Subsystem Design

SPI to CAN Bridge on MSPM0 MCUs

Kalicheti Vishnuvardhan Reddy, Ashwini Gopinath, Shubham Saurabh

Table of Contents

Trademarks

1 Introduction

Controller Area Network (CAN) and Serial Peripheral Interface (SPI) are two widely adopted communication protocols in modern microcontroller units (MCUs). While CAN protocol dominates automotive applications due to the robust error handling and reliability, many low-end microcontrollers and sensors lack CAN support. In contrast, SPI is nearly universal across MCU devices. To bridge this technological gap, SPI-CAN interfaces enable seamless communication between SPI-based devices and CAN networks, allowing low-cost sensors and microcontrollers to integrate with CAN-based systems.

The SPI-CAN bridge functionality is implemented using the MSPM0G3507 device, which acts as an interface between an external SPI controller and a CAN network. When the external SPI controller initiates communication frames, the MSPM0G3507 translates these into CAN protocol instructions and transmits them across the CAN network.

Figure 1-1 illustrates the system architecture, showing the interconnections between the MSPM0G3507 device and the external interfaces. The device communicates with the external controller using 4-wire SPI protocol, while connecting to the CAN bus through a CAN physical layer transceiver (CAN-PHY).

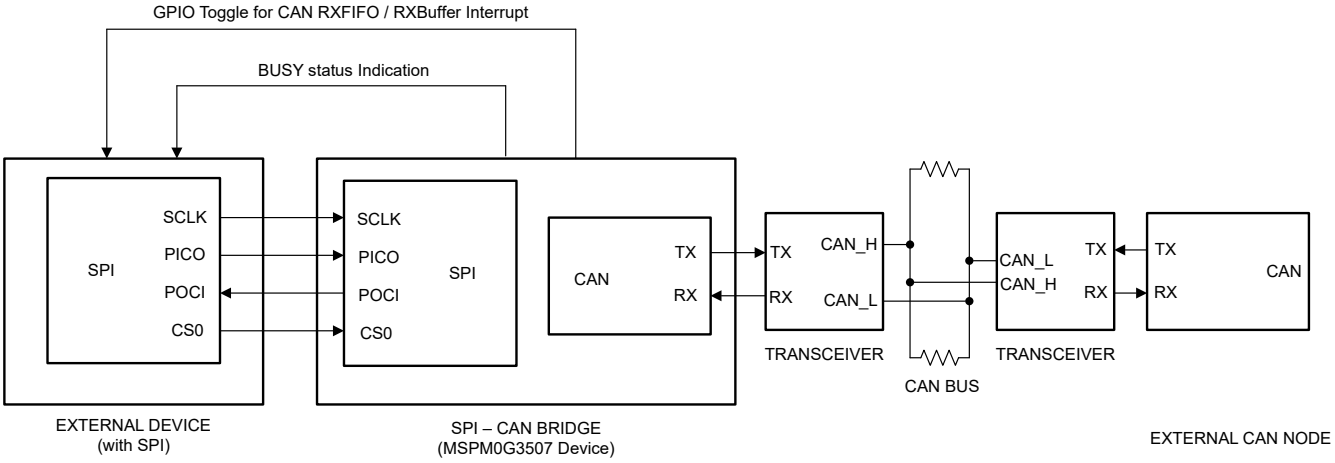

Figure 1-1. Block Diagram of SPI-CAN Bridge

1.1 Features Supported

CAN FD support

- Up to 1MbpS nominal bit rate, 5MbpS data bit rate
- Dedicated receive and transmit buffers (4 on each side)
- Transmit and Receive FIFO (each with depth of 4)
- Extended and Standard Filter elements (8 each on transmit & receive side)

SPI

- Frequency support up to 8MHz
- Implemented for 4-wire Motorola Phase0 Polarity0 mode

Error indications

- Instruction Error and Timeout status indication

SPI to CAN bridge supports Classical CAN and CAN-FD formats. Differences between CAN and CAN-FD frame formats are detailed in Table 1-1.

Table 1-1. CAN and CAN-FD Differences

Feature	CAN 2.0	CAN 2.0B	CAN FD
Identifier	Standard (11 bits)	Standard (11 bits) Extended (29 bits)	Standard (11 bits) Extended (29 bits)
Data Rate	Up to 1MbpS	Up to 1MbpS	Up to 8MbpS
Payload	0-8 bytes	0-8 bytes	0-64 bytes

1.2 CAN Frame Format

CAN-FD frame formats are shown in Figure 1-2, Figure 1-3, Figure 1-4; bit value of *1* is regarded as recessive and a value of *0* is regarded as dominant.

