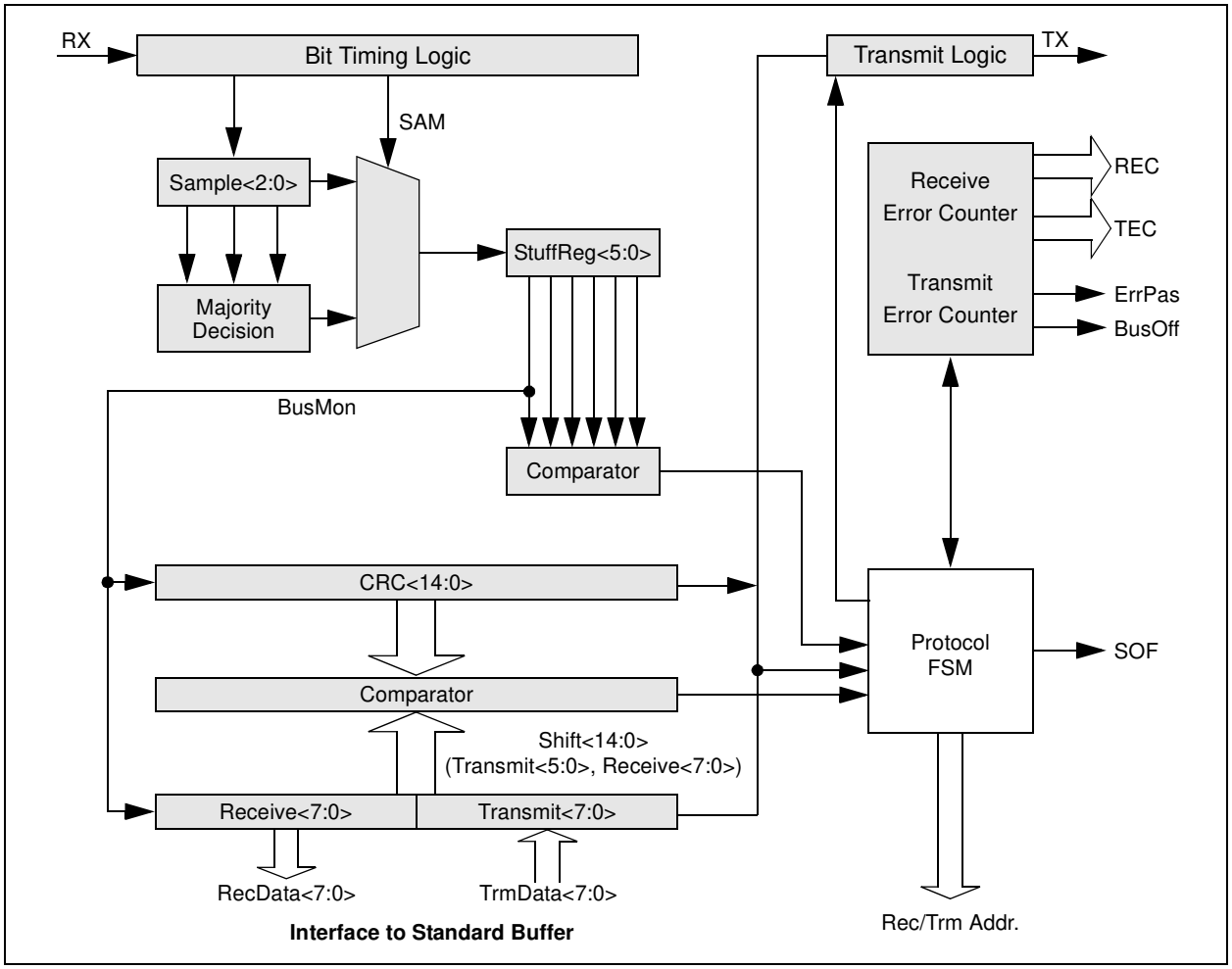
3.5.3 ERROR MANAGEMENT LOGIC

The Error Management Logic (EML) is responsible for the Fault confinement of the CAN device. Its two counters, the Receive Error Counter (REC) and the Transmit Error Counter (TEC), are incremented and decremented by commands from the bit stream processor. Based on the values of the error counters, the CAN controller is set into the states: error-active, error-passive or bus-off.

3.5.4 BIT TIMING LOGIC

The Bit Timing Logic (BTL) monitors the bus line input and handles the bus-related bit timing according to the CAN protocol. The BTL synchronizes on a Recessive-to-Dominant bus transition at Start-of-Frame (hard synchronization) and on any further Recessive-to-Dominant bus line transition if the CAN controller itself does not transmit a Dominant bit (resynchronization). The BTL also provides programmable time segments to compensate for the propagation delay time, phase shifts and to define the position of the sample point within the bit time. The programming of the BTL depends on the baud rate and external physical delay times.

FIGURE 3-3: CAN PROTOCOL ENGINE BLOCK DIAGRAM



3.6 Message Transmission

The transmit registers are described in **Section 4.1 "Message Transmit Registers"**.

3.6.1 TRANSMIT BUFFERS

The MCP25625 implements three transmit buffers. Each of these buffers occupies 14 bytes of SRAM and is mapped into the device memory map.

The first byte, TXBxCTRL, is a control register associated with the message buffer. The information in this register determines the conditions under which the message will be transmitted and indicates the status of the message transmission (see Register 4-1).

Five bytes are used to hold the Standard and Extended Identifiers, as well as other message arbitration information (see Registers 4-3 through 4-7). The last eight bytes are for the eight possible data bytes of the message to be transmitted (see Register 4-8).

At a minimum, the TXBxSIDH, TXBxSIDL and TXBxDLC registers must be loaded. If data bytes are present in the message, the TXBxDn registers must also be loaded. If the message is to use Extended Identifiers, the TXBxEIDn registers must also be loaded and the EXIDE bit in the TXBxSIDL register should be set.

Prior to sending the message, the microcontroller must initialize the TXxIE bit in the CANINTE register to enable or disable the generation of an interrupt when the message is sent.

Note:	The TXREQ bit in the TXBxCTRL register must be clear (indicating the transmit buffer is not pending transmission) before writing to the transmit buffer.

3.6.2 TRANSMIT PRIORITY

Transmit priority is a prioritization within the CAN controller of the pending transmittable messages. This is independent from, and not necessarily related to, any prioritization implicit in the message arbitration scheme built into the CAN protocol.

Prior to sending the Start-of-Frame (SOF), the priority of all buffers that are queued for transmission are compared. The transmit buffer with the highest priority will be sent first. For example, if Transmit Buffer 0 has a higher priority setting than Transmit Buffer 1, Buffer 0 will be sent first.

If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. For example, if Transmit Buffer 1 has the same priority setting as Transmit Buffer 0, Buffer 1 will be sent first.

The TXP<1:0> bits in the TXBxCTRL register (see Register 4-1) allow the selection of four levels of transmit priority for each transmit buffer individually. A buffer with the TXPx bits equal to '11' has the highest possible priority, while a buffer with the TXPx bits equal to '00' has the lowest possible priority.

