LogiCORE IP CAN v3.2

DS265 April 19, 2010 **Product Specification**

Introduction

The LogiCORE™ IP Controller Area Network (CAN) product specification describes the architecture and features of the Xilinx CAN controller core and the functionality of the various registers in the design. In addition, the CAN core user interface and its customization options are described. Defining the CAN protocol is outside the scope of this document, and knowledge of the specifications described in the References section is assumed.

Features

- Conforms to the *ISO 11898 -1, CAN 2.0A*, and *CAN 2.0B* standards
- Industrial (I) and Extended Temperature Range (Q) grade device support
- Supports both standard (11-bit identifier) and extended (29-bit identifier) frames
- Supports bit rates up to 1 Mbps
- Transmit message FIFO with a user-configurable depth of up to 64 messages
- Transmit prioritization through one High-Priority Transmit buffer
- Automatic re-transmission on errors or arbitration loss
- Receive message FIFO with a user-configurable depth of up to 64 messages
- Acceptance filtering (through a user-configurable number) of up to 4 acceptance filters
- Sleep Mode with automatic wakeup
- Loop Back Mode for diagnostic applications
- Maskable Error and Status Interrupts
- Readable Error Counters

LogiCORE IP Facts					
Core Specifics					
Supported Device Family[1]	XA[2] Spartan®-3, XA Spartan-3E, XA Spartan-3A, XA Spartan-3A DSP Spartan-3, Spartan-3E, Spartan-3A/3AN/3ADSP, Spartan-6, Virtex®-4, Virtex-5, Virtex-6, XA Spartan-6				
Resources Used	I/O[3]	LUTs	FFs	Block RAMs	Slices
	3	868 – 1056	411 – 593	2	569 – 885
Provided with Core					
Documentation	Product Specification Getting Started Guide				
Design Tool Requirements					
Xilinx Implementation Tools	ISE® v12.1				
Verification and Simulation	Mentor Graphics ModelSim 6.5c and above Cadence Incisive Enterprise Simulator (IES) v9.2 and above				
Synthesis	XST 12.1 Synopsys Synplify PRO D-2009.12				
Support					
Provided by Xilinx, Inc. @ www.xilinx.support.					

1. For the complete list of supported devices, see the 12.1 release notes for this core (Release Notes Guide XTP025).
2. Xilinx Automotive (XA)
3. External I/O only.

Functional Overview

Figure 1 illustrates the high-level architecture of the CAN core. Descriptions of the submodules follow.

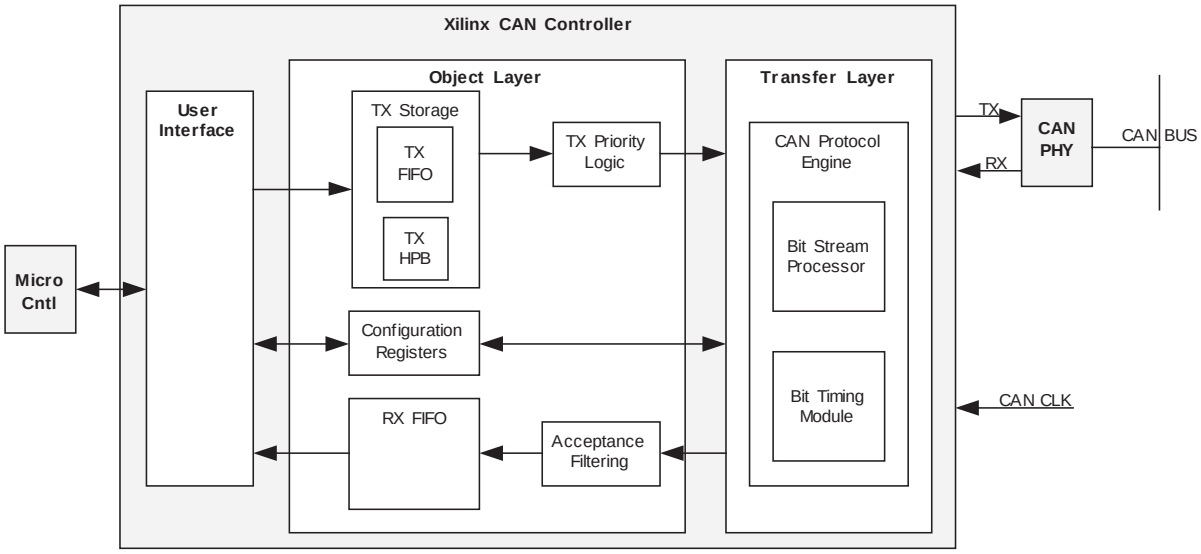

Figure 1: **CAN Core Block Diagram**

The CAN core requires an external 3.3 V compatible PHY chip.

Configuration Registers

Table 4 defines the configuration registers. This module allows for read and write access to the registers through the external micro-controller interface.

Transmit and Receive Messages

Separate storage buffers exist for transmit (TX FIFO) and receive (RX FIFO) messages through a FIFO structure. The depth of each buffer is individually configurable up to a maximum of 64 messages.

TX High Priority Buffer

The Transfer High Priority Buffer (TX HPB) provides storage for one transmit message. Messages written on this buffer have maximum transmit priority. They are queued for transmission immediately after the current transmission is complete, preempting any message in the TX FIFO.

Acceptance Filters

Acceptance Filters sort incoming messages with the user-defined acceptance mask and ID registers to determine whether to store messages in the RX FIFO, or to acknowledge and discard them. The number of acceptance filters can be configured from 0 to 4. Messages passed through acceptance filters are stored in the RX FIFO.

CAN Protocol Engine

The CAN protocol engine consists primarily of the Bit Timing Logic (BTL) and the Bit Stream Processor (BSP) modules.

Figure 2 illustrates a block diagram of the CAN protocol engine.

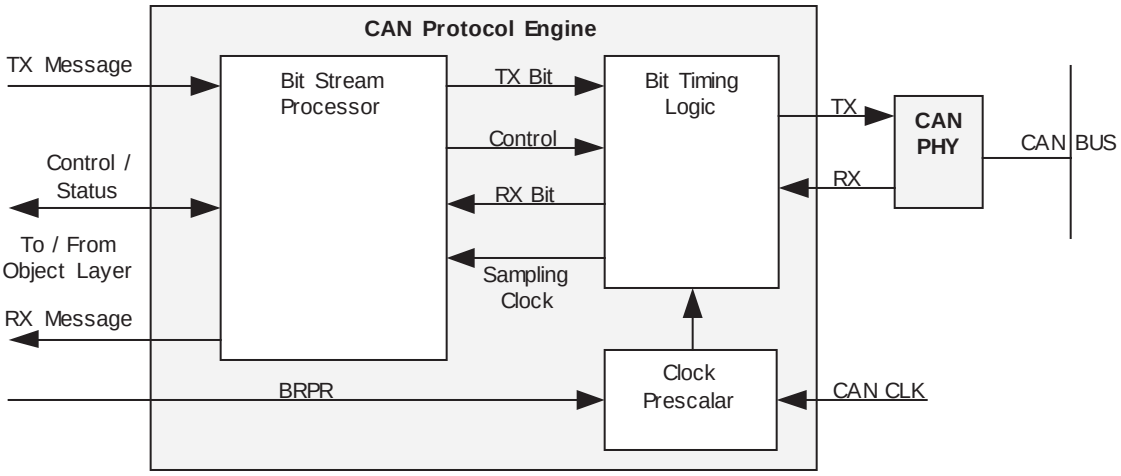

Figure 2: **CAN Protocol Engine**

Bit Timing Logic

The primary functions of the Bit Timing Logic (BTL) module include:

- Synchronizing the CAN controller to CAN traffic on the bus
- Sampling the bus and extracting the data stream from the bus during reception
- Inserting the transmit bit stream onto the bus during transmission
- Generating a sampling clock for the BSP module state machine

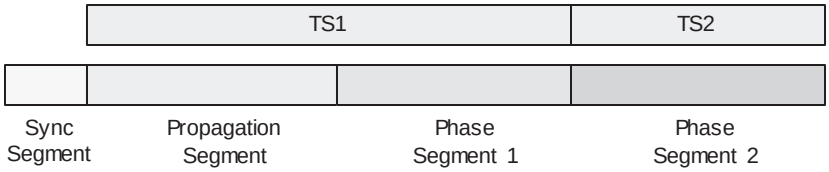

Figure 3: **CAN Bit Timing**

Figure 3 illustrates the CAN bit-time divided into four parts:

- Sync segment
- Propagation segment
- Phase segment 1
- Phase segment 2

The four bit-time parts are comprised of a number of smaller segments of equal length called *time quanta* (tq). The length of each time quantum is equal to the quantum clock time period (period = tq). The quantum clock is generated internally by dividing the incoming oscillator clock by the baud rate pre-scaler. The pre-scaler value is passed to the BTL module through the Baud Rate Prescaler (BRPR) register.

The propagation segment and phase segment 1 are joined together and called 'time segment1' (TS1), while phase segment 2 is called 'time segment2' (TS2). The number of time quanta in TS1 and TS2 vary with different networks and are specified in the Bit Timing Register (BTR), which is passed to the BTL module. The Sync segment is always one time quantum long.

The BTL state machine runs on the quantum clock. During the SOF bit of every CAN frame, the state machine is instructed by the Bit Stream Processor module to perform a hard sync, forcing the recessive (r) to dominant edge (d) to lie in the sync segment. During the rest of the recessive-to-dominant edges in the CAN frame, the BTL is prompted to perform re-synchronization.

During re-synchronization, the BTL waits for a recessive-to-dominant edge. After that occurs, it calculates the time difference (number of tqs) between the edge and the nearest sync segment. To compensate for this time difference, and to force the sampling point to occur at the correct instant in the CAN bit time, the BTL modifies the length of phase segment 1 or phase segment 2.

The maximum amount by which the phase segments can be modified is dictated by the Synchronization Jump Width (SJW) parameter, which is also passed to the BTL through the BTR. The length of the bit time of subsequent CAN bits are unaffected by this process. This synchronization process corrects for propagation delays and oscillator mismatches between the transmitting and receiving nodes.

After the controller is synchronized to the bus, the state machine waits for a time period of TS1 and then samples the bus, generating a digital '0' or '1.' This is passed on to the BSP module for higher level tasks.

Bit Stream Processor

The Bit Stream Processor (BSP) module performs several MAC/LLC functions during reception (RX) and transmission (TX) of CAN messages. The BSP receives a message for transmission from either the TX FIFO or the TX HPB and performs the following functions before passing the bit-stream to BTL.

- Serializing the message
- Inserting stuff bits, CRC bits, and other protocol defined fields during transmission

During transmission the BSP simultaneously monitors RX data and performs bus arbitration tasks. It then transmits the complete frame when arbitration is won, and retrying when arbitration is lost.

During reception the BSP removes stuff bits, CRC bits, and other protocol fields from the received bit stream. The BSP state machine also analyses bus traffic during transmission and reception for Form, CRC, ACK, Stuff and Bit violations. The state machine then performs error signaling and error confinement tasks. The CAN controller will not voluntarily generate overload frames but will respond to overload flags detected on the bus.