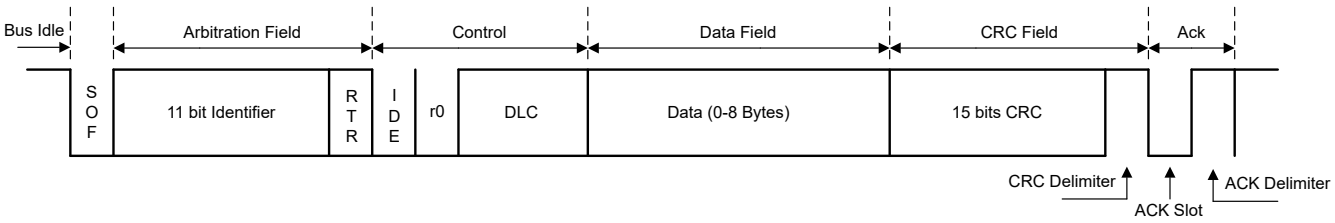

Figure 1-2. CAN 2.0A Frame Format

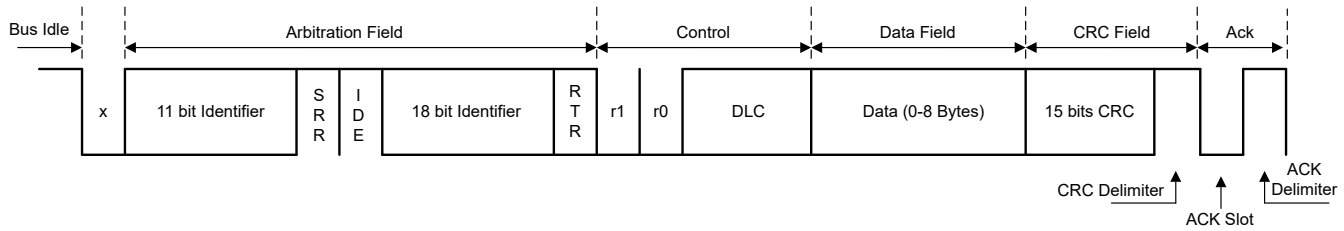

Figure 1-3. CAN 2.0B Frame Format

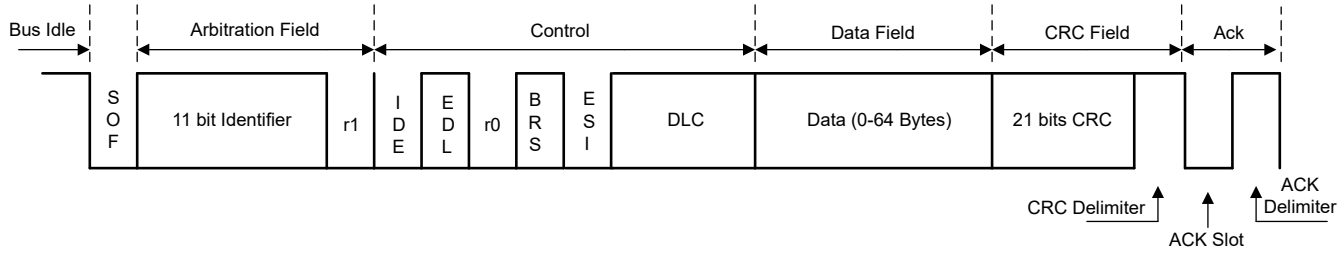

Figure 1-4. CAN-FD Frame Format

Table 1-2. CAN-FD Frame Bits and Definition

Bit	Description
SOF	Start of frame
IDE	Identifier extension
FDF	FD format indicator
BRS	Bit rate switching
ESI	Error status indicator
DLC	Data length code
CRC	Cyclic redundancy check

- IDE = recessive, indicates extended (29 bit) identifier; IDE = dominant, indicates standard (11 bit) identifier.
- FDF = recessive, indicates CAN-FD format; FDF=dominant, indicates CAN frame
- BRS=recessive, indicates bit-rate switching to a faster rate; BRS=dominant, indicates no change in rate

1.3 SPI Message Frame Format

SPI is a synchronous communication protocol with four lines – optional chip select (CS) optional, clock (SCLK), peripheral output controller input (POCI) and peripheral input and controller output (PICO). SPI-CAN bridge supports SPI data transfers up to 8MHz. A sample SPI frame is shown in Figure 1-5.