3.6.3 INITIATING TRANSMISSION

In order to initiate message transmission, the TXREQ bit in the TXBxCTRL register must be set for each buffer to be transmitted. This can be accomplished by:

- Writing to the register via the SPI `WRITE` command
- Sending the SPI `RTS` command
- Setting the $\overline{\text{TxnRTS}}$ pin low for the particular transmit buffer(s) that is to be transmitted

If transmission is initiated via the SPI interface, the TXREQ bit can be set at the same time as the TXPx priority bits.

When the TXREQ is set, the ABTF, MLOA and TXERR bits in the TXBxCTRL register will be cleared automatically.

Note:	Setting the TXREQ bit in the TXBxCTRL register does not initiate a message transmission. It merely flags a message buffer as being ready for transmission. Transmission will start when the device detects that the bus is available.

Once the transmission has completed successfully, the TXREQ bit will be cleared, the TXxIF bit in the CANINTF register will be set and an interrupt will be generated if the TXxIE bit in the CANINTE register is set.

If the message transmission fails, the TXREQ bit will remain set. This indicates that the message is still pending for transmission and one of the following condition flags will be set:

- If the message started to transmit but encountered an error condition, the TXERR bit in the TXBxCTRL register and the MERRF bit in the CANINTF register will be set, and an interrupt will be generated on the $\overline{\text{INT}}$ pin if the MERRE bit in the CANINTE register is set.
- If arbitration is lost, the MLOA bit in the TXBxCTRL register will be set.

Note:	If One-Shot mode is enabled (OSM bit in the CANCTRL register), the above conditions will still exist. However, the TXREQ bit will be cleared and the message will not attempt transmission a second time.

3.6.4 ONE-SHOT MODE

One-Shot mode ensures that a message will only attempt to transmit one time. Normally, if a CAN message loses arbitration or is destroyed by an error frame, the message is retransmitted. With One-Shot mode enabled, a message will only attempt to transmit one time, regardless of arbitration loss or error frame.

One-Shot mode is required to maintain time slots in deterministic systems, such as TTCAN.

3.6.5 $\overline{\text{TxnRTS}}$ PINS

The $\overline{\text{TxnRTS}}$ pins are input pins that can be configured as:

- Request-to-Send inputs, which provide an alternative means of initiating the transmission of a message from any of the transmit buffers
- Standard digital inputs

Configuration and control of these pins is accomplished using the TXRTSCTRL register (see Register 4-2). The TXRTSCTRL register can only be modified when the CAN controller is in Configuration mode (see **Section 2.0 "Modes of Operation"**). If configured to operate as a Request-to-Send pin, the pin is mapped into the respective TXREQ bit in the TXBxCTRL register for the transmit buffer. The TXREQ bit is latched by the falling edge of the $\overline{\text{TxnRTS}}$ pin. The $\overline{\text{TxnRTS}}$ pins are designed to allow them to be tied directly to the $\overline{\text{RxnBF}}$ pins to automatically initiate a message transmission when the $\overline{\text{RxnBF}}$ pin goes low.

The $\overline{\text{TxnRTS}}$ pins have internal pull-up resistors of 100 kΩ (nominal).

3.6.6 ABORTING TRANSMISSION

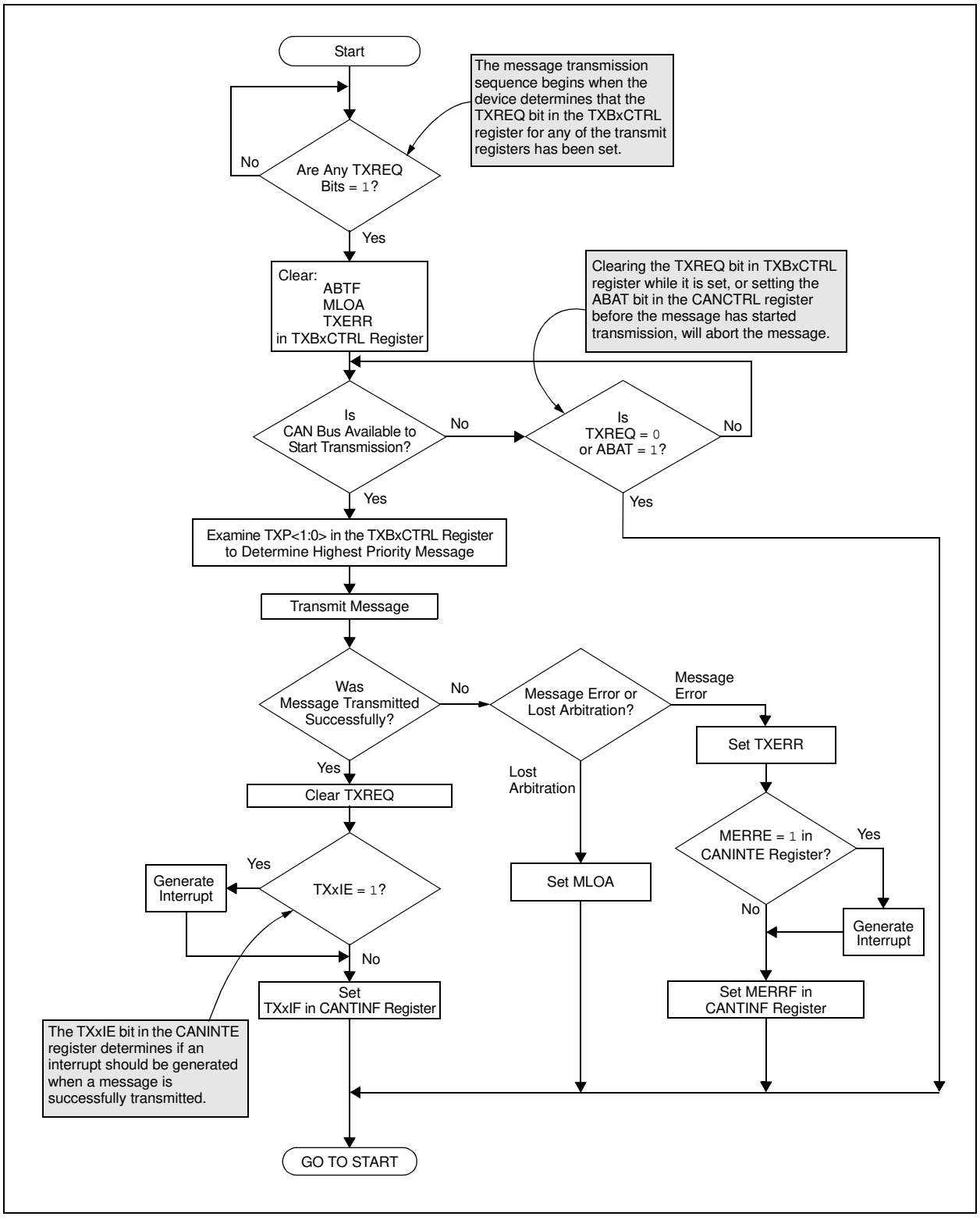
The MCU can request to abort a message in a specific message buffer by clearing the associated TXREQ bit.