This module determines the error state of the CAN controller: Error Active, Error Passive or Bus-off. When TX or RX errors are observed on the bus, the BSP updates the transmit and receive error counters according to the rules defined in the CAN 2.0 A, CAN 2.0 B and ISO 11898-1 standards. Based on the values of these counters, the error state of the CAN controller is updated by the BSP.

User Interface

The external interface of the CAN controller is a subset of the IPIC Signaling (of the OPB IPIF; see reference 3). This allows the CAN controller to be interfaced to many micro controllers.

Table 1 defines CAN controller interface signalling.

Table 1: **CAN Controller External I/Os**

No.	IPIC Name	I/O	Default Value	Description
1	Bus2IP_Reset	Input		Active high reset
2	Bus2IP_Data(0:31)	Input		Write Data bus
3	Bus2IP_Addr(0:5)	Input		Address Bus
4	Bus2IP_RNW	Input		Read or Write signaling '1' for a Read Transaction '0' for a Write Transaction
5	Bus2IP_CS	Input		Active high CS
6	IP2Bus_Data(0:31)	Output	X"00000000"	Read Data bus
7	IP2Bus_Ack	Output	0	R/W data acknowledgement
8	IP2Bus_IntrEvent	Output	0	Active high interrupt line[1]
9	IP2Bus_Error	Output	0	Active high R/W Error signal. Reserved for future use.
10	CAN_PHY_TX	Output	1	CAN bus transmit signal to PHY
11	CAN_PHY_RX	Input		CAN bus receive signal from PHY
12	CAN_CLK	Input		24 Mhz oscillator clock input
13	SYS_CLK	Input		Input interface clock

1. The interrupt line is an edge-sensitive interrupt. Interrupts are indicated by the transition of the interrupt line logic from 0 to 1.

The CAN controller supports two modes of transfer:

- Single read
- Single write

Single Read Transaction

When the transfer is enabled (Bus2IP_CS = '1' and Bus2IP_RNW = '1') for a read operation, the core samples the address on the Bus2IP_Addr pins and returns the corresponding read data on the IP2Bus_Data pins. Read data is returned on a successive clock rising edge, after a wait time. IP2Bus_Ack is asserted when the data is ready on the IP2Bus_Data pins. The address is assumed to be valid on the Bus2IP_Addr pins when Bus2IP_CS is asserted. The core samples the address on the next rising edge of SYS_CLK.

IP2Bus_Ack is asserted for all read transactions, regardless of whether the transaction is valid or not. Successive read operations require that the Bus2IP_CS be deasserted and reasserted. The timing diagram for a single read transaction is shown in Figure 5.

For single read transactions, note the following:

- Transactions from address locations defined as reserved return all 0s on the IP2Bus_Data bus
- Transactions from write only address locations return all 0s on the IP2Bus_Data bus
- Transactions from AFR register return all 0s on the IP2Bus_Data bus when the number of acceptance filters = 0
- Transactions on the AFIR and AFMR address locations return all 0s on the IP2Bus_Data bus when the number of acceptance filters = 0
- Transactions on any or all of the AFIR and AFMR address locations return the data that was written to these locations when the number of acceptance filters > 0
- Read transactions on an empty RX FIFO return invalid data

Single Write Transaction

When the transfer is enabled (Bus2IP_CS = '1' and Bus2IP_RNW = '0') for a write operation, the core samples both address and data from the Bus2IP_Addr and Bus2IP_Data pins, respectively. IP2Bus_Ack is asserted on a successive clock rising edge. The address on the Bus2IP_Addr bus and data and Bus2IP_Data bus are assumed to be valid when Bus2IP_CS is asserted.

IP2Bus_Ack is asserted for all write transactions, regardless of whether the transaction is valid or not. Successive write operations require that Bus2IP_CS be deasserted and reasserted.

For single-write transactions, the following applies:

- Address locations defined as reserved are not written to
- Address locations defined as read only are not written to
- AFR, AFIR, AFMR address locations are not written to when the number of acceptance filters = 0
- All the AFIR and AFMR address locations can be written to when the number of acceptance filters > 0

Interface Timing

Figure 4 and Figure 5 illustrate the timing diagrams for read and write operations,

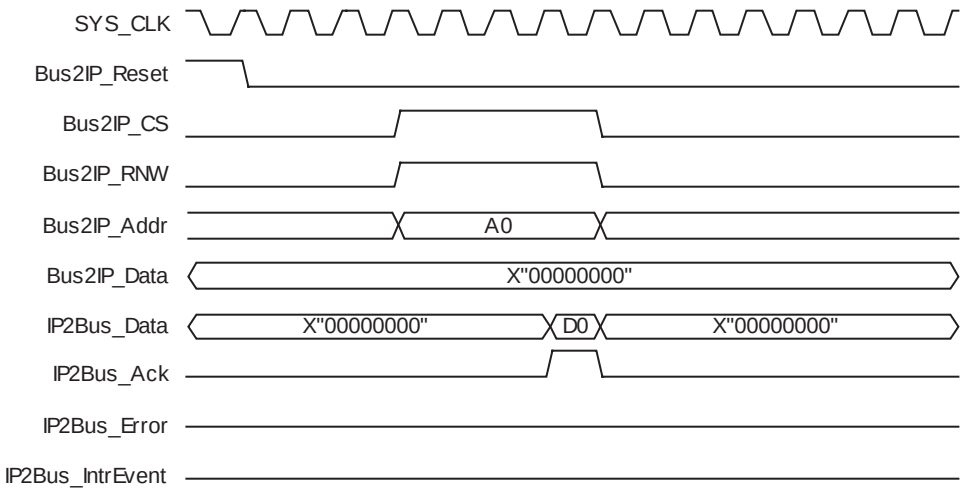

Figure 4: **Single Read Transactions**

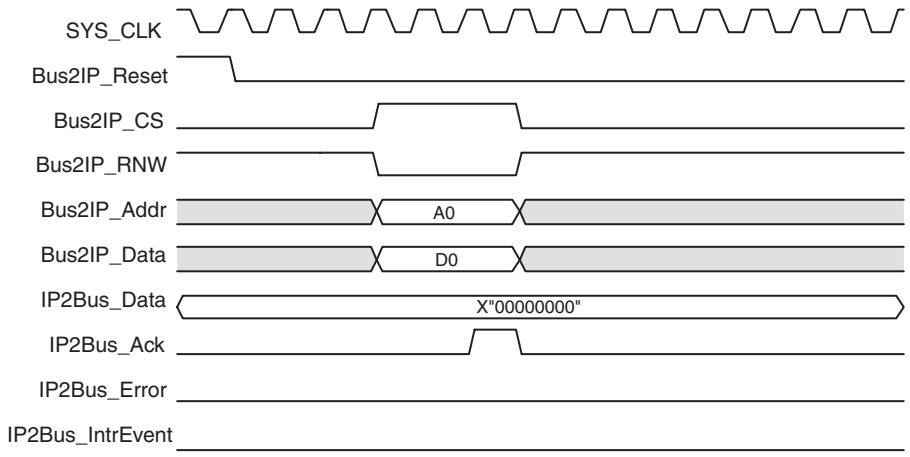

Figure 5: **Singe Write Transactions**

Operational CAN Controller Modes

The CAN controller supports the following modes of operation:

- Configuration
- Normal
- Sleep
- Loop Back

Table 2 defines the CAN Controller modes of operation and corresponding control and status bits. Figure 6 shows the Mode Transition Diagram and the conditions for mode transition. Inputs that affect the mode transitions are defined in "Xilinx CAN Controller Configuration Register Descriptions" on page 12.

Table 2: **CAN Controller Modes of Operation**

Bus2IP Reset	SRST Bit (SRR)	CEN Bit (SRR)	LBACK Bit (MSR)	SLEEP Bit (MSR)	Status Register Bits (SR) (Read Only)				Operation Mode
					CONFIG	LBACK	SLEEP	NORMAL	
'1'	X	X	X	X	'1'	'0'	'0'	'0'	core is Reset
'0'	'1'	X	X	X	'1'	'0'	'0'	'0'	core is Reset
'0'	'0'	'0'	X	X	'1'	'0'	'0'	'0'	Configuration Mode
'0'	'0'	'1'	'1'	X	'0'	'1'	'0'	'0'	Loop Back Mode
'0'	'0'	'1'	'0'	'1'	'0'	'0'	'1'	'0'	Sleep Mode
'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'1'	Normal Mode

Configuration Mode

The CAN controller enters Configuration mode, irrespective of the operation mode, when any of the following actions are performed:

- Writing a '0' to the CEN bit in the SRR register.
- Writing a '1' to the SRST bit in the SRR register. The core enters Configuration mode immediately following the software reset.
- Driving a '1' on the `Bus2IP_Reset` input. The core continues to be in reset as long as `Bus2IP_Reset` is '1'. The core enters Configuration mode after `Bus2IP_Reset` is negated to '0'.

Configuration Mode Features

- CAN controller loses synchronization with the CAN bus and drives a constant recessive bit on the bus line.
- ECR register is reset.
- ESR register is reset.
- BTR and BRPR registers can be modified.
- CONFIG bit in the Status Register is '1.'
- CAN controller will not receive any new messages.
- CAN controller will not transmit any messages. Messages in the TX FIFO and the TX high priority buffer are pended. These packets are sent when normal operation is resumed.
- Reads from the RX FIFO can be performed.
- Writes to the TX FIFO and TX HPB can be performed.
- Interrupt Status Register bits ARBLST, TXOK, RXOK, RXOFLW, ERROR, BSOFF, SLP and WKUP will be cleared.
- Interrupt Status Register bits RXNEMP, RXUFLW can be set due to read operations to the RX FIFO.
- Interrupt Status Register bits TXBFLL and TXFLL, and the Status Register bits TXBFLL and TXFLL, can be set due to write operations to the TX HPB and TX FIFO, respectively.
- Interrupts are generated if the corresponding bits in the IER are '1.'
- All Configuration Registers are accessible.

Once in Configuration mode, the CAN controller stays in this mode until the CEN bit in the SRR register is set to '1.' After the CEN bit is set to '1' the CAN controller waits for a sequence of 11 recessive bits before exiting Configuration mode.

The CAN controller enters Normal, Loop Back, or Sleep modes from Configuration mode, depending on the LBACK and SLEEP bits in the MSR Register.

Normal Mode

In Normal mode, the CAN controller participates in bus communication by transmitting and receiving messages. From Normal mode, the CAN controller can enter either Configuration or Sleep modes.