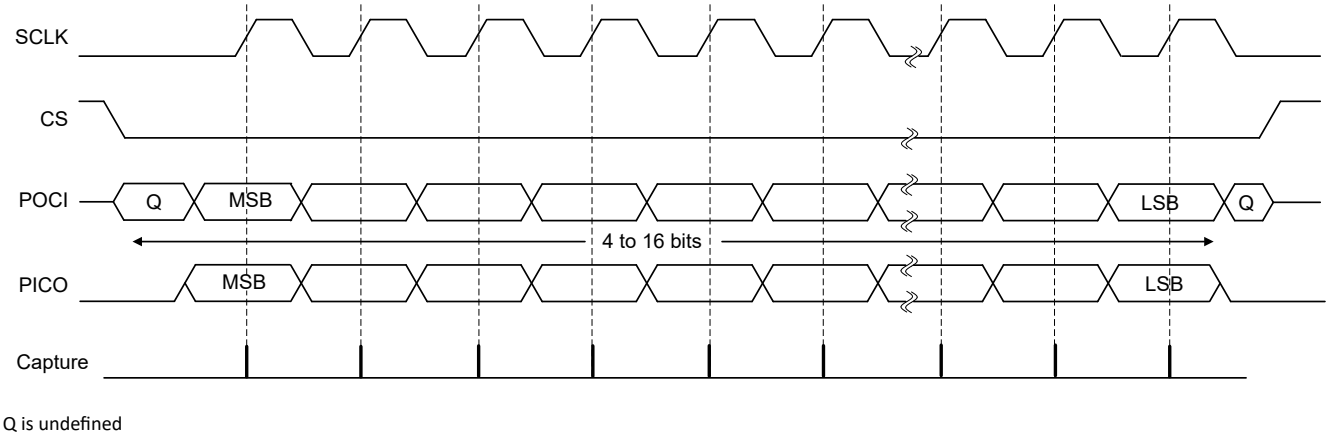

Figure 1-5. Motorola SPI Frame Format With SPO = 0 and SPH = 0

2 Implementation

The SPI-CAN bridge application is configured in two stages:

1. The MSPM0G3507 is set up with SPI operating in peripheral mode and CAN in transaction initiation mode.
2. CAN parameters and settings are configured remotely through SPI commands from the external master controller.

2.1 SPI Message Format

A sample SPI message frame is shown in Figure 2-1. Each message frame is split into:

- Instruction: lower 8 bits of 16-bit frame is used.
- Address offset: 16 bits (only lower 16 bits of address needs to be sent). Refer MSPM0G3507 Technical Reference Manual for register description and address.
- Data mask: 32 bits – two 16-bit frames with LSB first.
- Data: 32 bits – two 16-bit frames with LSB first.

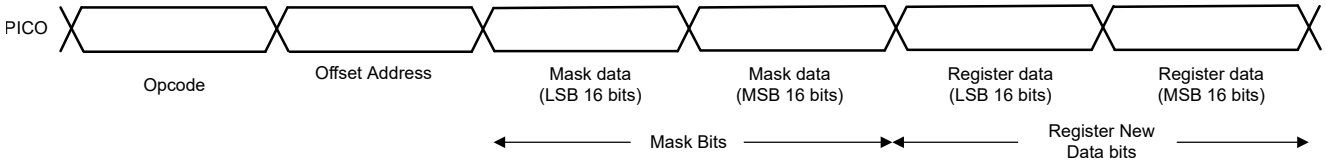

Figure 2-1. SPI Message Format

2.1.1 SPI Commands

Commands supported for configuring CAN module is listed in Table 2-1.

Table 2-1. SPI Commands

Instruction	Opcode	Offset Address	Data Size	Mask Data LSB	Mask Data MSB	Data LSB	Data MSB	Details
Reset	16'h0020	Not used	Not used	Not used	Not used	Not used	Not used	Resets CAN module
Power Enable and CAN configuration	16'h0030	Not used	Not used	Not used	Not used	Not used	Not used	Power enable CAN module and Configuring CAN Message RAM
Write Register	16'h0080	Register offset from CAN base address	Not used	Not used	Not used	LSB 16 bits	MSB 16 bits	Write data into CAN Register
Read Register	16'h0090	Register offset from CAN base address	Not used	Not used	Not used	Not used	Not used	Read data from CAN Register
Bit Modify Register	16'h0010	Register offset from CAN base address	Not used	Mask bits for LSB Data	Mask bits for MSB Data	LSB 16 bits	MSB 16 bits	Modify individual bits in CAN Register
Load transmit buffer	16'h0040	Not used	Data size in bytes	Not used	Not used	Not used	Not used	Write CAN frame into CAN TX Buffer
Load transmit FIFO	16'h0050	Not used	Data size in bytes	Not used	Not used	Not used	Not used	Write CAN frame into CAN TX FIFO

Table 2-1. SPI Commands (continued)

Instruction	Opcode	Offset Address	Data Size	Mask Data LSB	Mask Data MSB	Data LSB	Data MSB	Details
Request to send transmit buffer	16'h0060	Not used	Not used	Not used	Not used	Not used	Not used	Buffer Add Request to TX Buffer element
Request to send transmit FIFO	16'h0070	Not used	Not used	Not used	Not used	Not used	Not used	Buffer Add Request to TXFIFO element
Read receive FIFO	16'h00B0	Not used	Data size in bytes	Not used	Not used	Not used	Not used	Read data from RXFIFO
Read receive buffer	16'h00D0	Not used	Data size in bytes	Not used	Not used	Not used	Not used	Read data from RXBUFFER
Read status register	16'h00F0	Not used	Data size in bytes	Not used	Not used	Not used	Not used	Read Receive status Bits

Read register, Read receive FIFO, Read receive Buffer and Read Status Register instructions must be followed by Fetch Receive Data instruction.