In addition, all pending messages can be requested to be aborted by setting the ABAT bit in the CANCTRL register. This bit MUST be reset (typically, after the TXREQ bits have been verified to be cleared) to continue transmitting messages. The ABTF flag in the TXBxCTRL register will only be set if the abort was requested via the ABAT bit in the CANCTRL register. Aborting a message by resetting the TXREQ bit does NOT cause the ABTF bit to be set.

Note 1: Messages that were transmitting when the abort was requested will continue to transmit. If the message does not successfully complete transmission (i.e., lost arbitration or was interrupted by an error frame), it will then be aborted.

2: When One-Shot mode is enabled, if the message is interrupted due to an error frame or loss of arbitration, the ABTF bit in the TXBxCTRL register will be set.

FIGURE 3-4: TRANSMIT MESSAGE FLOWCHART



3.7 Message Reception

The registers required for message reception are described in **Section 4.2 "Message Receive Registers"**.

3.7.1 RECEIVE MESSAGE BUFFERING

The MCP25625 includes two full receive buffers with multiple acceptance filters for each. There is also a separate Message Assembly Buffer (MAB) that acts as a third receive buffer (see Figure 3-6).

3.7.1.1 Message Assembly Buffer

Of the three receive buffers, the MAB is always committed to receiving the next message from the bus. The MAB assembles all messages received. These messages will be transferred to the RXBx buffers (see Registers 4-12 to 4-17) only if the acceptance filter criteria is met.

3.7.1.2 RXB0 and RXB1

The remaining two receive buffers, called RXB0 and RXB1, can receive a complete message from the protocol engine via the MAB. The MCU can access one buffer, while the other buffer is available for message reception, or for holding a previously received message.

Note:	The entire content of the MAB is moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (Standard or Extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received.

3.7.1.3 Receive Flags/interrupts

When a message is moved into either of the receive buffers, the appropriate RXxIF bit in the CANINTF register is set. This bit must be cleared by the MCU in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the MCU has finished with the message before the CAN controller attempts to load a new message into the receive buffer.

If the RXxIE bit in the CANINTE register is set, an interrupt will be generated on the $\overline{\text{INT}}$ pin to indicate that a valid message has been received. In addition, the associated $\overline{\text{RxnBF}}$ pin will drive low if configured as a receive buffer full pin. See **Section 3.7.4 "Rx0BF and Rx1BF Pins"** for details.

3.7.2 RECEIVE PRIORITY

RXB0, the higher priority buffer, has one mask and two message acceptance filters associated with it. The received message is applied to the mask and filters for RXB0 first.

RXB1 is the lower priority buffer, with one mask and four acceptance filters associated with it.

In addition to the message being applied to the RB0 mask and filters first, the lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer.

When a message is received, the RXBxCTRL<3:0> bits will indicate the acceptance filter number that enabled reception and whether the received message is a remote transfer request.

3.7.2.1 Rollover

Additionally, the RXB0CTRL register can be configured such that, if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1, regardless of the acceptance criteria of RXB1.

3.7.2.2 RXM<1:0> Bits

The RXM<1:0> bits in the RXBxCTRL register set special Receive modes. Normally, these bits are cleared to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the RFXxSIDL register.

If the RXMx bits are set to '11', the buffer will receive all messages, regardless of the values of the acceptance filters. Also, if a message has an error before the End-of-Frame (EOF), that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode has some value in debugging a CAN system and would not be used in an actual system environment.

Setting the RXMx bits to '01' or '10' is not recommended.

3.7.3 START-OF-FRAME SIGNAL

If enabled, the Start-of-Frame signal is generated on the SOF bit at the beginning of each CAN message detected on the RxCAN pin.

The RxCAN pin monitors an Idle bus for a Recessive-to-Dominant edge. If the Dominant condition remains until the sample point, the MCP25625 interprets this as a SOF and a SOF pulse is generated. If the Dominant condition does not remain until the sample point, the MCP25625 interprets this as a glitch on the bus and no SOF signal is generated. Figure 3-5 illustrates SOF signaling and glitch filtering.

As with One-Shot mode, one use for SOF signaling is for TTCAN-type systems. In addition, by monitoring both the RxCAN pin and the SOF bit, an MCU can detect early physical bus problems by detecting small glitches before they affect the CAN communication.

3.7.4 Rx0BF AND Rx1BF PINS

In addition to the $\overline{\text{INT}}$ pin, which provides an interrupt signal to the MCU for many different conditions, the Receive Buffer Full pins ($\overline{\text{Rx0BF}}$ and $\overline{\text{Rx1BF}}$) can be used to indicate that a valid message has been loaded into RXB0 or RXB1, respectively. The pins have three different configurations (see Table 3-1):