For Normal mode, the CAN controller normal mode state transitions include the following:

- Enters Configuration mode when any configuration condition is satisfied
- Enters Sleep mode when the SLEEP bit in the MSR is '1'
- Enters Normal mode from Configuration mode only when LBACK and SLEEP bits in the MSR are '0' and CEN bit is '1'
- Enters Normal mode from Sleep mode when a wake-up condition occurs

Sleep Mode

The CAN controller enters Sleep mode from Configuration mode when the LBACK bit in MSR is '0,' the SLEEP bit in MSR is '1,' and the CEN bit in SRR is '1.' The CAN controller enters Sleep mode only when there are no pending transmission requests from either the TX FIFO or the TX High Priority Buffer.

The CAN controller enters Sleep mode from Normal mode only when the SLEEP bit is '1,' the CAN bus is idle, and there are no pending transmission requests from either the TX FIFO or TX High Priority Buffer.

When another node transmits a message, the CAN controller receives the transmitted message and exits Sleep mode. When the controller is in Sleep mode, if there are new transmission requests from either the TX FIFO or the TX High Priority Buffer, these requests are serviced, and the CAN controller exits Sleep mode. Interrupts are generated when the CAN controller enters Sleep mode or wakes up from Sleep mode. From sleep mode, the CAN controller may enter either the Configuration or Normal modes.

The CAN controller can enter Configuration mode when any configuration condition is satisfied, and enters Normal mode under the following (wake-up) conditions:

- Whenever the SLEEP bit is set to '0'
- Whenever the SLEEP bit is '1,' and bus activity is detected
- Whenever there is a new message in the TX FIFO or the TX High Priority Buffer

Loop Back Mode

In Loop Back mode, the CAN controller transmits a recessive bit stream on to the CAN Bus. Any message transmitted is looped back to the RX line and is acknowledged. The CAN controller receives any message that it transmits. It does not participate in normal bus communication and does not receive any messages transmitted by other CAN nodes.

This mode is used for diagnostic purposes—when in Loop Back mode, the CAN controller may only enter Configuration mode. The CAN controller enters Configuration mode when any of the configuration conditions are satisfied.

The CAN controller enters Loop Back mode from the Configuration mode if the LBACK bit in MSR is '1' and the CEN bit in SRR is '1.'

Clocking and Reset

Clocking

The CAN core has two clocks: CAN CLK and SYS CLK. There is no fixed-frequency dependency between the two clocks. The following conditions apply for clock frequencies:

- CAN CLK can be 8 to 24 MHz in frequency
- SYS CLK can be 8 to 100 MHz in frequency
- CAN CLK and SYS CLK can be asynchronous or can be clocked from the same source

Either of these clocks can be sourced from external oscillator sources or generated within the FPGA. The oscillator used for CAN CLK must be compliant with the oscillator tolerance range given in the ISO 11898 -1, CAN 2.0A and CAN 2.0B standards.

SYS_CLK

You can specify the operating frequency for SYS_CLK; using a DCM to generate sys_clk is optional.

CAN_CLK

The range of CAN_CLK clock is 8–24 MHz.

You determine whether a DCM or an external oscillator is used to generate the CAN_CLK. If an external oscillator is used, it should meet the tolerance requirements specified in the *ISO 11898-1, CAN 2.0A* and *CAN 2.0B* standards.

Reset Mechanism

Two different reset mechanisms are provided for the CAN controller. The `Bus2IP_Reset` input mentioned in Table 1 acts as the system reset. Apart from the system reset, a software reset is provided through the SRST bit in the SRR register. The software and system reset both reset the complete CAN core (both the Object Layer and the Transfer Layer as shown in Figure 1.)

Software Reset

The software reset can be enabled by writing a '1' to the SRST bit in the SRR Register. When a software reset is asserted, all the configuration registers including the SRST bit in the SRR Register are reset to their default values. Read/Write transactions can be performed starting at the next valid transaction window.

System Reset

The system reset can be enabled by driving a '1' on the `Bus2IP_Reset` input. All the configuration registers are reset to their default values. Read/Write transactions cannot be performed when the `Bus2IP_Reset` input is '1.'

Exceptions

The contents of the acceptance filter mask registers and acceptance filter ID registers are not cleared when the software reset or system reset is asserted.

Reset Synchronization

A reset synchronizer resets each clock domain in the core. Because of this, some latency exists between the assertion of reset and the actual reset of the core.

Interrupts

The CAN IP core uses a hard-vector interrupt mechanism. It has a single interrupt line (`IP2Bus_IntrEvent`) to indicate an interrupt. Interrupts are indicated by asserting the `IP2Bus_IntrEvent` line (transition of the `IP2Bus_IntrEvent` line from a logic '0' to a logic '1').

Events such as errors on the bus line, message transmission and reception, FIFO overflows and underflow conditions can generate interrupts. During power on, the Interrupt line is driven low.

The Interrupt Status Register (ISR) indicates the interrupt status bits. These bits are set and cleared regardless of the status of the corresponding bit in the Interrupt Enable Register (IER). The IER handles the interrupt-enable functionality. The clearing of a status bit in the ISR is handled by writing a '1' to the corresponding bit in the Interrupt Clear Register (ICR).

The following two conditions cause the `IP2Bus_IntrEvent` line to be asserted:

- If a bit in the ISR is '1' and the corresponding bit in the IER is '1.'
- Changing an IER bit from a '0' to '1;' when the corresponding bit in the ISR is already '1.'

Two conditions cause the `IP2Bus_IntrEvent` line to be deasserted:

- Clearing a bit in the ISR that is '1' (by writing a '1' to the corresponding bit in the ICR); provided the corresponding bit in the IER is '1.'
- Changing an IER bit from '1' to '0'; when the corresponding bit in the ISR is '1'.

When both deassertion and assertion conditions occur simultaneously, the `IP2Bus_IntrEvent` line is deasserted first, and is reasserted if the assertion condition remains true.

Figure 6 is a sample interrupt operation. It illustrates a case where two bits are set in the ISR at the same time and the corresponding bits in the IER are set. The software performs a write to the ICR and clears one of the bits. The interrupt line goes low when the software clears one of the bits in the Interrupt Status Register. However, because another bit is already set in the ISR, in the next clock cycle the interrupt line goes high again. The interrupt line goes low when the software clears the bit that is still '1.'

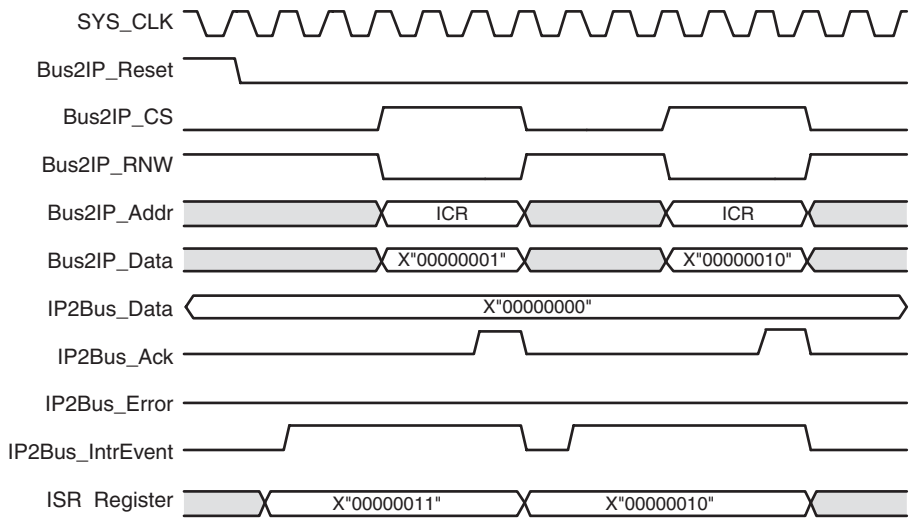

Figure 6: **Sample Interrupt Operation**

Xilinx CAN Controller Design Parameters

To obtain a CAN controller tailored to meet your minimum system requirements, specific features are parameterized. This results in a design using only the required resources, providing the best possible performance. Table 3 shows the CAN controller features that can be parameterized.

Table 3: **CAN Controller Design Parameters**

Feature	Feature Description	Parameter Name	Allowable Values	Default Value
CAN Controller	Depth of the RX FIFO	RX FIFO Depth	2, 4, 8, 16, 32, 64	2
	Depth of the TX FIFO	TX FIFO Depth	2, 4, 8, 16, 32, 64	2
Interface	Number of Acceptance Filters used	Number of Acceptance Filters	0 to 4	0

Xilinx CAN Controller Configuration Register Descriptions

Table 4 defines the CAN controller configuration registers. Each of these registers is 32-bits wide and is represented in big endian format. Since the controller supports 32-bit word access, the `Bus2IP_Addr` is appended with 2'b00 internally. Any read operations to reserved bits or bits that are not used return '0.' A '0' should be written to reserved bits and bit fields not used. Writes to reserved locations are ignored.

Table 4: **CAN Controller Configuration Register**

Register Name	Address	Access
Control Registers		
Software Reset Register (SRR)	0x000	Read/Write
Mode Select Register (MSR)	0x004	Read/Write
Transfer Layer Configuration Registers		
Baud Rate Prescaler Register (BRPR)	0x008	Read/Write
Bit Timing Register (BTR)	0x00C	Read/Write
Error Indication Registers		
Error Counter Register (ECR)	0x010	Read
Error Status Register (ESR)	0x014	Read/Write to Clear
CAN Status Registers		
Status Register (SR)	0x018	Read
Interrupt Registers		
Interrupt Status Register (ISR)	0x01C	Read
Interrupt Enable Register (IER)	0x020	Read/Write
Interrupt Clear Register (ICR)	0x024	Write
Reserved		
Reserved Locations	0x028 to 0x02C	Reads Return 0/ Write has no affect
Messages		

Table 4: **CAN Controller Configuration Register *(Cont'd)***

Transmit Message FIFO (TX FIFO)			
	ID	0x030	Write
	DLC	0x034	Write
	Data Word 1	0x038	Write
	Data Word 2	0x03C	Write
Transmit High Priority Buffer (TX HPB)			
	ID	0x040	Write
	DLC	0x044	Write
	Data Word 1	0x048	Write
	Data Word 2	0x04C	Write
Receive Message FIFO (RX FIFO)			
	ID	0x050	Read
	DLC	0x054	Read
	Data Word 1	0x058	Read
	Data Word 2	0x05C	Read
Acceptance Filtering			
Acceptance Filter Register (AFR)		0x060	Read/Write
Acceptance Filter Mask Register 1 (AFMR1)		0x064	Read/Write
Acceptance Filter ID Register 1 (AFIR1)		0x068	Read/Write
Acceptance Filter Mask Register 2(AFMR2)		0x06C	Read/Write
Acceptance Filter ID Register 2 (AFIR2)		0x070	Read/Write
Acceptance Filter Mask Register 3(AFMR3)		0x074	Read/Write
Acceptance Filter ID Register 3 (AFIR3)		0x078	Read/Write
Acceptance Filter Mask Register 4(AFMR4)		0x07C	Read/Write
Acceptance Filter ID Register 4 (AFIR4)		0x080	Read/Write
Reserved			
Reserved Locations		0x084 to 0x0FC	Reads Return 0/ Write has no affect

Control Registers

Software Reset Register

Writing to the Software Reset Register (SRR) places the CAN controller in Configuration mode. Once in Configuration mode, the CAN controller drives recessive on the bus line and does not transmit or receive messages. During power-up, CEN and SRST bits are '0' and CONFIG bit in the Status Register (SR) is '1.' The Transfer Layer Configuration Registers can be changed only when CEN bit in the SRR Register is '0.'