2.1.2 Instruction Set

Bit Modify Instruction

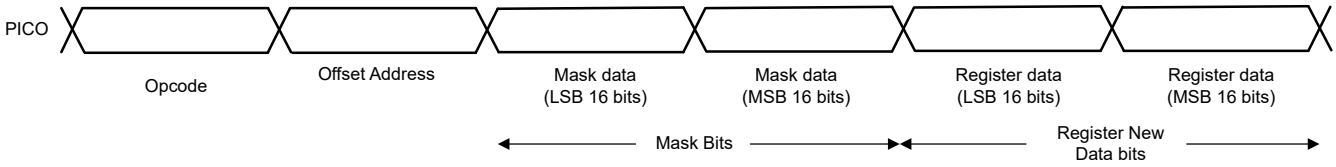

Figure 2-2. Bit Modify Instruction Format

Opcode: 0b0000000000010000

The Bit modify Instruction is used to set or clear individual bits in a specific CAN register. The mask bits determine which bits in the register can be changed. A bit 1 in the mask allows the corresponding bit in the register to change, while a bit 0 prevents the bit from changing.

Reset Instruction

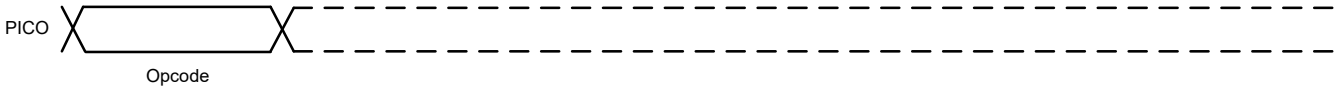

Figure 2-3. Reset Instruction Format

Opcode: 0b0000000000100000

The Reset instruction resets the CAN module in the SPI – CAN Bridge device. To reset the CAN module, the user needs to send the corresponding opcode through the SPI protocol.

Power Enable and CAN Configuration Instruction:

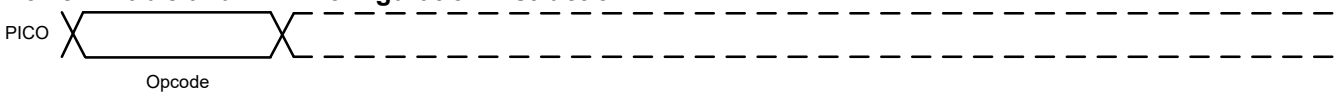

Figure 2-4. Power Enable and CAN Configuration Instruction Format

Opcode: 0b0000000000110000

The Power Enable and CAN Configuration Instruction is used to Power enable the CAN module in the Bridge device. Also, the predefined CAN message RAM configurations are done once CAN module is enabled.

Load TX Buffer Instruction:

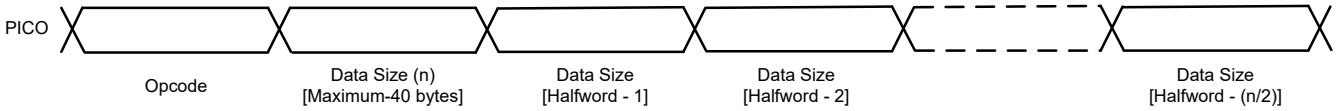

Figure 2-5. Load TX Buffer Instruction Format

Opcode: 0b00000000010000ab

The Load TX Buffer instruction is used to load the CAN frame into the selected Transmit buffers. The transmit buffer selection is encoded in the opcode. One of the four transmit buffer can be selected based on the a, b bits in the instruction field.

Table 2-2. Load TX Buffer mapping

a	b	Buffer
0	0	Buffer – 0
0	1	Buffer – 1
1	0	Buffer – 2
1	1	Buffer – 3

Load TX FIFO Instruction:

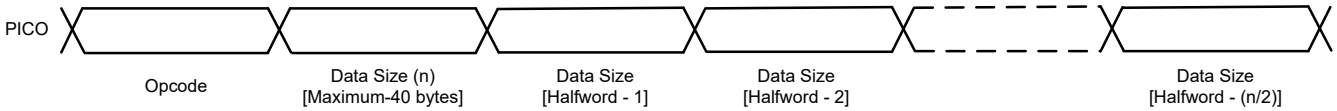

Figure 2-6. Load TX FIFO Instruction

Opcode: 0b0000000001010000

The Load TX FIFO instruction is used to load the CAN frame to be transmitted into the Transmit FIFO. User must take care that the transmit FIFO is not full before loading CAN frame into the Transmit FIFO.