1. Disabled
2. Buffer Full Interrupt
3. Digital Output

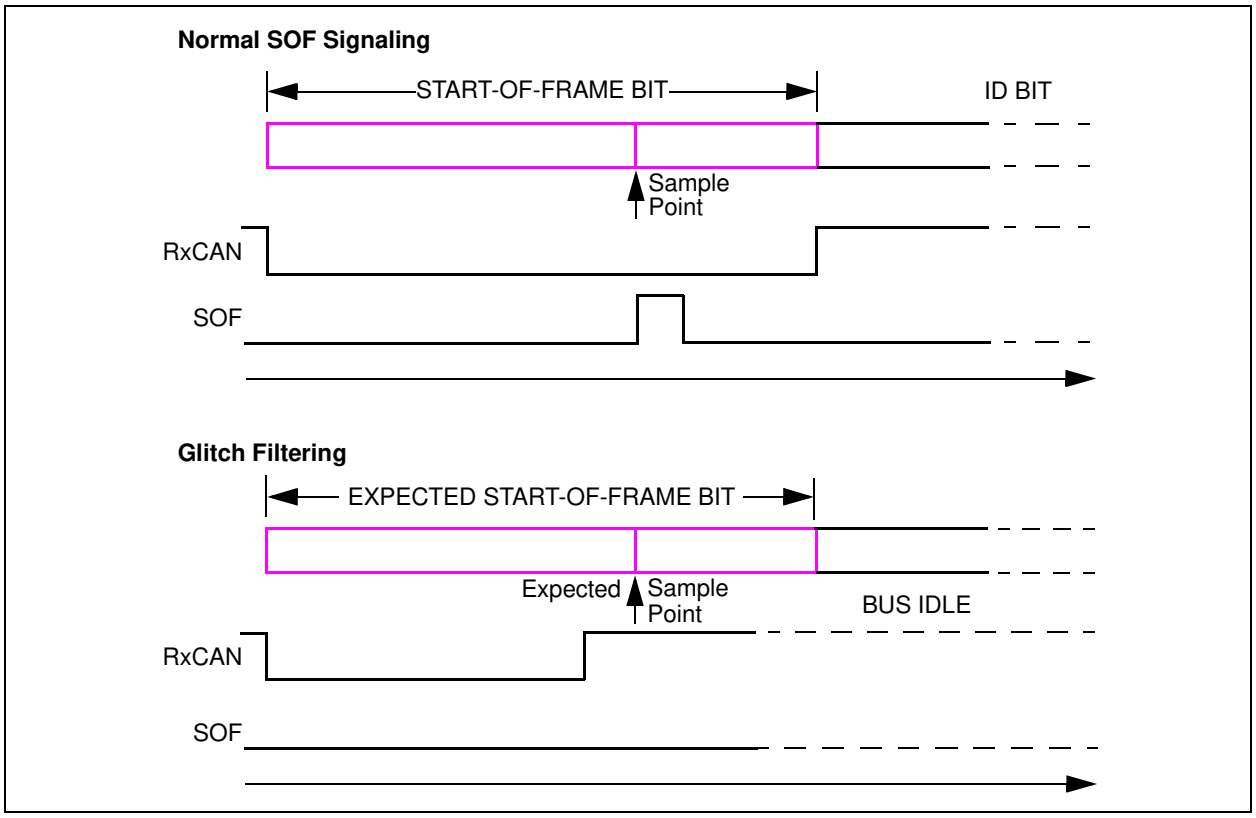
3.7.4.1 Disabled

The $\overline{\text{RxnBF}}$ pins can be disabled to the high-impedance state by clearing the BxBFE bit in the BFPCTRL register.

3.7.4.2 Configured as Buffer Full

The $\overline{\text{RxnBF}}$ pins can be configured to act as either buffer full interrupt pins or as standard digital outputs. Configuration and status of these pins is available via the BFPCTRL register (Register 4-11). When set to operate in Interrupt mode (by setting the BxBFE and BxBFM bits in the BFPCTRL register), these pins are active-low and are mapped to the RXxIF bit in the CANINTF register for each receive buffer. When this bit goes high for one of the receive buffers (indicating that a valid message has been loaded into the buffer), the corresponding $\overline{\text{RxnBF}}$ pin will go low. When the RXxIF bit is cleared by the MCU, the corresponding interrupt pin will go to the logic-high state until the next message is loaded into the receive buffer.

FIGURE 3-5: START-OF-FRAME SIGNALING



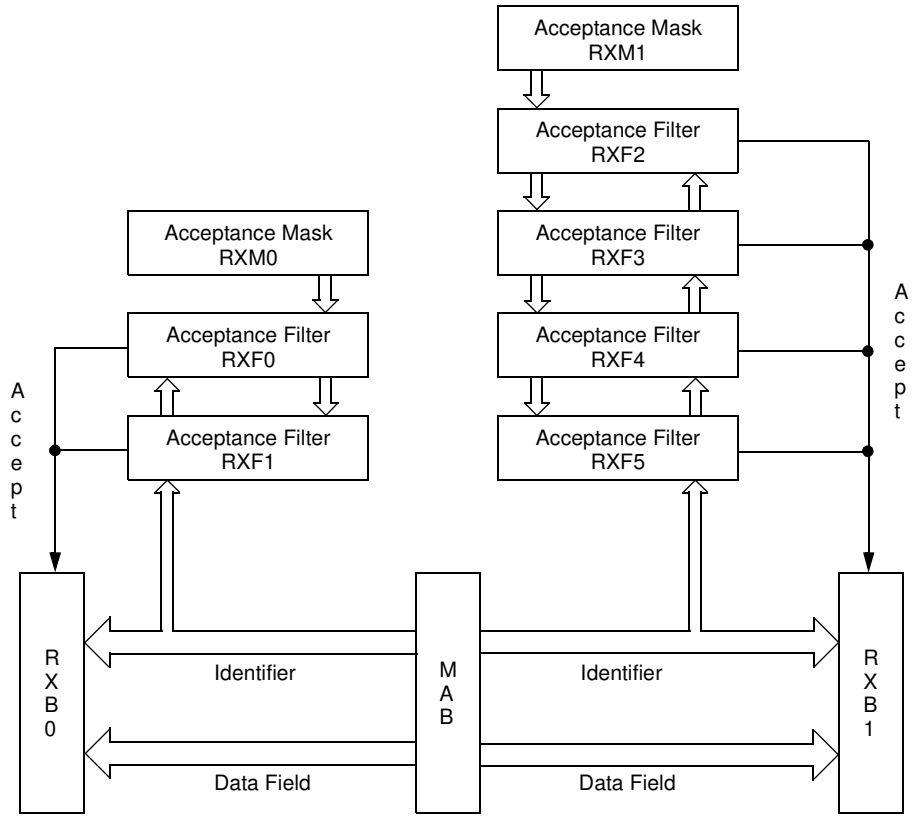
3.7.4.3 Configured as Digital Output

When used as digital outputs, the BxBFM bits in the BFPCTRL register must be cleared and the BxBFE bits must be set for the associated buffer. In this mode, the state of the pin is controlled by the BxBFS bits in the same register. Writing a '1' to the BxBFS bits will cause a high level to be driven on the associated buffer full pin, while a '0' will cause the pin to drive low. When using the pins in this mode, the state of the pin should be modified only by using the SPI `BIT MODIFY` command to prevent glitches from occurring on either of the buffer full pins.