Use the following steps to configure the CAN controller at power up:

1. Configure the Transfer Layer Configuration Registers (BRPR and BTR) with the values calculated for the particular bit rate.

   See Baud Rate Prescaler Register and Bit Timing Register.
2. Do one of the following:
   - For Loop Back mode, write '1' to LBACK bit in the MSR.
   - For Sleep mode, write '1' to the SLEEP bit in the MSR.
     See Operational CAN Controller Modes for information about operational modes.
3. Set the set the CEN bit in the SRR to 1.

   After the occurrence of 11 consecutive recessive bits, the CAN controller clears the CONFIG bit in the Status Register to '0,' and sets the appropriate Status bit in the Status Register.

Table 5 defines the bit positions in the SR register and Table 6 defines the Software Reset Register bits.

Table 5: **Software Reset Register BIT Positions**

0 — 29	30	31
Reserved	CEN	SRST

Table 6: **Software Reset Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–29	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
30	CEN	Read/Write	0	**Can Enable** The Enable bit for the CAN controller. '1' = The CAN controller is in Loop Back, Sleep or Normal mode depending on the LBACK and SLEEP bits in the MSR. '0' = The CAN controller is in the Configuration mode.
31	SRST	Read/Write	0	**Reset** The Software reset bit for the CAN controller. '1' = CAN controller is reset. If a '1' is written to this bit, all the CAN controller configuration registers (including the SRR) are reset. Reads to this bit always return a '0.'

Mode Select Register

Writing to the Mode Select Register (MSR) enables the CAN controller to enter Sleep, Loop Back, or Normal modes. In Normal mode, the CAN controller participates in normal bus communication. If the SLEEP bit is set to '1,' the CAN controller enters Sleep mode. If the LBACK bit is set to '1,' the CAN controller enters Loop Back mode.

The LBACK and SLEEP bits should never be set to '1' at the same time. At any given point the CAN controller can be either in Loop Back mode or Sleep mode, but not both simultaneously. If both are set, the LBACK Mode takes priority.

Table 7 shows the bit positions in the MSR and Table 8 describes the MSR bits.

Table 7: **Mode Select Register Bit Positions**

0–29	30	31
Reserved	LBACK	SLEEP

Table 8: **Mode Select Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–29	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
30	LBACK	Read/Write	0	**Loop Back Mode Select** The Loop Back Mode Select bit. ‘1’ = CAN controller is in Loop Back mode. ‘0’ = CAN controller is in Normal, Configuration, or Sleep mode. This bit can be written to only when CEN bit in SRR is ‘0.’
31	SLEEP	Read/Write	0	**Sleep Mode Select** The Sleep Mode select bit. ‘1’ = CAN controller is in Sleep mode. ‘0’ = CAN controller is in Normal, Configuration or Loop Back mode. This bit is cleared when the CAN controller wakes up from the Sleep mode.

Transfer Layer Configuration Registers

There are two Transfer Layer Configuration Registers: Baud Rate Prescaler Register (BRPR) and Bit Timing Register (BTR). These registers can be written to only when CEN bit in the SRR is '0.'

Baud Rate Prescaler Register

The CAN clock for the CAN controller is divided by (prescaler+1) to generate the quantum clock needed for sampling and synchronization. Table 9 shows the bit positions in the BRPR, and Table 10 defines the BRPR bits.

Table 9: **Baud Rate Prescaler Register Positions**

0 — 23	24 — 31
Reserved	BPR [7.0]

Table 10: **Baud Rate Prescaler Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–23	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
24–31	BRP[7.0]	Read/Write	0	**Baud Rate Prescaler** These bits indicate the prescaler value. The actual value ranges from 1—256.

The BRPR can be programmed to any value in the range 0–255. The actual value is 1 more than the value written into the register.

The CAN quantum clock can be calculated using the following equation:

```
tq = tosc*(BRP+1)
```

–where tq and tosc are the time periods of the quantum and oscillator/system clocks respectively.

Note: A given CAN bit rate can be achieved with several bit-time configurations, but values should be selected after careful consideration of oscillator tolerances and CAN propagation delays. For more information about CAN bit-time register settings, see the *CAN 2.0A, CAN 2.0B, ISO 11898-1* specifications.

Bit Timing Register

The Bit Timing Register (BTR) specifies the bits needed to configure bit time. Specifically, the Propagation Segment, Phase segment 1, Phase segment 2, and Synchronization Jump Width (as defined in *CAN 2.0A, CAN 2.0B and ISO 11891-1*) are written to the BTR. The actual value of each of these fields is one more than the value written to this register. Table 11 shows the bit positions in the BTR and Table 12 defines the BTR bits.

Table 11: **Bit Timing Register BIT Positions**

0—22	23—24	25—27	28—31
Reserved	SJW[1..0]	TS2[2..0]	TS1[3..0]

Table 12: **Bit Timing Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–22	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
23–24	SJW[1..0]	Read/Write	0	**Synchronization Jump Width** Indicates the Synchronization Jump Width as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.
25–27	TS2[2..0]	Read/Write	0	**Time Segment 2** Indicates Phase Segment 2 as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.
28–31	TS1[3..0]	Read/Write	0	**Time Segment 1** Indicates the Sum of Propagation Segment and Phase Segment 1 as specified in the CAN 2.0A and CAN 2.0B standard. The actual value is one more than the value written to the register.

The following equations can be used to calculate the number of time quanta in bit-time segments:

```
tTSEG1= tq*(8*TSEG1[3]+4*TSEG1[2]+2*TSEG1[1]+TSEG1[0]+1)

tTSEG2= tq*(4*TSEG2[2]+2*TSEG2[1]+TSEG2[0]+1)

tSJW = tq*(2*SJW[1]+SJW[0]+1)
```

–where tTSEG1, tTSEG2 and tSJW are the lengths of TS1, TS2 and SJW.

Note: A given bit-rate can be achieved with several bit-time configurations, but values should be selected after careful consideration of oscillator tolerances and CAN propagation delays. For more information on CAN bit-time register settings, see the *CAN 2.0A, CAN 2.0B*, and *ISO 11898-1* specifications.

Error Indication Registers

The Error Counter Register (ECR) and the Error Status Register (ESR) comprise the Error Indication Registers.

Error Counter Register

The ECR is a read-only register. Writes to the ECR have no effect. The value of the error counters in the register reflect the values of the transmit and receive error counters in the CAN Protocol Engine Module (see Figure 1).

The following conditions reset the Transmit and Receive Error counters:

- When '1' is written to the SRST bit in the SRR
- When '0' is written to the CEN bit in the SRR
- When the CAN controller enters Bus Off state
- During Bus Off recovery when the CAN controller enters Error Active state after 128 occurrences of 11 consecutive recessive bits

When in Bus Off recovery, the Receive Error counter is advanced by 1 when a sequence of 11 consecutive recessive bits is seen.

Table 13 shows the bit positions in the ECR and Table 14 defines the ECR bits.

Table 13: **Error Count Register BIT Positions**

0 —15	16 — 23	24 — 31
Reserved	REC[7..0]	TEC[7..0]

Table 14: **Error Count Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–15	Reserved	Read Only	0	**Reserved** Reserved for future expansion.
16–23	REC[7..0]	Read Only	0	**Receive Error Counter** Indicates the Value of the Receive Error Counter.
24–31	TEC[7..0]	Read Only	0	**Transmit Error Counter** Indicates the Value of the Transmit Error Counter.

Error Status Register

The Error Status Register (ESR) indicates the type of error that has occurred on the bus. If more than one error occurs, all relevant error flag bits are set in this register. The ESR is a write-to-clear register. Writes to this register will not set any bits, but will clear the bits that are set.

Table 15 shows the bit positions in the ESR and Table 16 describes the ESR bits. All the bits in the ESR are cleared when '0' is written to the CEN bit in the SRR.

Table 15: **Error Status Register BIT Positions**

0 — 26	27	28	29	30	31
Reserved	ACKER	BERR	STER	FMER	CRCER

Table 16: **Error Status Register Bits**

Bit(s)	Name	Access	Default Value	Description
0—26	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
27	ACKER	Write to Clear	0	**ACK Error** Indicates an acknowledgement error. '1' = Indicates an acknowledgement error has occurred. '0' = Indicates an acknowledgement error has not occurred on the bus since the last write to this register. If this bit is set, writing a '1' clears it.
28	BERR	Write to Clear	0	**Bit Error** Indicates the received bit is not the same as the transmitted bit during bus communication. '1' = Indicates a bit error has occurred. '0' = Indicates a bit error has not occurred on the bus since the last write to this register. If this bit is set, writing a '1' clears it.
29	STER	Write to Clear	0	**Stuff Error** Indicates an error if there is a stuffing violation. '1' = Indicates a stuff error has occurred. '0' = Indicates a stuff error has not occurred on the bus since the last write to this register. If this bit is set, writing a '1' clears it.
30	FMER	Write to Clear	0	**Form Error** Indicates an error in one of the fixed form fields in the message frame. '1' = Indicates a form error has occurred. '0' = Indicates a form error has not occurred on the bus since the last write to this register. If this bit is set, writing a '1' clears it.
31	CRCER	Write to Clear	0	**CRC Error**[1] Indicates a CRC error has occurred. '1' = Indicates a CRC error has occurred. '0' = Indicates a CRC error has not occurred on the bus since the last write to this register. If this bit is set, writing a '1' clears it.

1. In case of a CRC Error and a CRC delimiter corruption, only the FMER bit is set.

CAN Status Register

The CAN Status Register provides a status of all conditions of the core. Specifically, FIFO status, Error State, Bus State and Configuration mode are reported.

Status Register

Table 17 shows the SR bit positions in the SR and Table 18 provides SR bit descriptions.