Request to Send TX Buffer Instruction

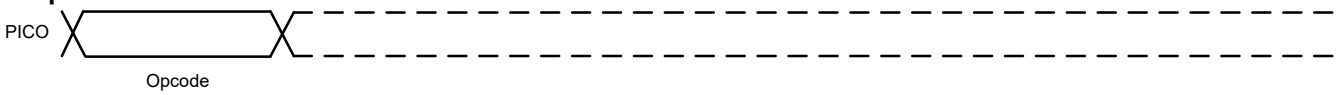

Figure 2-7. Request to send TX Buffer Instruction format

Opcode: 0b00000000011000ab

The Request to send TX Buffer Instruction adds the transmission request for one of the four transmit buffers. The transmit buffer selection is encoded in the opcode. One of the four transmit buffer can be selected based on the a, b bits in the instruction field.

Table 2-3. Request to send TX Buffer mapping

a	b	Buffer
0	0	Buffer – 0
0	1	Buffer – 1
1	0	Buffer – 2
1	1	Buffer – 3

Request to send TXFIFO Instruction

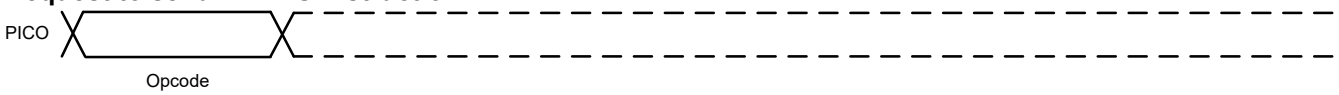

Figure 2-8. Request to Send TXFIFO Instruction Format

Opcode: 0b0000000001110000

The request to send TX FIFO instruction adds the transmission request for one of the transmit FIFO elements based on the Transmit FIFO put index. The user needs to make sure that the Transmit FIFO has a CAN Frame before sending buffer add request.

Write Register Instruction

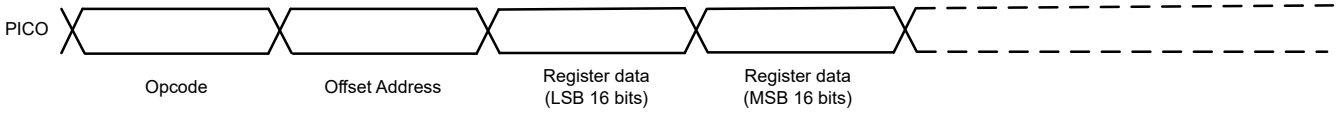

Figure 2-9. Write Register Instruction Format

Opcode: 0b0000000010000000

Offset Address – To point to the register address to write the register data.

The Write Register instruction is used to configure the CAN Registers through the SPI Peripheral. The data is written into the corresponding register based on the Offset address.

Read Register Data Instruction:

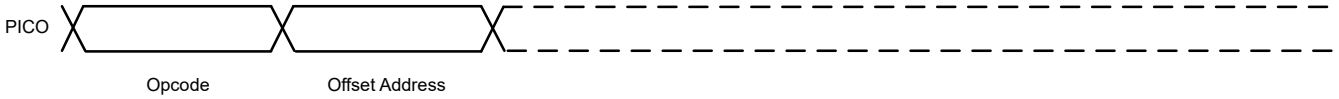

Figure 2-10. Read Register Data Instruction Format

Opcode: 0b0000000010000000

Offset Address – To point to the register address to read the Register Data.

Read register data Instruction is used to read the data from the specific CAN register. The Read register data instruction only reads the register but does not transmit. Hence, each Read register data instruction must be followed by a Fetch register data instruction.

Fetch Register Data Instruction:

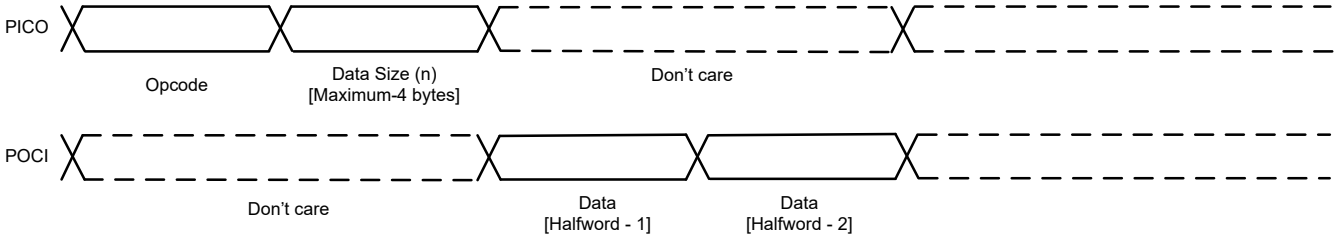

Figure 2-11. Fetch Register Data Instruction format

Opcode: 0b0000000010100000

The Fetch register data instruction transmits the register value that was read using Read register data instruction.