TABLE 3-1: CONFIGURING RxnBF PINS

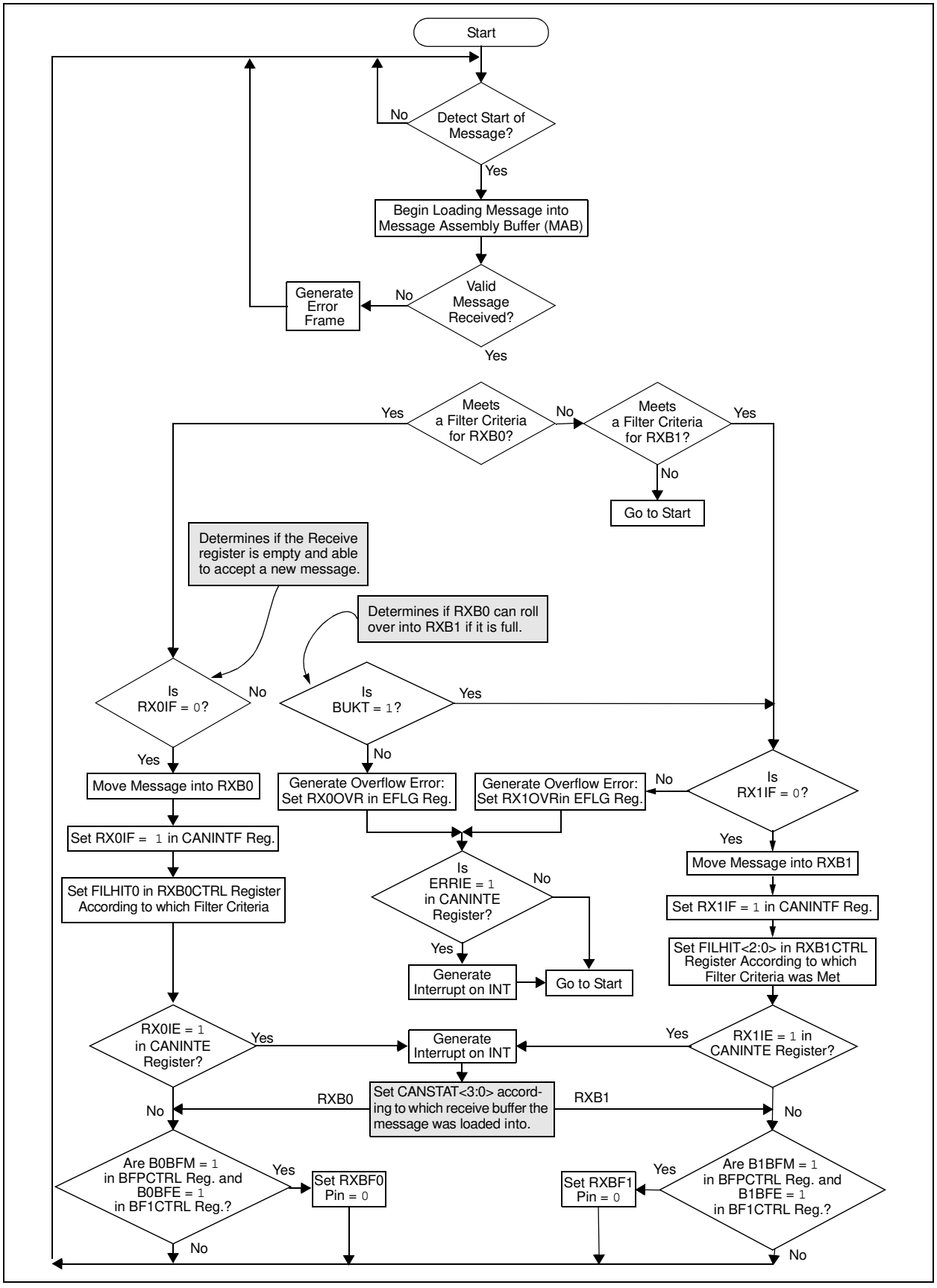
BnBFE	BnBFM	BnBFS	Pin Status
0	X	X	Disabled, high-impedance
1	1	X	Receive buffer interrupt
1	0	0	Digital output = 0
1	0	1	Digital output = 1

FIGURE 3-6: RECEIVE BUFFER BLOCK DIAGRAM



Note: Messages received in the MAB are initially applied to the mask and filters of RXB0. In addition, only one filter match occurs (e.g., if the message matches both RXF0 and RXF2, the match will be for RXF0 and the message will be moved into RXB0).

FIGURE 3-7: RECEIVE FLOW FLOWCHART



3.7.5 MESSAGE ACCEPTANCE FILTERS AND MASKS

The message acceptance filters and masks are used to determine if a message in the Message Assembly Buffer should be loaded into either of the receive buffers (see Figure 3-9). Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer.

The registers required for message filtering are described in **Section 4.3 "Acceptance Filter Registers"**.

3.7.5.1 Data Byte Filtering

When receiving standard data frames (11-bit identifier), the MCP25625 automatically applies 16 bits of masks and filters, normally associated with Extended Identifiers, to the first 16 bits of the data field (data bytes 0 and 1). Figure 3-8 illustrates how masks and filters apply to extended and standard data frames.

Data byte filtering reduces the load on the MCU when implementing Higher Layer Protocols (HLPs) that filter on the first data byte (e.g., DeviceNet™).

3.7.5.2 Filter Matching

The filter masks (see Registers 4-22 through 4-25) are used to determine which bits in the identifier are examined with the filters. A truth table is shown in Table 3-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if the message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, that bit will automatically be accepted, regardless of the filter bit.

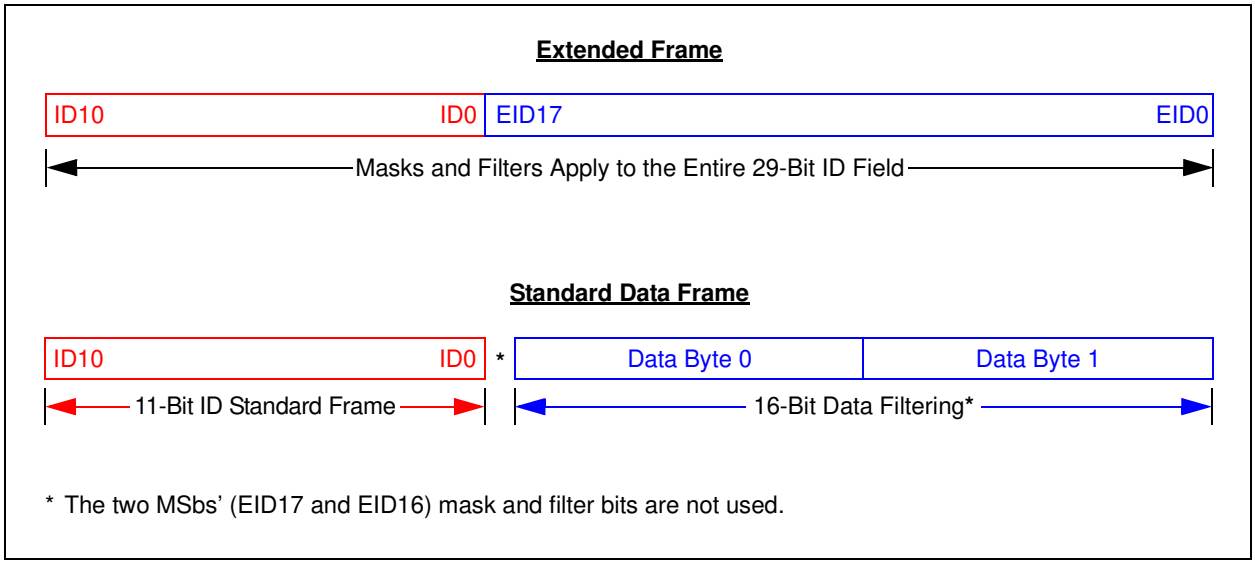
TABLE 3-2: FILTER/MASK TRUTH TABLE

Mask Bit n	Filter Bit n	Message Identifier Bit	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

Note: x = Don't care.

As shown in the Receive Buffer Block Diagram (Figure 3-6), acceptance filters, RXF0 and RXF1 (and filter mask, RXM0), are associated with RXB0. Filters, RXF2, RXF3, RXF4, RXF5 and RXM1 mask, are associated with RXB1.