Table 17: **Status Register BIT Positions**

0 — 19	20	21	22	23 — 24	25
Reserved	ACFBSY	TXFLL	TXBFLL	ESTAT[1..0]	ERRWRN
26	**27**	**28**	**29**	**30**	**31**
BBSY	BIDLE	NORMAL	SLEEP	LBACK	CONFIG

Table 18: **Status Register Bits**

Bit(s)	Name	Access	Default Value	Description
0—19	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
20	ACFBSY	Read Only	0	**Acceptance Filter Busy** This bit indicates that the Acceptance Filter Mask Registers and the Acceptance Filter ID Registers cannot be written to. ‘1’ = Acceptance Filter Mask Registers and Acceptance Filter ID Registers cannot be written to. ‘0’ = Acceptance Filter Mask Registers and the Acceptance Filter ID Registers can be written to. This bit exists only when the number of acceptance filters is not ‘0’ This bit is set when a ‘0’ is written to any of the valid UAF bits in the Acceptance Filter Register.
21	TXFLL	Read Only	0	**Transmit FIFO Full** Indicates that the TX FIFO is full. ‘1’ = Indicates the TX FIFO is full. ‘0’ = Indicates the TX FIFO is not full.
22	TXBFLL	Read Only	0	**High Priority Transmit Buffer Full** Indicates the High Priority Transmit Buffer is full. ‘1’ = Indicates the High Priority Transmit Buffer is full. ‘0’ = Indicates the High Priority Transmit Buffer is not full.
23–24	ESTAT[1..0]	Read Only	0	**Error Status** Indicates the error status of the CAN controller. “00” = Indicates Configuration Mode (CONFIG = ‘1’). Error State is undefined. “01” = Indicates Error Active State. “11” = Indicates Error Passive State. “10” = Indicates Bus Off State.
25	ERRWRN	Read Only	0	**Error Warning** Indicates that either the Transmit Error counter or the Receive Error counter has exceeded a value of 96. ‘1’ = One or more error counters have a value greater than or equal to 96. ‘0’ = Neither of the error counters has a value greater than or equal to 96.

Table 18: **Status Register Bits *(Cont'd)***

26	BBSY	Read Only	0	**Bus Busy** Indicates the CAN bus status. '1' = Indicates that the CAN controller is either receiving a message or transmitting a message. '0' = Indicates that the CAN controller is either in Configuration mode or the bus is idle.
27	BIDLE	Read Only	0	**Bus Idle** Indicates the CAN bus status. '1' = Indicates no bus communication is taking place. '0' = Indicates the CAN controller is either in Configuration mode or the bus is busy.
28	NORMAL	Read Only	0	**Normal Mode** Indicates the CAN controller is in Normal Mode. '1' = Indicates the CAN controller is in Normal Mode. '0' = Indicates the CAN controller is not in Normal mode.
29	SLEEP	Read Only	0	**Sleep Mode** Indicates the CAN controller is in Sleep mode. '1' = Indicates the CAN controller is in Sleep mode. '0' = Indicates the CAN controller is not in Sleep mode.
30	LBACK	Read Only	0	**Loop Back Mode** Indicates the CAN controller is in Loop Back mode. '1' = Indicates the CAN controller is in Loop Back mode. '0' = Indicates the CAN controller is not in Loop Back mode.
31	CONFIG	Read Only	1	**Configuration Mode Indicator** Indicates the CAN controller is in Configuration mode. '1' = Indicates the CAN controller is in Configuration mode. '0' = Indicates the CAN controller is not in Configuration mode.

Interrupt Registers

The CAN controller contains a single interrupt line, but several interrupt conditions. Interrupts are controlled by the interrupt status, enable, and clear registers.

Interrupt Status Register

The Interrupt Status Register (ISR) contains bits that are set when a particular interrupt condition occurs. If the corresponding mask bit in the Interrupt Enable Register is set, an interrupt is generated.

Interrupt bits in the ISR can be cleared by writing to the Interrupt Clear Register. For all bits in the ISR, a set condition takes priority over the clear condition and the bit continues to remain '1.'

Table 19 shows the bit positions in the ISR and Table 20 describes the ISR bits.

Table 19: **Interrupt Status Register BIT Positions**

0 — 19	20	21	22	23	24	25

Table 19: **Interrupt Status Register BIT Positions *(Cont'd)***

Reserved	WKUP	SLP	BSOFF	ERROR	RXNEMP	RXOFLW
26	**27**	**28**	**29**	**30**	**31**	
RXUFLW	RXOK	TXBFLL	TXFLL	TXOK	ARBLST	

Table 20: **Interrupt Status Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–19	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
20	WKUP	Read Only	0	**Wake up Interrupt** A '1' indicates that the CAN controller entered Normal mode from Sleep Mode. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
21	SLP	Read Only	0	**Sleep Interrupt** A '1' indicates that the CAN controller entered Sleep mode. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
22	BSOFF	Read Only	0	**Bus Off Interrupt** A '1' indicates that the CAN controller entered the Bus Off state. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
23	ERROR	Read Only	0	**Error Interrupt** A '1' indicates that an error occurred during message transmission or reception. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
24	RXNEMP	Read Only	0	**Receive FIFO Not Empty Interrupt** A '1' indicates that the Receive FIFO is not empty. This bit can be cleared only by writing to the ICR.
25	RXOFLW	Read Only	0	**RX FIFO Overflow Interrupt** A '1' indicates that a message has been lost. This condition occurs when a new message is being received and the Receive FIFO is Full. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
26	RXUFLW	Read Only	0	**RX FIFO Underflow Interrupt** A '1' indicates that a read operation was attempted on an empty RX FIFO. This bit can be cleared only by writing to the ICR.

Table 20: **Interrupt Status Register Bits *(Cont'd)***

27	RXOK	Read Only	0	**New Message Received Interrupt** A '1' indicates that a message was received successfully and stored into the RX FIFO. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
28	TXBFLL	Read Only	0	**High Priority Transmit Buffer Full Interrupt** A '1' indicates that the High Priority Transmit Buffer is full. The status of the bit is unaffected if write transactions occur on the High Priority Transmit Buffer when it is already full. This bit can be cleared only by writing to the ICR.
29	TXFLL	Read Only	0	**Transmit FIFO Full Interrupt** A '1' indicates that the TX FIFO is full. The status of the bit is unaffected if write transactions occur on the Transmit FIFO when it is already full. This bit can be cleared only by writing to the Interrupt Clear Register.
30	TXOK[1]	Read Only	0	**Transmission Successful Interrupt** A '1' indicates that a message was transmitted successfully. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.
31	ARBLST	Read Only	0	**Arbitration Lost Interrupt** A '1' indicates that arbitration was lost during message transmission. This bit can be cleared by writing to the ICR. This bit is also cleared when a '0' is written to the CEN bit in the SRR.

1. In Loop Back mode, both TXOK and RXOK bits are set. The RXOK bit is set before the TXOK bit.

Interrupt Enable Register

The Interrupt Enable Register (IER) is used to enable interrupt generation. Table 21 shows the bit positions in the IER and Table 22 describes the IER bits.

Table 21: **Interrupt Enable Register Bit Positions**

0 — 19	20	21	22	23	24	25
Reserved	EWKUP	ESLP	EBSOFF	EERROR	ERXNEMP	ERXOFLW
26	**27**	**28**	**29**	**30**	**31**	
ERXUFLW	ERXOK	ETXBFLL	ETXFLL	ETXOK	EARBLST	

Table 22: **Interrupt Enable Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–19	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
20	EWKUP	Read/Write	0	**Enable Wake up Interrupt** Writes to this bit enable or disable interrupts when the WKUP bit in the ISR is set. '1' = Enable interrupt generation if WKUP bit in ISR is set. '0' = Disable interrupt generation if WKUP bit in ISR is set.
21	ESLP	Read/Write	0	**Enable Sleep Interrupt** Writes to this bit enable or disable interrupts when the SLP bit in the ISR is set. '1' = Enable interrupt generation if SLP bit in ISR is set. '0' = Disable interrupt generation if SLP bit in ISR is set.
22	EBSOFF	Read/Write	0	**Enable Bus OFF Interrupt** Writes to this bit enable or disable interrupts when the BSOFF bit in the ISR is set. '1' = Enable interrupt generation if BSOFF bit in ISR is set. '0' = Disable interrupt generation if BSOFF bit in ISR is set.
23	EERROR	Read/Write	0	**Enable Error Interrupt** Writes to this bit enable or disable interrupts when the ERROR bit in the ISR is set. '1' = Enable interrupt generation if ERROR bit in ISR is set. '0' = Disable interrupt generation if ERROR bit in ISR is set.
24	ERXNEMP	Read/Write	0	**Enable Receive FIFO Not Empty Interrupt** Writes to this bit enable or disable interrupts when the RXNEMP bit in the ISR is set. '1' = Enable interrupt generation if RXNEMP bit in ISR is set. '0' = Disable interrupt generation if RXNEMP bit in ISR is set.
25	ERXOFLW	Read/Write	0	**Enable RX FIFO Overflow Interrupt** Writes to this bit enable or disable interrupts when the RXOFLW bit in the ISR is set. '1' = Enable interrupt generation if RXOFLW bit in ISR is set. '0' = Disable interrupt generation if RXOFLW bit in ISR is set.
26	ERXUFLW	Read/Write	0	**Enable RX FIFO Underflow Interrupt** Writes to this bit enable or disable interrupts when the RXUFLW bit in the ISR is set. '1' = Enable interrupt generation if RXUFLW bit in ISR is set. '0' = Disable interrupt generation if RXUFLW bit in ISR is set.
27	ERXOK	Read/Write	0	**Enable New Message Received Interrupt** Writes to this bit enable or disable interrupts when the RXOK bit in the ISR is set. '1' = Enable interrupt generation if RXOK bit in ISR is set. '0' = Disable interrupt generation if RXOK bit in ISR is set.

Table 22: **Interrupt Enable Register Bits *(Cont'd)***

28	ETXBFLL	Read/Write	0	**Enable High Priority Transmit Buffer Full Interrupt** Writes to this bit enable or disable interrupts when the TXBFLL bit in the ISR is set. '1' = Enable interrupt generation if TXBFLL bit in ISR is set. '0' = Disable interrupt generation if TXBFLL bit in ISR is set.
29	ETXFLL	Read/Write	0	**Enable Transmit FIFO Full Interrupt** Writes to this bit enable or disable interrupts when TXFLL bit in the ISR is set. '1' = Enable interrupt generation if TXFLL bit in ISR is set. '0' = Disable interrupt generation if TXFLL bit in ISR is set.
30	ETXOK	Read/Write	0	**Enable Transmission Successful Interrupt** Writes to this bit enable or disable interrupts when the TXOK bit in the ISR is set. '1' = Enable interrupt generation if TXOK bit in ISR is set. '0' = Disable interrupt generation if TXOK bit in ISR is set.
31	EARBLST	Read/Write	0	**Enable Arbitration Lost Interrupt** Writes to this bit enable or disable interrupts when the ARBLST bit in the ISR is set. '1' = Enable interrupt generation if ARBLST bit in ISR is set. '0' = Disable interrupt generation if ARBLST bit in ISR is set.

Interrupt Clear Register

The Interrupt Clear Register (ICR) is used to clear interrupt status bits.Table 23 shows the bit positions in the ICR and Table 24 describes the ICR bits.

Table 23: **Interrupt Clear Register Bit Positions**

0 — 19	20	21	22	23	24	25
Reserved	CWKUP	CSLP	CBSOFF	CERROR	CRXNEMP	CRXOFLW
26	**27**	**28**	**29**	**30**	**31**	
CRXUFLW	CRXOK	CTXBFLL	CTXFLL	CTXOK	CARBLST	

Table 24: **Interrupt Clear Register Bit Descriptions**

Bit(s)	Name	Access	Default Value	Description
0–19	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
20	CWKUP	Write Only	0	**Clear Wake up Interrupt** Writing a '1' to this bit clears the WKUP bit in the ISR.
21	CSLP	Write Only	0	**Clear Sleep Interrupt** Writing a '1' to this bit clears the SLP bit in the ISR.
22	CBSOFF	Write Only	0	**Clear Bus Off Interrupt** Writing a '1' to this bit clears the BSOFF bit in the ISR.
23	CERROR	Write Only	0	**Clear Error Interrupt** Writing a '1' to this bit clears the ERROR bit in the ISR.