Read RX Buffer Data Instruction:

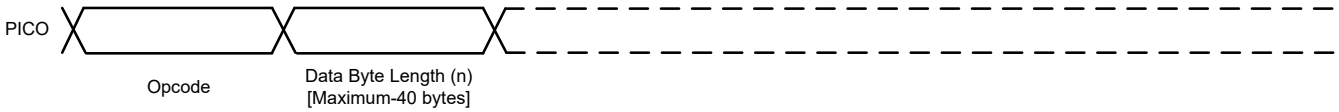

Figure 2-12. Read RX Buffer Data Instruction Format

Opcode: 0b00000000101100xy

When the SPI-CAN bridge receives a new CAN frame from the CAN network, the SPI-CAN bridge toggles the GPIO pin as shown in Figure 2-12 to indicate the reception of the frame to the SPI controller.

To retrieve received CAN frames from the CAN module RX Buffer in the SPI-CAN bridge, the following sequence must be followed:

1. First, use the Read RX status data instruction followed by Fetch RX status data instruction to identify which RX Buffer contains the received CAN frame.
2. Then, execute the Read RX buffer instruction to access the buffer contents. Note that this instruction only reads the buffer location but does not transfer the data.
3. Finally, execute the Fetch RX Buffer data instruction to actually retrieve the data from the buffer.

Each Read RX buffer instruction must be paired with a corresponding Fetch RX Buffer data instruction to complete the data transfer.

The RX buffer selection is encoded in the opcode. One of the four RX buffer can be selected based on the x, y bits in the instruction.

Table 2-4. Read RX Buffer mapping

X	y	Buffer
0	0	Buffer – 0
0	1	Buffer – 1
1	0	Buffer – 2
1	1	Buffer – 3

Fetch RX Buffer Data Instruction

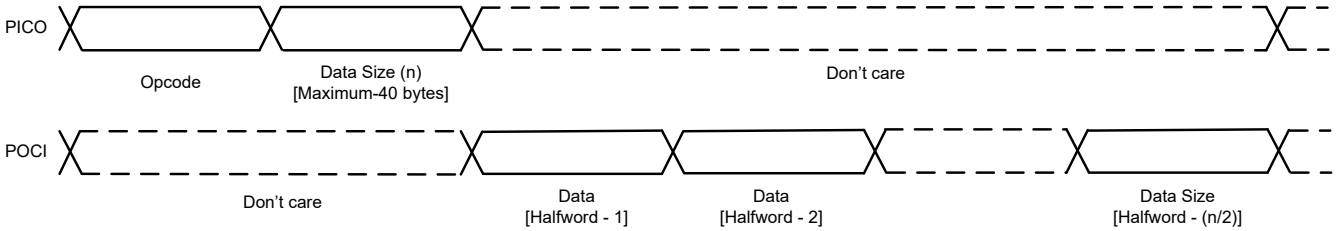

Figure 2-13. Fetch RX Buffer Data Instruction format

Opcode: 0b0000000011000000

The Fetch RX buffer Data Instruction transmits the RX buffer data that was read using Read RX buffer instruction.

Read RX FIFO Data Instruction:

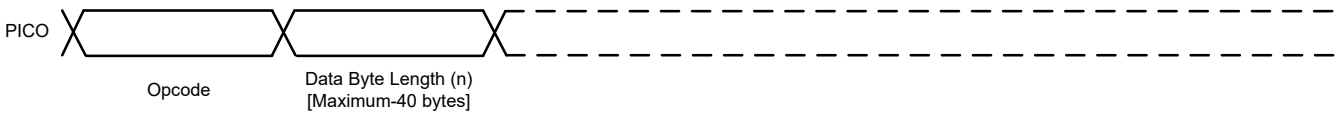

Figure 2-14. Read RXFIFO Data Instruction Format

Opcode: 0b0000000011010000

When the SPI-CAN bridge receives a new CAN frame from the CAN network, the SPI-CAN bridge toggles the GPIO pin as shown in Figure 2-14 to indicate the reception of the frame to the SPI controller.

To retrieve received CAN frames from the CAN module's RX FIFO in the SPI-CAN bridge, the following sequence must be followed:

1. First, use the Read RX status data instruction followed by Fetch RX status data instruction to get the information about the RXFIFO state.
2. Then, execute the Read RX FIFO data instruction to access the fifo element contents. Based on the RX FIFO get index, Rx FIFO element is selected and data is read and stored in an Array. Note that this instruction only reads the fifo element location but does not transfer the data.
3. Finally, execute the Fetch RX FIFO data instruction to actually retrieve the data from the fifo.

