CAN Bus Conversion Module

BM42D7611-1

Revision: V1.00 Date: November 27, 2024

www.bestmodulescorp.com

Table of Contents

Features

- Operaing voltage (V_{CC}): 3.15V~3.6V
- Operating current: 13mA @ 3.3V
- Bi-directional conversion between SPI/UART and CAN interfaces
- Single SPI interface: communication speed up to 500kHz in non-custom mode and up to 300kHz in custom mode (at least 30μs between bytes in these modes)
- Single UART interface supports baud rates from 1200bps to 128kbps
- Single CAN interface supports baud rates from 5kbps to 1Mbps
- Three transmission modes: transparent transmission, transparent transmission with ID, custom conversion
- 6 sets of CAN receiving filter settings
- Operating temperature: -40℃~125℃

Product Picture

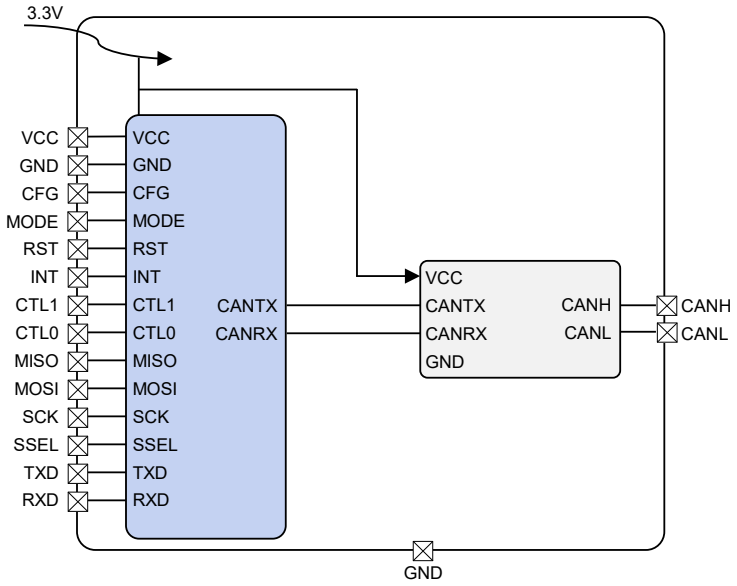
General Description

The BM42D7611-1 is an non-isolated SPI/UART-to-CAN conversion module that integrates a microcontroller and a CAN transceiver. The module is used to convert SPI/UART interface to CAN interface. In case of insufficient functional peripherals, users can use this module to extend the CAN interface and send data information to another CAN interface via SPI or UART, thus implementing the data communication between SPI or UART devices and CAN bus network.

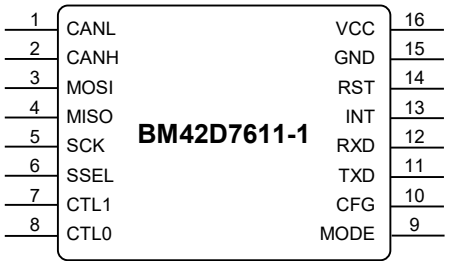
Applications

- Industrial automation systems
- Home electronics
- Power monitor systems
- Agricultural machinery
- Robots

Block Diagram



Pin Assignment



Pin Description

Pin	Function	Type	Description
1	CANL	—	CANL pin
2	CANH	—	CANH pin
3	MOSI	I	SPI MOSI pin
4	MISO	O	SPI MISO pin
5	SCK	I	SPI SCK pin
6	SSEL	I	SPI chip selection pin
7	CTL1	I	SPI master control pin1
8	CTL0	I	SPI master control pin0
9	MODE	I	Mode control pin
10	CFG	I	Configuration pin
11	TXD	O	Module UART transmitting pin
12	RXD	I	Module UART receiving pin
13	INT	O	Slave feedback pin
14	RST	I	Reset signal input; module resets when the pin is logic low
15	GND	PWR	Logical negative power supply
16	VCC	PWR	Logical positive power supply

Note: PWR: power supply; I: input; O: output

Technical Specifications

Absolute Maximum Ratings

Parameter	Value	Unit
V_{CC}	-0.3~+3.6	V
Storage Temperature	-40~125	°C
Storage Relative Humidity	20%~60%	RH
Operating Temperature	-40~125	°C
Ambient Humidity	10%~95%	RH

Recommended Operating Conditions

Parameter	Value	Unit
V_{CC}	3.15~3.6	V
Operating Temperature	-40~125	°C
Ambient Humidity	10%~95%	RH

Note: The Absolute Maximum Ratings indicate limitations beyond which damage to the device may occur. Recommended Operating Ratings indicate conditions for which the device is intended to be functional, but do not guarantee specified performance limits.