Table 24: **Interrupt Clear Register Bit Descriptions *(Cont'd)***

24	CRXNEMP	Write Only	0	**Clear Receive FIFO Not Empty Interrupt** Writing a '1' to this bit clears the RXNEMP bit in the ISR.
25	CRXOFLW	Write Only	0	**Clear RX FIFO Overflow Interrupt** Writing a '1' to this bit clears the RXOFLW bit in the ISR.
26	CRXUFLW	Write Only	0	**Clear RX FIFO Underflow Interrupt** Writing a '1' to this bit clears the RXUFLW bit in the ISR.
27	CRXOK	Write Only	0	**Clear New Message Received Interrupt** Writing a '1' to this bit clears the RXOK bit in the ISR.
28	CTXBFLL	Write Only	0	**Clear High Priority Transmit Buffer Full Interrupt** Writing a '1' to this bit clears the TXBFLL bit in the ISR.
29	CTXFLL	Write Only	0	**Clear Transmit FIFO Full Interrupt** Writing a '1' to this bit clears the TXFLL bit in the ISR.
30	CTXOK	Write Only	0	**Clear Transmission Successful Interrupt** Writing a '1' to this bit clears the TXOK bit in the ISR.
31	CARBLST	Write Only	0	**Clear Arbitration Lost Interrupt** Writing a '1' to this bit clears the ARBLST bit in the ISR.

Message Storage

The CAN controller has a Receive FIFO (RX FIFO) for storing received messages. The RX FIFO depth is configurable and can store up to 64 messages. Messages that pass any of the acceptance filters are stored in the RX FIFO. When no acceptance filter has been selected, all received messages will be stored in the RX FIFO.

The CAN controller has a configurable Transmit FIFO (TX FIFO) that can store up to 64 messages. The CAN controller also has a High Priority Transmit Buffer (TX HPB), with storage for one message. When a higher priority message needs to be sent, write the message to the High Priority Transmit Buffer. The message in the Transmit Buffer has priority over messages in the TX FIFO.

Message Transmission and Reception

The following rules apply regarding message transmission and reception:

- A message in the TX High Priority Buffer (TX HPB) has priority over messages in the TX FIFO.
- In case of arbitration loss or errors during the transmission of a message, the CAN controller tries to retransmit the message. No subsequent message, even a newer, higher priority message is transmitted until the original message is transmitted without errors or arbitration loss.
- The messages in the TX FIFO, TX HPB and RX FIFO are retained even if the CAN controller enters Bus off state or Configuration mode.

Message Structure

Each message is 16 bytes. Byte ordering for CAN message structure is shown in Tables 25 through 28.

Table 25: **Message Identifier [IDR]**

0 — 10	11	12	13 — 30	31
ID [28..18]	SRR/RTR	IDE	ID[17..0]	RTR

Table 26: **Data Length Code [DLCR]**

0 — 3	4 — 31
DLC [3..0]	Reserved

Table 27: **Data Word 1 [DW1R]**

0 — 7	8 — 15	16 — 23	24 — 31
DB0[7..0]	DB1[7..0]	DB2[7..0]	DB3[7..0]

Table 28: **Data Word 2 [DW2R]**

0 — 7	8 — 15	16 — 23	24 — 31
DB4[7..0]	DB5[7..0]	DB6[7..0]	DB7[7..0]

Reads from RX FIFO

All 16 bytes must be read from the RX FIFO to receive the complete message. The first word read (4 bytes) returns the identifier of the received message (IDR). The second read returns the Data Length Code (DLC) field of the received message (DLCR). The third read returns Data Word 1 (DW1R), and the fourth read returns Data Word 2 (DW2R).

All four words have to be read for each message, even if the message contains less than 8 data bytes. Write transactions to the RX FIFO are ignored. Reads from an empty RX FIFO return invalid data.

Writes to TX FIFO and High Priority TX Buffer

When writing to the TX FIFO or the TX HPB, all 16 bytes must be written. The first word written (4 bytes) is the Identifier (IDR). The second word written is the DLC field (DLCR). The third word written is Data Word 1 (DW1R) and the fourth word written is Data Word 2 (DW2R).

When transmitting on the CAN bus, the CAN controller transmits the data bytes in the following order (DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7). The MSb of a data byte is transmitted first.

All four words must be written for each message, including messages containing fewer than 8 data bytes. Reads transactions from the TX FIFO or the TX High Priority Buffer return 0.

- '0's must be written to unused Data Fields in the DW1R and DW2R registers
- '0's must be written to bits 4 to 31 in the DLCR
- '0's must be written to IDR [13 to 31] for standard frames

Identifier

The Identifier (IDR) word contains the identifier field of the CAN message. Two formats exist for the Identifier field of the CAN message frame:

- **Standard Frames**. Standard frames have an 11-bit identifier field called the Standard Identifier. Only the ID[28..18], SRR/RTR, and IDE bits are valid. ID[28..18] is the 11 bit identifier. The SRR/RTR bit differentiates between data and remote frames. IDE is '0' for standard frames. The other bit fields are not used.
- **Extended Frames**. Extended frames have an 18-bit identifier extension in addition to the Standard Identifier. All bit fields are valid. The RTR bit is used to differentiate between data and remote frames (The SRR/RTR bit and IDE bit are both '1' for all Extended Frames).

Table 29 provides bit descriptions for the Identifier Word. Table 30 describes the DLC Word bits. Table 31 describes the Data Word 1 and Data Word 2 bits.

Table 29: **Identifier Word Bits**

Bit(s)	Name	Access	Default Value	Description
0–10	ID[28..18]	Reads from RX FIFO Writes to TX FIFO and TX HPB	0	**Standard Message ID** The Identifier portion for a Standard Frame is 11 bits. These bits indicate the Standard Frame ID. This field is valid for both Standard and Extended Frames.
11	SRR/RTR	Reads from RX FIFO Writes to TX FIFO and TX HPB	0	**Substitute Remote Transmission Request** This bit differentiates between data frames and remote frames. Valid only for Standard Frames. For Extended frames this bit is 1. ‘1’ = Indicates that the message frame is a Remote Frame. ‘0’ = Indicates that the message frame is a Data Frame.

Table 29: **Identifier Word Bits *(Cont'd)***

12	IDE	Reads from RX FIFO Writes to TX FIFO and TX HPB	0	**Identifier Extension** This bit differentiates between frames using the Standard Identifier and those using the Extended Identifier. Valid for both Standard and Extended Frames. '1' = Indicates the use of an Extended Message Identifier. '0'= Indicates the use of a Standard Message Identifier.
13–30	ID[18..0]	Reads from RX FIFO Writes to TX FIFO and TX HPB	0	**Extended Message ID** This field indicates the Extended Identifier. Valid only for Extended Frames. For Standard Frames, reads from this field return '0's For Standard Frames, writes to this field should be '0's
31	RTR	Reads from RX FIFO Writes to TX FIFO and TX HPB	0	**Remote Transmission Request** This bit differentiates between data frames and remote frames. Valid only for Extended Frames. '1' = Indicates the message object is a Remote Frame '0' = Indicates the message object is a Data Frame For Standard Frames, reads from this bit returns '0' For Standard Frames, writes to this bit should be '0'

Table 30: **DLC Word Bits**

Bit(s)	Name	Access	Default Value	Description
0–3	DLC	Read/Write	0	**Data Length Code** This is the data length portion of the control field of the CAN frame. This indicates the number valid data bytes in Data Word 1 and Data Word 2 registers.
4–31	Reserved	Read/Write		Reads from this field return '0's. Writes to this field should be '0's.

Table 31: **Data Word 1 and Data Word 2 Bits**

Register	Field	Access	Default Value	Description
DW1R [0..7]	DB0[7..0]	Read/Write	0	**Data Byte 0** Reads from this field return invalid data if the message has no data.
DW1R [8..15]	DB1[7..0]	Read/Write	0	**Data Byte 1** Reads from this field return invalid data if the message has only 1 byte of data or fewer.
DW1R [16..23]	DB2[7..0]	Read/Write	0	**Data Byte 2** Reads from this field return invalid data if the message has 2 bytes of data or fewer.

Table 31: **Data Word 1 and Data Word 2 Bits *(Cont'd)***

DW1R [24..31]	DB3[7..0]	Read/Write	0	**Data Byte 3** Reads from this field return invalid data if the message has 3 bytes of data or fewer.
DW2R [0..7]	DB4[7..0]	Read/Write	0	**Data Byte 4** Reads from this field return invalid data if the message has 4 bytes of data or fewer.
DW2R [8..15]	DB5[7..0]	Read/Write	0	**Data Byte 5** Reads from this field return invalid data if the message has 5 bytes of data or fewer.
DW2R [16..23]	DB6[7..0]	Read/Write	0	**Data Byte 6** Reads from this field return invalid data if the message has 6 bytes of data or fewer.
DW2R [24..31]	DB7[7..0]	Read/Write	0	**Data Byte 7** Reads from this field return invalid data if the message has 7 bytes of data or fewer.

Acceptance Filters

The number of acceptance filters is configurable from 0 to 4. The *Number of Acceptance Filters* parameter specifies the number of acceptance filters chosen. Each acceptance filter has an Acceptance Filter Mask Register and an Acceptance Filter ID Register.

Acceptance filtering is performed in the following sequence:

1. The incoming Identifier is masked with the bits in the Acceptance Filter Mask Register.
2. The Acceptance Filter ID Register is also masked with the bits in the Acceptance Filter Mask Register.
3. Both resulting values are compared.
4. If both these values are equal, then the message is stored in the RX FIFO.
5. Acceptance Filtering is processed by each of the defined filters. If the incoming identifier passes through any acceptance filter, then the message is stored in the RX FIFO.

The following rules apply to the Acceptance Filtering Process:

- If no acceptance filters are selected (for example, if all the valid UAF bits in the AFR register are '0's or if the parameter Number of Acceptance Filters = 0), all received messages are stored in the RX FIFO.
- If the number of acceptance filters is greater than or equal to 1, all the Acceptance Filter Mask Register and the Acceptance Filter ID Register locations can be written to and read from. However, the use of these filter pairs for acceptance filtering is governed by the existence of the UAF bits in the AFR register.

Acceptance Filter Register

The Acceptance Filter Register (AFR) defines which acceptance filters to use. Each Acceptance Filter ID Register (AFIR) and Acceptance Filter Mask Register (AFMR) pair is associated with a UAF bit.