Fetch RX FIFO Data Instruction:

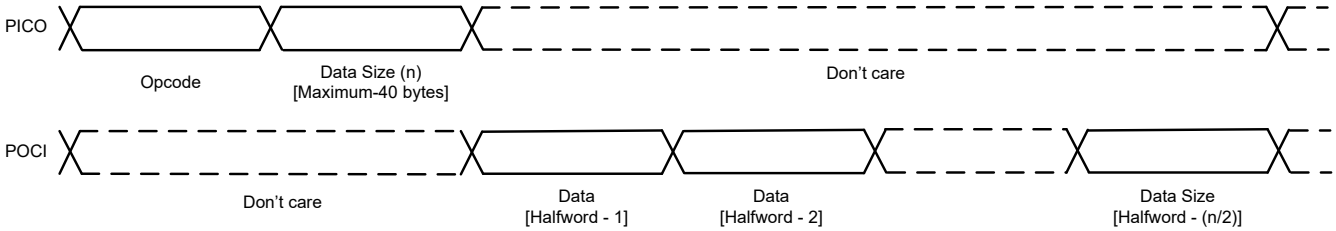

Figure 2-15. Fetch RXFIFO Data Instruction format

Opcode: 0b0000000011100000

The Fetch RX FIFO Data Instruction transmits the RX FIFO data that was read using Read RX FIFO instruction.

Read Receive Status Data Instruction

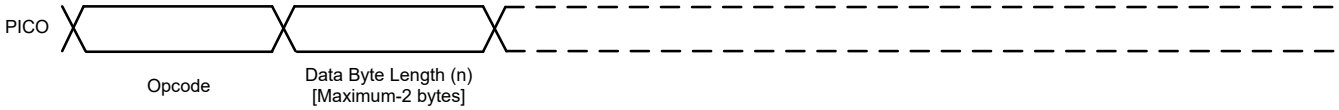

Figure 2-16. Read Receive Status Data Instruction Format

Opcode: 0b0000000011110000

The Read receive status data instruction is used to get status bits of CAN module from SPI to CAN device. The meaning of data bits received as response to this instruction is shown in Figure 2-17.

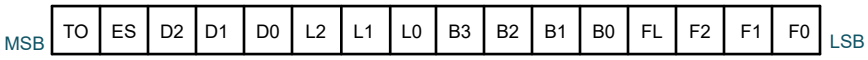

Figure 2-17. Status Bits indication

- **B0 – B4**: RX Buffers New data status from NDAT register. Each bit indicates status of each RX buffer. 1: Corresponding Rx buffer has CAN Frame. 0: Corresponding Rx buffer is empty.
- **FL**: Indicates the CAN RXFIFO full flag status. 1: RXFIFO is full. 0: RXFIFO is not full.
- **F2, F1, F0**: Indicates CAN RXFIFO fill level. 1: Corresponding Rx FIFO has CAN Frame. 0: Corresponding Rx FIFO is empty.
- **L2, L1, L0**: Indicates LEC (Last Error Code) bits in CAN protocol.

Table 2-5. Last Error Code (LEC) Bit Diagnostic Mapping

LEC Bits [L2, L1, L0]	Error code
000	No Error
001	Stuff Error
010	Form Error
011	Ack Error

Table 2-5. Last Error Code (LEC) Bit Diagnostic Mapping (continued)

LEC Bits [L2, L1, L0]	Error code
100	Bit1 Error
101	Bit0 Error
110	CRC Error
111	No change

- **D2, D1, D0**: Indicates DLEC (Data Phase Last Error Code) bits in CAN Protocol. Error codes are same as for LEC.
- **ES**: Indicates Error status of previous instruction. 1: Error while receiving previous instruction. 0: No Error while receiving previous instruction.
- **TO**: Timeout status of previous instruction. 1: Timeout occurred for previous instruction. 0: No Timeout occurred for previous instruction

Fetch Receive Status Data Instruction

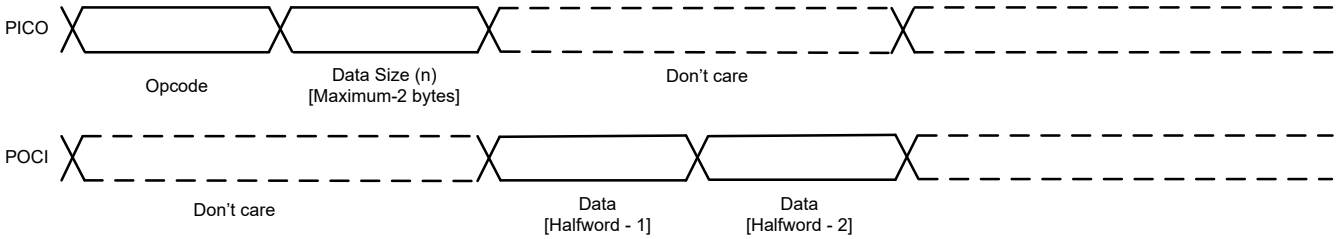

Figure 2-18. Fetch Receive Status Data Instruction Format

Opcode: 0b0000000000001000

The Fetch RX Status Data Instruction is used to receive the RX Status data which has already been read and stored in a variable with Read RX Status data instruction.

2.2 Timeout Feature

The SPI-CAN bridge implements timeout protection with the following behavior:

- Message Timeout:
  - If a complete instruction is not received within five seconds.
  - The bridge triggers a timeout.
  - The incomplete instruction is discarded.
- Buffer Loading Timeout:
  - If timeout occurs during TX buffer or FIFO loading.
  - Partial data can be written to message RAM.
  - Subsequent transmit instructions to the same buffer starts from the beginning.
  - Previous incomplete data is effectively discarded.