FIGURE 3-8: MASKS AND FILTERS APPLIED TO CAN FRAMES



3.7.5.3 FILHIT Bits

Filter matches on received messages can be determined by the FILHIT<2:0> bits in the associated RXBxCTRL register. The FILHIT0 bit in the RXB0CTRL register is associated with Buffer 0 and the FILHIT<2:0> bits in the RXB1CTRL register are associated with Buffer 1.

The three FILHITx bits for Receive Buffer 1 (RXB1) are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

Note: '000' and '001' can only occur if the BUKT bit in RXB0CTRL is set, allowing RXB0 messages to roll over into RXB1.

RXB0CTRL contains two copies of the BUKT bit (BUKT1) and the FILHIT<0> bit.

The coding of the BUKT bit enables these three bits to be used similarly to the FILHITx bits in the RXB1CTRL register and to distinguish a hit on filters, RXF0 and RXF1, in either RXB0 or after a rollover into RXB1.

- 111 = Acceptance Filter 1 (RXB1)
- 110 = Acceptance Filter 0 (RXB0)
- 001 = Acceptance Filter 1 (RXB1)
- 000 = Acceptance Filter 0 (RXB0)

If the BUKT bit is clear, there are six codes corresponding to the six filters. If the BUKT bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to the RXF0 and RXF1 filters that roll over into RXB1.

3.7.5.4 Multiple Filter Matches

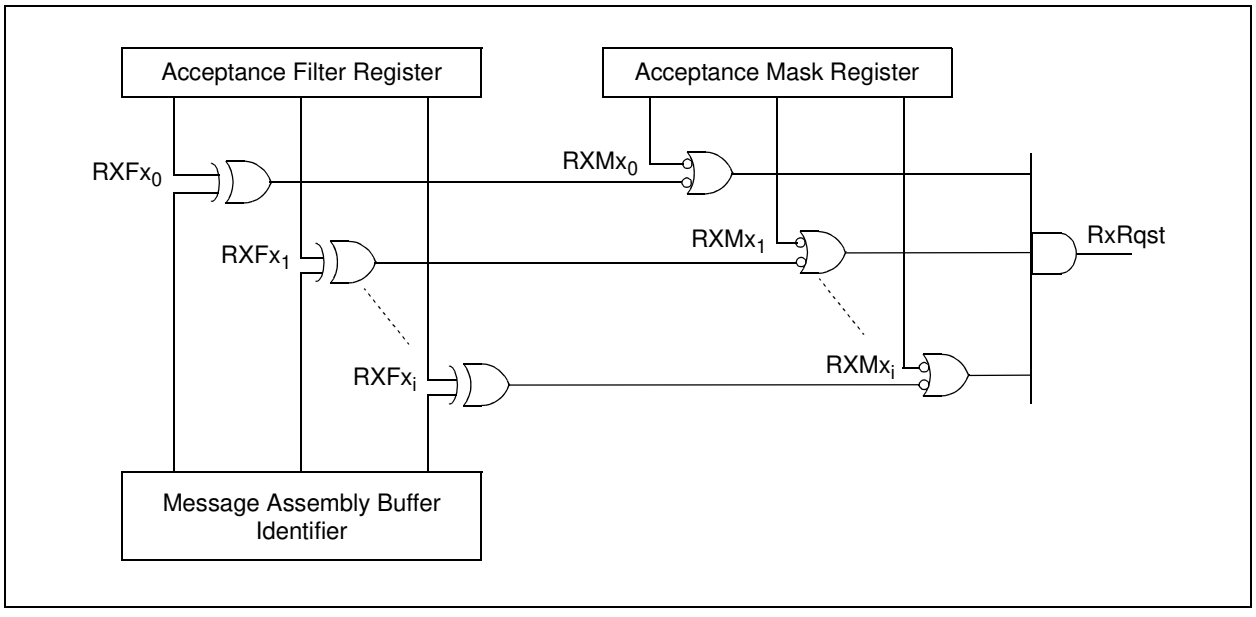
If more than one acceptance filter matches, the FILHITx bits will encode the binary value of the lowest numbered filter that matched. For example, if filters, RXF2 and RXF4, match, FILHITx will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower numbered filter having higher priority. Messages are compared to filters in ascending order of filter number. This also ensures that the message will only be received into one buffer. This implies that RXB0 has a higher priority than RXB1.

3.7.5.5 Configuring the Masks and Filters

The Mask and Filter registers can only be modified when the MCP25625 is in Configuration mode (see **Section 2.0 "Modes of Operation"**).

Note: The Mask and Filter registers read all '0's when in any mode except Configuration mode.

FIGURE 3-9: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION



3.8 CAN Bit Time

The Nominal Bit Rate (NBR) is the number of bits per second transmitted on the CAN bus (see Equation 3-1).

EQUATION 3-1: NOMINAL BIT RATE/TIME

$$NBR = \frac{1}{NBT}$$

The Nominal Bit Time (NBT) is made up of four non-overlapping segments. Each of these segments is made up of an integer number of so called Time Quanta (T_Q).

The length of each Time Quantum is based on the oscillator period (T_{OSC}). Equation 3-2 illustrates how the Time Quantum can be programmed using the Baud Rate Prescaler (BRP):

EQUATION 3-2: TIME QUANTA

$$T_Q = 2 \times (BRP<5:0> + 1) \times T_{OSC} = \frac{2 \times (BRP<5:0> + 1)}{F_{OSC}}$$

Figure 3-10 illustrates how the Nominal Bit Time is made up of four segments:

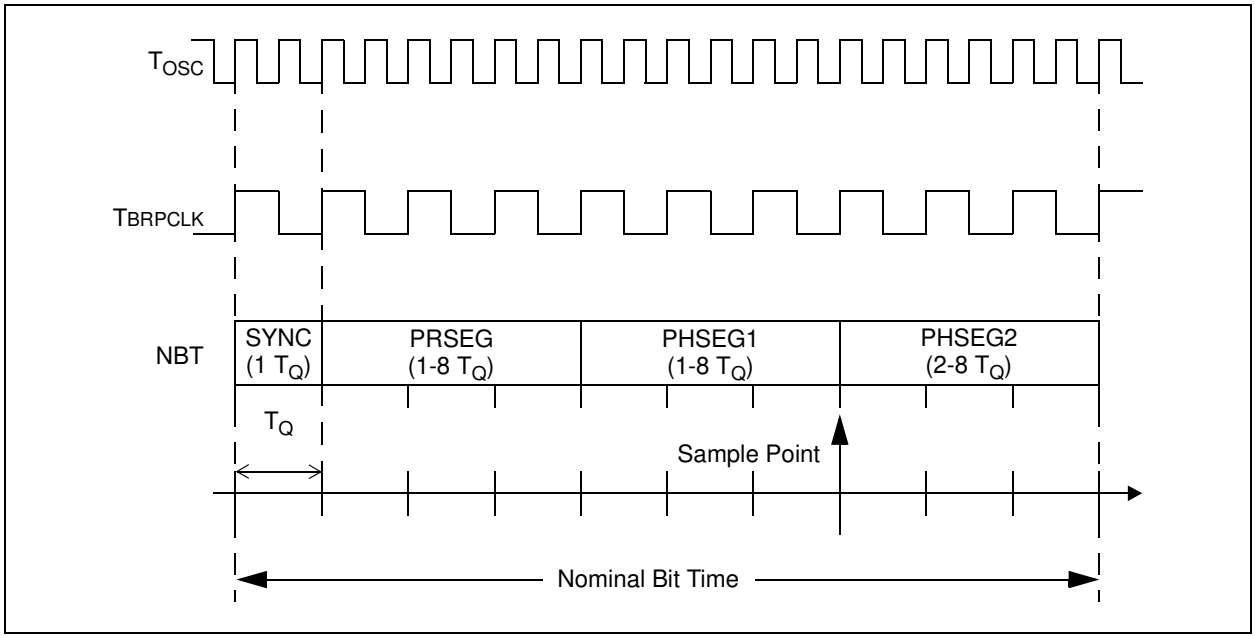
- **Synchronization Segment (SYNC)** – Synchronizes the different nodes connected on the CAN bus. A bit edge is expected to be within this segment. Based on the CAN protocol, the Synchronization Segment is 1 T_Q. See **Section 3.8.3 "Synchronization"** for more details on synchronization.
- **Propagation Segment (PRSEG)** – Compensates for the propagation delay on the bus. It is programmable from 1 to 8 T_Q.
- **Phase Segment 1 (PHSEG1)** – This time segment compensates for errors that may occur due to phase shifts in the edges. The time segment may be automatically lengthened during resynchronization to compensate for the phase shift. It is programmable from 1 to 8 T_Q.
- **Phase Segment 2 (PHSEG2)** – This time segment compensates for errors that may occur due to phase shifts in the edges. The time segment may be automatically shortened during resynchronization to compensate for the phase shift. It is programmable from 2 to 8 T_Q.

The total number of Time Quanta in a Nominal Bit Time is programmable and can be calculated using Equation 3-3.

EQUATION 3-3: T_Q PER NBT

$$\frac{NBT}{T_Q} = SYNC + PRSEG + PHSEG1 + PHSEG2$$

FIGURE 3-10: ELEMENTS OF A NOMINAL BIT TIME



3.8.1 SAMPLE POINT

The sample point is the point in the Nominal Bit Time at which the logic level is read and interpreted. The CAN bus can be sampled once or three times, as configured by the SAM bit in the CNF2 register:

- SAM = 0: The sample point is located between PHSEG1 and PHSEG2.
- SAM = 1: One sample point is located between PHSEG1 and PHSEG2. Additionally, two samples are taken at one-half T_Q intervals prior to the end of PHSEG1, with the value of the bit being determined by a majority decision.

The sample point in percent can be calculated using Equation 3-4.

EQUATION 3-4: SAMPLE POINT

$$SP = \frac{PRSEG + PHSEG1}{\frac{NBT}{T_Q}} \times 100$$

3.8.2 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time required for the CAN controller to determine the bit level of a sampled bit. The IPT for the MCP25625 is 2 T_Q. Therefore, the minimum of PHSEG2 is also 2 T_Q.

3.8.3 SYNCHRONIZATION

To compensate for phase shifts between the oscillator frequencies of the nodes on the bus, each CAN controller must be able to synchronize to the relevant edge of the incoming signal.

The CAN controller expects an edge in the received signal to occur within the SYNC segment. Only Recessive-to-Dominant edges are used for synchronization.

There are two mechanisms used for synchronization:

- **Hard Synchronization** – Forces the edge that has occurred to lie within the SYNC segment. The bit time counter is restarted with SYNC.
- **Resynchronization** – If the edge falls outside the SYNC segment, PHSEG1 and PHSEG2 will be adjusted.

For a more detailed description of the CAN synchronization, please refer to *AN754*, *"Understanding Microchip's CAN Module Bit Timing"* (DS00754) and ISO11898-1.

3.8.4 SYNCHRONIZATION JUMP WIDTH

The Synchronization Jump Width (SJW) is the maximum amount PHSEG1 and PHSEG2 can be adjusted during resynchronization. SJW is programmable from 1 to 4 T_Q.

3.8.5 OSCILLATOR TOLERANCE

According to the CAN specification, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 kbps, as a rule of thumb. For the full bus speed range of the CAN protocol, a quartz oscillator is required. A maximum node-to-node oscillator variation of 1.58% is allowed.

The oscillator tolerance (df), around the nominal frequency of the oscillator ($fnom$), is defined in Equation 3-5.

Equation 3-6 and Equation 3-7 describe the conditions for the maximum tolerance of the oscillator.

EQUATION 3-5: OSCILLATOR TOLERANCE

$$(1 - df) \times (fnom \leq F_{OSC} \leq (1 + df) \times fnom)$$

EQUATION 3-6: CONDITION 1

$$df \leq \frac{SJW}{2 \times 10 \times \frac{NBT}{T_Q}}$$

EQUATION 3-7: CONDITION 2

$$df \leq \frac{min(PHSEG1, PHSEG2)}{2 \times \left(13 \times \frac{NBT}{T_Q} - PHSEG2\right)}$$

3.8.6 PROPAGATION DELAY

Figure 3-11 illustrates the propagation delay between two CAN nodes on the bus. Assuming Node A is transmitting a CAN message, the transmitted bit will propagate from the transmitting CAN Node A, through the transmitting CAN transceiver, over the CAN bus, through the receiving CAN transceiver into the receiving CAN Node B.

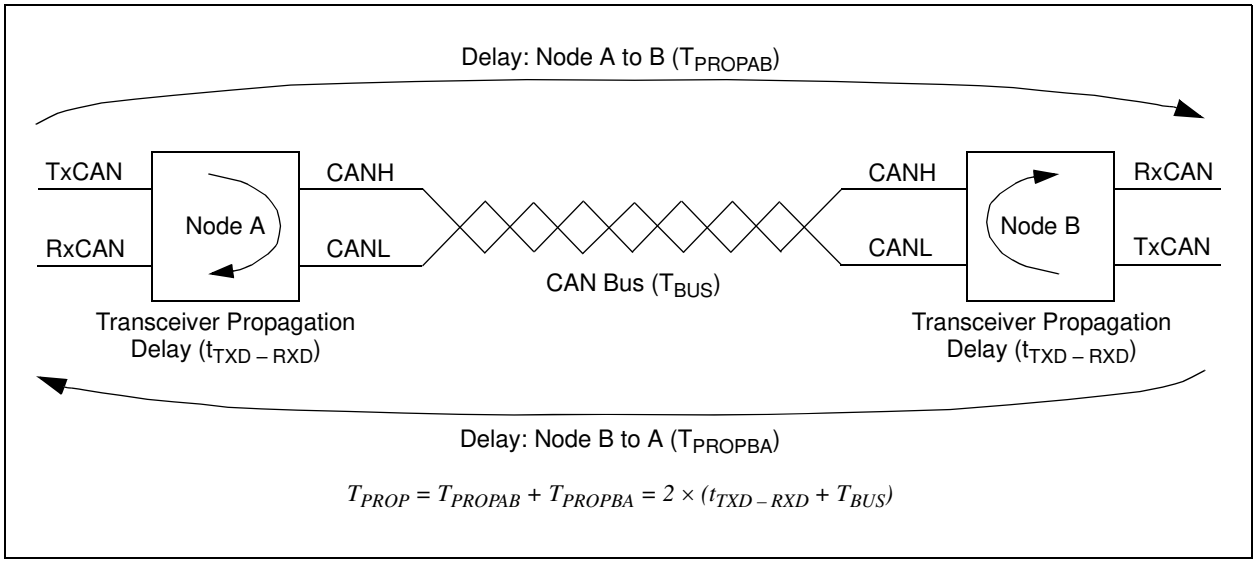
During the arbitration phase of a CAN message, the transmitter samples the bus and checks if the transmitted bit matches the received bit. The transmitting node has to place the sample point after the maximum propagation delay.