When the UAF bit is '1,' the corresponding acceptance filter pair is used for acceptance filtering. When the UAF bit is '0,' the corresponding acceptance filter pair is not used for acceptance filtering. The AFR exists only if the Number of Acceptance Filters parameter is not set to '0.'

To modify an acceptance filter pair in Normal mode, the corresponding UAF bit in this register must be set to '0.' After the acceptance filter is modified, the corresponding UAF bit must be set to '1.'

The following conditions govern the number of UAF bits that can exist in the AFR.

- If the number of acceptance filters is 1:UAF1 bit exists
- If the number of acceptance filters is 2:UAF1 and UAF2 bits exist
- If the number of acceptance filters is 3:UAF1, UAF2 and UAF3 bits exist
- If the number of acceptance filters is 4:UAF1, UAF2, UAF3 and UAF4 bits exist
- UAF bits that do not exist are not written to
- Reads from UAF bits that do not exist return '0's
- If all existing UAF bits are set to '0,' then all received messages are stored in the RX FIFO
- If the UAF bits are changed from a '1' to '0' during reception of a CAN message, the message may not be stored in the RX FIFO.

Table 32 shows the bit positions in the AFR and Table 33 describes the AFR bits.

Table 32: **Acceptance Filter Register Bit Positions**

0 — 27	28	29	30	31
Reserved	UAF4	UAF3	UAF2	UAF1

Table 33: **Acceptance Filter Register Bits**

Bit(s)	Name	Access	Default Value	Description
0–27	Reserved	Read/Write	0	**Reserved** Reserved for future expansion.
28	UAF4	Read/Write	0	**Use Acceptance Filter Number 4** Enables the use of acceptance filter pair 4. ‘1’ = Indicates Acceptance Filter Mask Register 4 and Acceptance Filter ID Register 4 are used for acceptance filtering. ‘0’ = Indicates Acceptance Filter Mask Register 4 and Acceptance Filter ID Register 4 are not used for acceptance filtering.
29	UAF3	Read/Write	0	**Use Acceptance Filter Number 3** Enables the use of acceptance filter pair 3. ‘1’ = Indicates Acceptance Filter Mask Register 3 and Acceptance Filter ID Register 3 are used for acceptance filtering. ‘0’ = Indicates Acceptance Filter Mask Register 3 and Acceptance Filter ID Register 3 are not used for acceptance filtering.

Table 33: **Acceptance Filter Register Bits *(Cont'd)***

30	UAF2	Read/Write	0	**Use Acceptance Filter Number 2** Enables the use of acceptance filter pair 2. '1' = Indicates Acceptance Filter Mask Register 2 and Acceptance Filter ID Register 2 are used for acceptance filtering. '0' = Indicates Acceptance Filter Mask Register 2 and Acceptance Filter ID Register 2 are not used for acceptance filtering.
31	UAF1	Read/Write	0	**Use Acceptance Filter Number 1** Enables the use of acceptance filter pair 1. '1' = Indicates Acceptance Filter Mask Register 1 and Acceptance Filter ID Register 1 are used for acceptance filtering. '0' = Indicates Acceptance Filter Mask Register 1 and Acceptance Filter ID Register 1 are not used for acceptance filtering.

Acceptance Filter Mask Registers

The Acceptance Filter Mask Registers (AFMR) contain mask bits used for acceptance filtering. The incoming message identifier portion of a message frame is compared with the message identifier stored in the acceptance filter ID register. The mask bits define which identifier bits stored in the acceptance filter ID register are compared to the incoming message identifier.

There are at most four AFMRs. These registers are stored in a Block RAM. Asserting a software reset or system reset does not clear register contents. If the number of acceptance filters is greater than or equal to 1, then all the four AFMRs are defined. These registers can be read from and written to. However, filtering operations will only be performed on the number of filters defined by the Number of Acceptance Filters parameter. These registers are written to only when the corresponding UAF bits in the AFR are '0' and ACFBSY bit in the SR is '0.'

The following conditions govern AFMRs:

- If the number of acceptance filters is 1:AFMR 1 is used for acceptance filtering
- If the number of acceptance filters is 2:AFMR 1 and AFMR 2 are used for acceptance filtering
- If the number of acceptance filters is 3:AFMR 1, AFMR 2 and AFMR 3 are used for acceptance filtering
- If the number of acceptance filters is 4:AFMR 1, AFMR 2, AFMR 3 and AFMR 4 are used for acceptance filtering
- **Extended Frames**. All bit fields (AMID [28..18], AMSRR, AMIDE, AMID [17..0] and AMRTR) need to be defined.
- **Standard Frames**. Only AMID [28..18], AMSRR and AMIDE need to be defined. AMID [17..0] and AMRTR should be written as '0.'

Table 34 shows the bit positions in the AFMR and Table 35 describes the AFMR bits.

Table 34: **Acceptance Filter Mask Registers Bit Positions**

0 — 10	11	12	13 — 30	31
AMID[28..18]	AMSRR	AMIDE	AMID[17..0]	AMRTR

Table 35: **Acceptance Filter Mask Bit Descriptions**

Bit(s)	Name	Access	Default Value	Description
0–10	AMID [28..18]	Read/Write	0	**Standard Message ID Mask** These bits are used for masking the Identifier in a Standard Frame. '1'= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. '0' = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
11	AMSRR	Read/Write	0	**Substitute Remote Transmission Request Mask** This bit is used for masking the RTR bit in a Standard Frame. '1'= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. '0' = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
12	AMIDE	Read/Write	0	**Identifier Extension Mask** Used for masking the IDE bit in CAN frames. '1'= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. '0' = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier. If AMIDE = '1' and the AIIDE bit in the corresponding Acceptance ID register is '0', this mask is applicable to only Standard frames. If AMIDE = '1' and the AIIDE bit in the corresponding Acceptance ID register is '1', this mask is applicable to only extended frames. If AMIDE = '0' this mask is applicable to both Standard and Extended frames.
13–30	AMID[17..0]	Read/Write	0	**Extended Message ID Mask** These bits are used for masking the Identifier in an Extended Frame. '1'= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. '0' = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.
31	AMRTR	Read/Write	0	**Remote Transmission Request Mask.** This bit is used for masking the RTR bit in an Extended Frame. '1'= Indicates the corresponding bit in Acceptance Mask ID Register is used when comparing the incoming message identifier. '0' = Indicates the corresponding bit in Acceptance Mask ID Register is not used when comparing the incoming message identifier.

Acceptance Filter ID Registers

The Acceptance Filter ID registers (AFIR) contain Identifier bits, which are used for acceptance filtering. There are at most four Acceptance Filter ID Registers. These registers are stored in a Block RAM. Asserting a software reset or system reset will not clear the contents of these registers. If the number of acceptance filters is greater than or equal to 1, then all four AFIRs are defined. These registers can be read from and written to. These registers should be written to only when the corresponding UAF bits in the AFR are '0' and ACFBSY bit in the SR is '0.'

The following conditions govern the use of the AFIRs:

- If the number of acceptance filters is 1: AFIR 1 is used for acceptance filtering
- If the number of acceptance filters is 2: AFIR 1 and AFIR 2 are used for acceptance filtering
- If the number of acceptance filters is 3: AFIR 1, AFIR 2 and AFIR 3 are used for acceptance filtering
- If the number of acceptance filters is 4: AFIR 1, AFIR 2, AFIR 3 and AFIR 4 are used for acceptance filtering
- **Extended Frames**. All the bit fields (AIID [28..18], AISRR, AIIDE, AIID [17..0] and AIRTR) must be defined
- **Standard Frames**. Only AIID [28..18], AISRR and AIIDE need to be defined. AIID [17..0] and AIRTR should be written with '0'

Table 36 shows AFIR bit positions, and Table 37 describes the AFIR bits.

Table 36: **Acceptance Filter ID Registers Bit Positions**

0 — 10	11	12	13 — 30	31
AIID[28..18]	AISRR	AIIDE	AIID[17..0]	AIRTR

Table 37: **Acceptance Filter ID Registers Bits**

Bit(s)	Name	Access	Default Value	Description
0–10	AIID [28..18]	Read/Write	0	**Standard Message ID** Standard Identifier
11	AISRR	Read/Write	0	**Substitute Remote Transmission Request** Indicates the Remote Transmission Request bit for Standard frames
12	AIIDE	Read/Write	0	**Identifier Extension** Differentiates between Standard and Extended frames
13–30	AIID[17..0]	Read/Write	0	**Extended Message ID Mask** Extended Identifier
31	AIRTR	Read/Write	0	**Remote Transmission Request Mask** RTR bit for Extended frames.

Configuring the CAN Controller

This section covers the various configuration steps that must be performed to program the CAN core for operation.

The following key configuration steps are detailed in this section.

1. Choosing the operation mode
2. Programming the configuration registers to initialize the core
3. Writing messages to the TX FIFO/ TX HPB
4. Reading messages from the RX FIFO

Programming the Configuration Registers

The following are steps to configure the core when the core is powered on or after system or software reset.

1. Choose the operation mode
   - For Loop Back mode, write a '1' to the LBACK bit in the MSR and '0' to the SLEEP bit in the MSR.
   - For Sleep mode, write a '1' to the SLEEP bit in the MSR and '0' to the LBACK bit in the MSR.
   - For Normal Mode, write '0's to the LBACK and SLEEP bits in the MSR.
2. Configure the Transfer Layer Configuration Registers
   - Program the Baud Rate Prescaler Register and the Bit Timing Register to correspond to the network timing parameters and the network characteristics of the system.
3. Configure the Acceptance Filter Registers

   The number of Acceptance Filter Mask and Acceptance Filter ID Register pairs is chosen at build time. To configure these registers do the following:
   - Write a '0' to the UAF bit in the AFR register corresponding to the Acceptance Filter Mask and ID Register pair to be configured.
   - Wait until the ACFBSY bit in the SR is '0.'
   - Write the appropriate mask information to the Acceptance Filter Mask Register.
   - Write the appropriate ID information to the to the Acceptance Filter ID Register.
   - Write a '1' to the UAF bit corresponding to the Acceptance Filter Mask and ID Register pair.
   - Repeat the preceding steps for each Acceptance Filter Mask and ID Register pair.
4. Write to the Interrupt Enable Register to choose the bits in the Interrupt Status Register that can generate an interrupt.
5. Enable the CAN controller by writing a '1' to the CEN bit in the SRR register.

Transmitting a Message

A message to be transmitted can be written to either the TX FIFO or the TX HPB. A message in the TX HPB gets priority over the messages in the TX FIFO. The TXOK bit in the ISR is set after the CAN core successfully transmits a message.

Writing a Message to the TX FIFO

All messages written to the TX FIFO should follow the format defined in "Message Storage" on page 26.

To perform a write:

1. Poll the TXFLL bit in the SR. The message can be written into the TX FIFO when the TXFLL bit is '0.'
2. Write the ID of the message to the TX FIFO ID memory location (0x030).
3. Write the DLC of the message to the TX FIFO DLC memory location (0x034).
4. Write Data Word 1 of the message to the TX FIFO DW1 memory location (0x038).
5. Write Data Word 2 of the message to the TX FIFO DW2 memory location (0x03C).