External SPI controller can use *Read receive status instruction* to check timeout events. This allows the controller to take appropriate recovery actions.

2.3 Error Indication

An error is generated in the below conditions:

- When instruction received by the Bridge device is not valid (received instruction opcode is not included in the list of instructions).
- When offset address received by the bridge is not valid (received offset address not related to any of the CAN register address).

External SPI controller can use *Read receive status instruction* to check Error indication for the previous instruction.

2.4 Busy Status Indication

The SPI-CAN bridge uses a GPIO pin to indicate busy status:

1. The BUSY pin is asserted when:
   - An opcode is received from the SPI controller.
   - CAN network activity is in progress.
2. The BUSY pin is cleared when:
   - The current instruction completes processing.
   - The CAN transfer finishes.

Important: The external SPI controller must not initiate new transactions while the BUSY pin is asserted. This prevents conflicts and maintains reliable data transfer.

2.5 Message RAM Configuration

When Power Enable and CAN Configuration Instruction is received, the predefined CAN message RAM configurations are setup. Default message RAM configuration is shown in Figure 2-19.

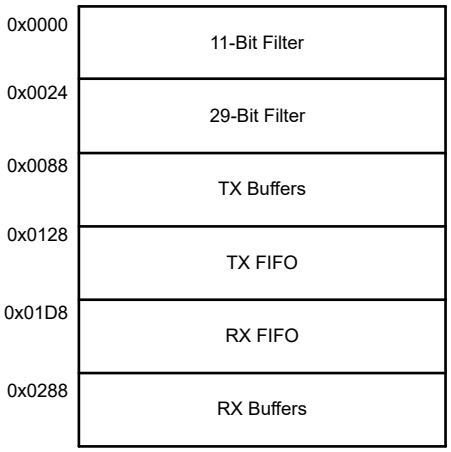

Figure 2-19. Message Ram Configurations

Message ID, transmit FIFO and transmit buffer can be configured by external SPI master as detailed in Section 2.1.1.

2.6 Test Environment

Test environment is detailed in Figure 2-20; SPI-CAN bridge is implemented on MSPM0G3507 launchpad, an external SPI controller is realized on another MSPM0G3507 launch pad. SPI-CAN bridge is connected to an external node through CAN-PHY.

Pin Connectivity

Table 2-6. Pin Connectivity

Device	Pin function	Pin number
MSPM0G3507 – SPI Controller	SCLK	PA17
	PICO	PA18
	POCI	PA16
	CS	PB1
	Busy Status	PB7
	CAN new message received indication	PB8
MSPM0G3507 - Bridge	SCLK	PA17
	PICO	PA18
	POCI	PA16
	CS	PB1
	Busy Status	PB7
	CAN new message received indication	PB8
	CAN_TX	PA12
	CAN_RX	PA13
MSPM0G3507 – CAN Node	CAN_TX	PA12
	CAN_RX	PA13

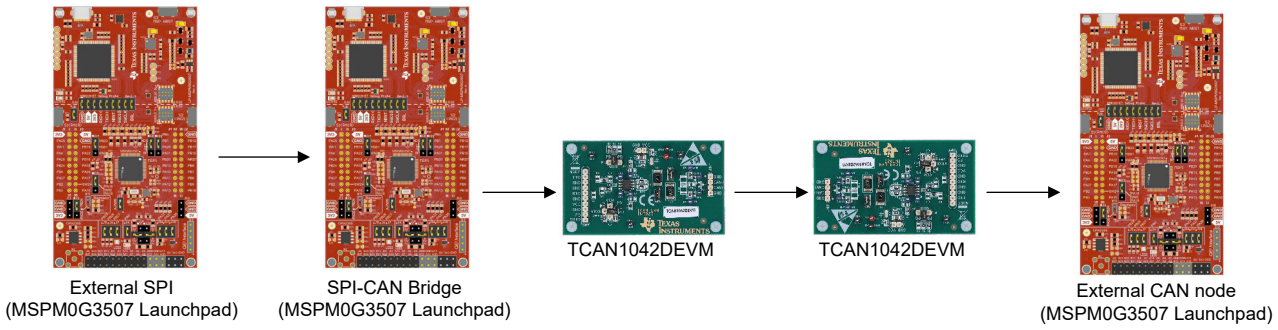

Figure 2-20. Test Setup for SPI-CAN Bridge

Application code can be downloaded at MSPM0-SDK Software development kit (SDK) | TI.com.

3 References

- MSPM0G3507 data sheet, product information and support | TI.com
- LP-MSPM0G3507 Evaluation board | TI.com
- TCAN1042DEVM Evaluation board | TI.com
- MSPM0-SDK Software development kit (SDK) | TI.com

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265