Equation 3-8 describes the maximum propagation delay; where $t_{TXD-RXD}$ is the propagation delay of the transceiver, 235 ns for the MCP25625; T_{BUS} is the delay on the CAN bus, approximately 5 ns/m. The factor two comes from the worst case, when Node B starts transmitting exactly when the bit from Node A arrives.

EQUATION 3-8: MAXIMUM PROPAGATION DELAY

$$T_{PROP} = 2 \times (t_{TXD-RXD} + T_{BUS})$$

FIGURE 3-11: PROPAGATION DELAY



3.8.7 BIT TIME CONFIGURATION EXAMPLE

The following example illustrates the configuration of the CAN Bit Time registers. Assuming we want to set up a CAN network in an automobile with the following parameters:

- 500 kbps Nominal Bit Rate (NBR)
- Sample point between 60 and 80% of the Nominal Bit Time (NBT)
- 40 meters minimum bus length

Table 3-3 illustrates how the bit time parameters are calculated. Since the parameters depend on multiple constraints and equations, and are calculated using an iterative process, it is recommended to enter the equations into a spread sheet.

A detailed description of the Bit Time Configuration registers can be found in **Section 4.4 "Bit Time Configuration Registers"**.

TABLE 3-3: STEP-BY-STEP REGISTER CONFIGURATION EXAMPLE

Parameter	Register	Constraint	Value	Unit	Equations and Comments
NBT	—	NBT ≥ 1 µs	2	µs	Equation 3-1
F_{OSC}	—	F_{OSC} ≤ 25 MHz	16	MHz	Select crystal or resonator frequency; usually 16 or 20 MHz work
T_Q/Bit	—	5 to 25	16		The sum of the T_Q of all four segments must be between 5 and 25; selecting 16 T_Q per bit is a good starting point
T_Q	—	NBT, F_{OSC}	125	ns	Equation 3-3
BRP<5:0>	CNF1	0 to 63	0		Equation 3-2
SYNC	—	Fixed	1	T_Q	Defined in ISO 11898-1
PRSEG	CNF2	1 to 8 T_Q; PRSEG > T_{PROP}	7	T_Q	Equation 3-8: T_{PROP} = 870 ns, minimum PRSEG = T_{PROP}/T_Q = 6.96 T_Q; selecting 7 will allow 40m bus length
PHSEG1	CNF2	1 to 8 T_Q; PHSEG1 ≥ SJW<1:0>	4	T_Q	There are 8 T_Q remaining for PHSEG1 + PHSEG2; divide the remaining T_Q in half to maximize SJW<1:0>
PHSEG2	CNF3	2 to 8 T_Q; PHSEG2 ≥ SJW<1:0>	4	T_Q	There are 4 T_Q remaining
SJW<1:0>	CNF1	1 to 4 T_Q; SJW<1:0> ≤ min(PHSEG1, PHSEG2)	4	T_Q	Maximizing SJW<1:0> lessens the requirement for the oscillator tolerance
Sample Point	—	Usually between 60 and 80%	69	%	Use Equation 3-4 to double check the sample point
Oscillator Tolerance Condition 1	—	Double Check	1.25	%	Equation 3-6
Oscillator Tolerance Condition 2	—	Double Check	0.98	%	Equation 3-7; better than 1% crystal oscillator required

3.9 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

The registers required for error detection are described in **Section 4.5 "Error Detection Registers"**.

3.9.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence from the Start-of-Frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated; the message is repeated.

3.9.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which has been sent out as a Recessive bit) contains a Dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

3.9.3 FORM ERROR

If a node detects a Dominant bit in one of the four segments (including End-of-Frame, inter-frame space, Acknowledge delimiter or CRC delimiter), a form error has occurred and an error frame is generated. The message is repeated.

3.9.4 BIT ERROR

A bit error occurs if a transmitter detects the opposite bit level to what it transmitted (i.e., transmitted a Dominant and detected a Recessive, or transmitted a Recessive and detected a Dominant).

Exception: In the case where the transmitter sends a Recessive bit, and a Dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

3.9.5 STUFF ERROR

If, between the Start-of-Frame and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit-stuffing rule has been violated. A stuff error occurs and an error frame is generated; the message is repeated.

3.9.6 ERROR STATES

Detected errors are made known to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states according to the value of the internal error counters:

- Error-active
- Error-passive
- Bus-off (transmitter only)

The error-active state is the usual state where the node can transmit messages and active error frames (made of Dominant bits) without any restrictions.

In the error-passive state, messages and passive error frames (made of Recessive bits) may be transmitted.

The bus-off state makes it temporarily impossible for the station to participate in the bus communication. During this state, messages can neither be received nor transmitted. Only transmitters can go bus-off.

3.10 Error Modes and Error Counters

The MCP25625 contains two error counters: the Receive Error Counter (REC) (see Register 4-30) and the Transmit Error Counter (TEC) (see Register 4-29). The values of both counters can be read by the MCU. These counters are incremented/decremented in accordance with the CAN bus specification.

The MCP25625 is error-active if both error counters are below the error-passive limit of 128.

The device is error-passive if at least one of the error counters equals or exceeds 128.

The device goes to bus-off if the TEC exceeds the bus-off limit of 255. The device remains in this state until the bus-off recovery sequence is received. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive Recessive bits (see Figure 3-12).

Note:	The MCP25625, after going bus-off, will recover back to error-active without any intervention by the MCU if the bus remains Idle for 128 x 11 bit times. If this is not desired, the error Interrupt Service Routine (ISR) should address this.

The current Error mode of the MCP25625 can be read by the MCU via the EFLG register (see Register 4-31).

Additionally, there is an error state warning flag bit (EWARN bit in the EFLG register), which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.