Messages can be continuously written to the TX FIFO until the TX FIFO is full. When the TX FIFO is full the TXFLL bit in the ISR and the TXFLL bit in the SR are set. If polling, the TXFLL bit in the Status Register should be polled after each write. If using interrupt mode, writes can continue until the TXFLL bit in the ISR generates an interrupt.

Writing a Message to the TX HPB

All messages written to the TX FIFO should follow the format described in "Message Storage" on page 26.

To write a message to the TX HPB:

1. Poll the TXBFLL bit in the SR.

   The message can be written into the TX HPB when the TXBFLL bit is '0.'
2. Write the ID of the message to the TX HPB ID memory location (0x040).
3. Write the DLC of the message to the TX HPB DLC memory location (0x044).
4. Write Data Word 1 of the message to the TX HPB DW1 memory location (0x048).
5. Write Data Word 2 of the message to the TX HPB DW2 memory location (0x04C).

After each write to the TX HPB, the TXBFLL bit in the Status Register and the TXBFLL bit in the Interrupt Status Register are set.

Receiving a Message

Whenever a new message is received and written into the RX FIFO, the RXNEMP bit and the RXOK bits in the ISR are set. In case of a read operation on an empty RX FIFO, the RXUFLW bit in the ISR is set.

Reading a Message from the RX FIFO

Perform the following steps to read a message from the RX FIFO.

1. Poll the RXOK or RXNEMP bits in the ISR. In interrupt mode, the reads can occur after the RXOK or RXNEMP bits in the ISR generate an interrupt.
   - Read from the RX FIFO memory locations. All the locations must be read regardless of the number of data bytes in the message.
   - Read from the RX FIFO ID location (+ 0x050)
   - Read from the RX FIFO DLC location (+ 0x054)
   - Read from the RX FIFO DW1 location (+ 0x058)
   - Read from the RX FIFO DW2 location (+ 0x05C)
2. After performing the read, if there are one or more messages in the RX FIFO, the RXNEMP bit in the ISR is set. This bit can either be polled or can generate an interrupt.
3. Repeat until the FIFO is empty.

CAN Graphical User Interface

The CAN graphical user interface (GUI) provides a single screen for configuring the CAN core.

Main Screen

Figure 7 shows the main CAN customization screen, which you use to set the component name and core options, described in the following sections

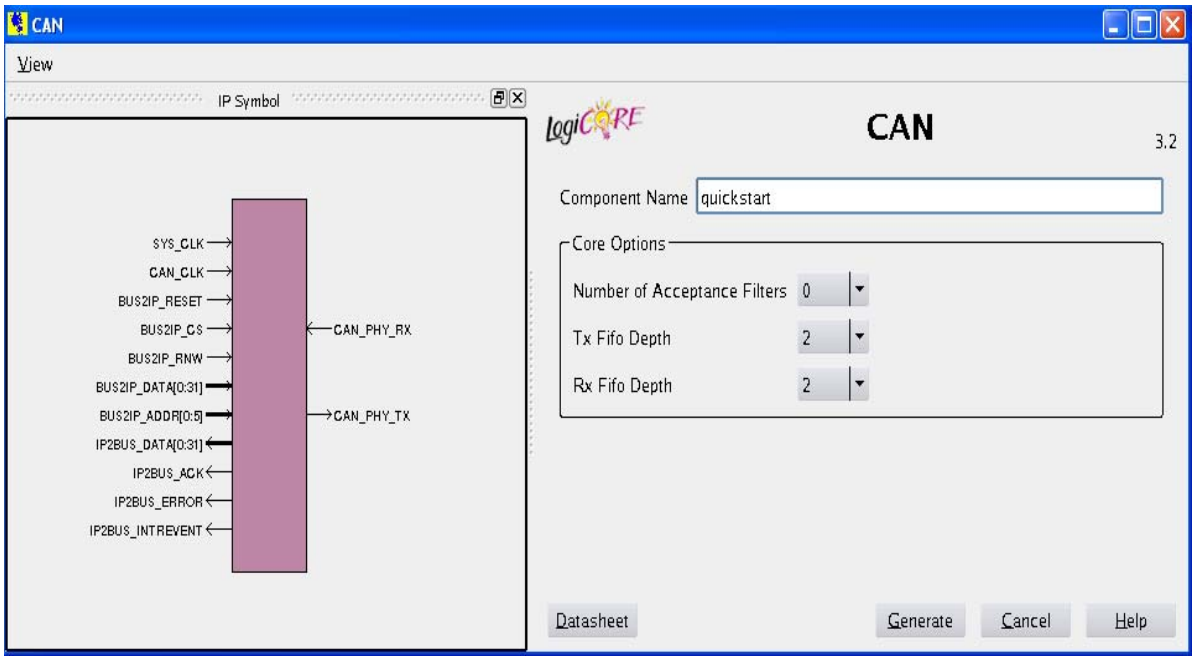

Figure 7: **Main Screen**

Component Name

The Component Name is the base name of the output files generated for this core. The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9 and "-."

Core Options

Number of Acceptance Filters

This specifies the number of acceptance filter pairs used by the CAN controller. Each acceptance filter pair consists of a Mask Register and an ID register. These registers can be configured so that a specific identifier or a range of identifiers can be received. Valid range is from 0 to 4.

TX FIFO Depth

The TX FIFO depth is measured in terms of the number of CAN messages. For example, TX FIFO with a depth of 2 can hold at most 2 CAN messages.

Valid values are 2, 4, 8, 16, 32, 64 to configure the depth of the TX FIFO.

RX FIFO Depth

The RX FIFO depth is measured in terms of the number of CAN messages. For example, RX FIFO with a depth of 2 can hold at most 2 CAN messages.

Valid values are 2, 4, 8, 16, 32, 64 to configure the depth of the RX FIFO.

Ordering Information

A free evaluation version of the CAN core is provided with the Xilinx® CORE Generator™ software, which lets you assess the core functionality and demonstrates the various interfaces of the core in simulation. After purchase, the core may be downloaded from the **Xilinx IP Center** for use with the CORE Generator software v12.1 and higher. The CORE Generator software is bundled with ISE Design Suite v11.3 software at no additional charge.

Please contact your local **Xilinx sales representative** or for pricing and availability about the CAN module or go to the **CAN product page** for additional information.

Extra Design Consideration

The CAN and XPS CAN cores require an input register on the RX line to avoid a potential error condition where multiple registers receive different values resulting in error frames. This error condition is rare; however, the work-around should be implemented in all cases.

To work around this issue, insert a register on the RX line clocked by CAN_CLK with an initial value of '1'. This applies to all versions of the CAN and XPS CAN cores.

Constraining the CAN Controller

Device and Package Selection

The CAN controller can be implemented in XA Spartan-3, XA Spartan-3E, XA Spartan-3A, XA Spartan-3A DSP, Spartan-3, Spartan-3E, Spartan-3A/AN/3A DSP, XA Spartan-6, Spartan-6, Virtex-4, Virtex-5, and Virtex-6 devices. Ensure that the device used has the following attributes:

- The device is large enough to accommodate the core.
- The device contains a sufficient number of IOBs.

Location Constraints

No specific I/O location constraints.

Placement Constraints

No specific placement constraints.

Timing Constraints

The core has two different clock domains: SYS_CLK and CAN_CLK. The following constraints can be used with the CAN Controller.

PERIOD Constraints for Clock Nets

CAN_CLK

The clock provided to CAN_CLK must be constrained for a clock frequency of less than or equal to 24 MHz, based on the input oscillator frequency.

```
# Set the CAN_CLK constraints
NET "CAN_CLK" TNM_NET = "CAN_CLK";
TIMESPEC "TS_CAN_CLK" = PERIOD "CAN_CLK" 40 ns HIGH 50%;
```

SYS_CLK

The clock provided to SYS_CLK must be constrained for a clock frequency of 100 MHz or less.

```
# Set the SYS_CLK constraints
# This can be relaxed based on the actual frequency
NET "SYS_CLK" TNM_NET = "SYS_CLK";
TIMESPEC "TS_SYS_CLK" = PERIOD "SYS_CLK" 10 ns HIGH 50%;
```

Timing Ignore Constraints

A timing ignore (TIG) constraint must be specified on all the signals that cross clock domains.

```
# Timing Ignore constraint on all signals that cross from CAN_CLK domain to SYS_CLK domain
TIMESPEC "TS_CAN_SYS_TIG" = FROM "CAN_CLK" TO "SYS_CLK"  TIG;
# Timing Ignore constraint on all signals that cross from SYS_CLK domain to CAN_CLK domain
TIMESPEC "TS_SYS_CAN_TIG" = FROM "SYS_CLK" TO "CAN_CLK" TIG;
```

I/O Constraints

The following constraints ensure that the flops associated with the external I/O signals are placed in IOBs.

```
# The following constraints need to be given if the CAN controller is used in a Standalone mode.
INST "Bus2IP_Data*"IOB = true;
INST "Bus2IP_Addr*"IOB = true;
INST "Ip2Bus_Data*"IOB = true;
INST "Ip2Bus_Ack"IOB = true;
INST "Bus2IP_RNW"IOB = true;
INST "Ip2Bus_IntrEvent"IOB = true;
INST "Bus2IP_CS"IOB = true;
INST "BUS2IP_ResetIOB = true;
INST "Ip2BUS_ErrorIOB = true;
```

IO Standards

The pins that interface to the CAN PHY chip have a 3.3 volt signal level interface. The following constraints may be used provided the device IO Banking rules are followed:

```
# Select the I/O standards for the interface to the CAN PHY
INST "CAN_PHY_TX"            IOSTANDARD = "LVTTL"
INST "CAN_PHY_RX"            IOSTANDARD = "LVTTL"
```

References

1. ISO 11898-1: Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed Communication.
2. Controller Area Network (CAN) version 2.0A and B Specification, Robert Bosch GmbH.
3. OPB IPIF (v3.01.a) Product Specification.

Revision History

Date	Version	Revision Notes
08/31/05	1.0	Initial Xilinx Release
11/10/05	1.1	Minor release updates, version number advanced, release date updated.
1/18/06	1.2	Minor release updates, version number advanced, release date updated.
9/21/06	1.3	Core version updated to 1.4, support for Spartan-3A device added, correction to add Virtex-II Pro device to Facts table.
8/08/07	1.4	Updated for the IP1 Jade Minor release. Added support for Spartan 3AN/3A DSP, Virtex-5 device families, corrected number of IOs in LogiCORE Facts Table.
4/24/09	1.5	Updated for the IP1 Lava release. Added support for XA Spartan-3A, XA Spartan-3A DSP, and Spartan-6 device families.
6/24/09	1.6	Minor documentation updates.
9/16/09	1.7	Core version updated to 3.1; Xilinx tools updated to 11.3. Added support for Virtex-6 Device Family.
4/19/10	1.8	Core version updated to 3.2; Xilinx tools updated to 12.1.

Notice of Disclaimer