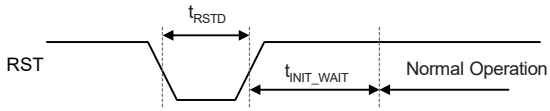
D.C. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{CC}	Logical Power Supply	—	3.15	—	3.6	V
V_{IL}	Input Logic Low Voltage	$3.15 \le V_{CC} \le 3.6V$	-0.5	—	$0.35V_{CC}$	V
V_{IH}	Input Logic High Voltage	$3.15 \le V_{CC} \le 3.6V$	$0.65\ V_{CC}$	—	$0.5+V_{CC}$	V
I_{DD}	Operating Current	V_{CC}=3.3V, UART baud rate: 128kbps, CAN baud rate: 1Mbps	—	—	13	mA

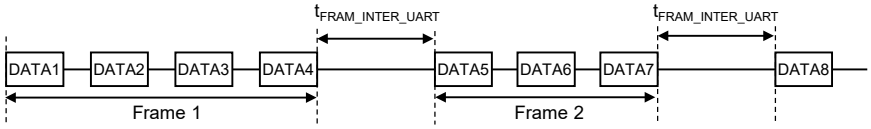
A.C. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
t_{RSTD}	RST Low Voltage Width to System Reset	Figure: Reset Timing	100	—	—	μs
t_{INIT_WAIT}	Waiting Time for Initialisation	Figure: Reset Timing	55	—	—	ms
f_{CLK_UART}	UART Baud Rate	—	1200	—	128k	bps
f_{CLK_SPI}	SPI Clock	Non-custom protocol conversion	—	—	500k	Hz
		Custom protocol conversion	—	—	300k	
f_{CLK_CAN}	CAN Baud Rate	—	5k	—	1M	bps
$t_{FRAM\_INTER\_UART}$	UART Frame Interval (Note)	Figure:UART Frame Interval	$10\times n/f_{CLK\_UART}$	—	—	s
$t_{BYTE\_TO\_BYTE\_SPI}$	SPI Byte Interval	Figure: SPI Byte Interval	30	—	—	μs
$t_{FRAM\_INTER\_SPI}$	SPI Frame Interval	Figure:SPI Frame Interval	40	—	—	μs
t_{SWITCH_WAIT}	SPI State Switching Operation Delay	Figure: SPI State Switching	50	—	—	μs
t_{WRITE_HOLD}	State Hold Time after SPI Write Operation	Figure: SPI State Switching	5	—	—	μs
t_{CFG_WAIT}	Waiting Time after Configuration	Write configuration	260	—	—	ms
		Read configuration	5	—	—	ms

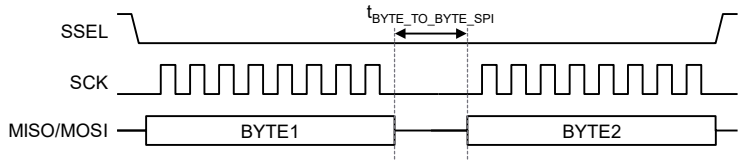
Note: n is the number of UART frame interval characters which is configured by the user.



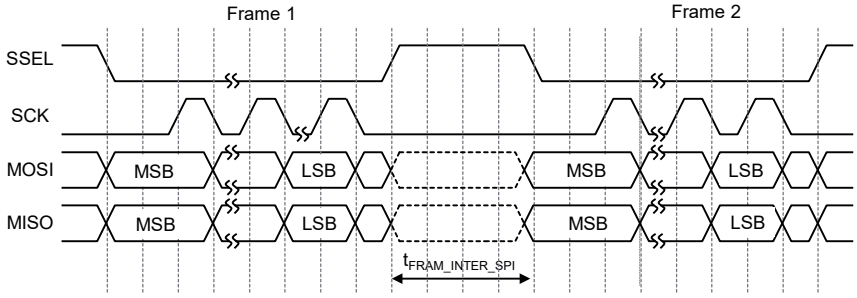
Reset Timing



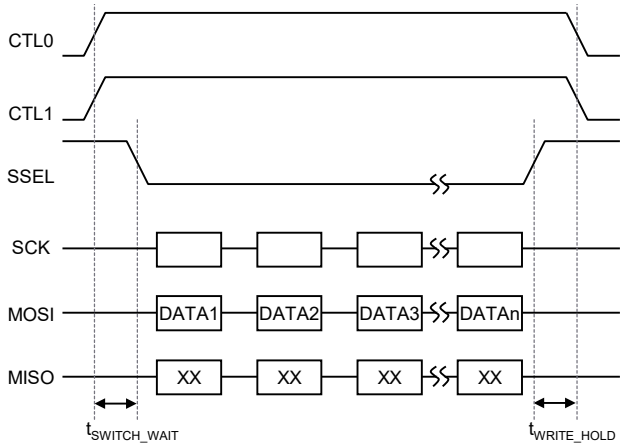
UART Frame Interval



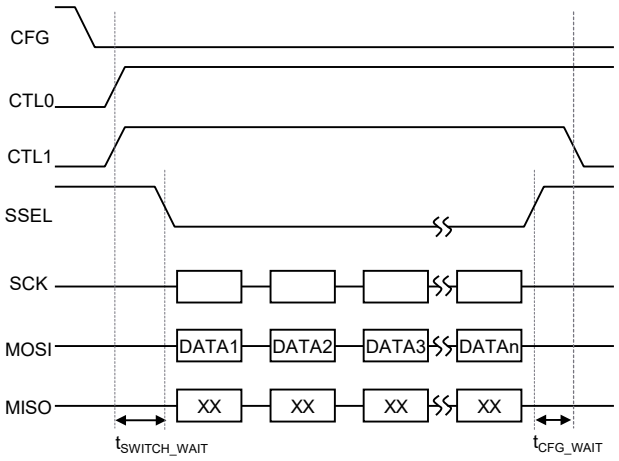
SPI Byte Interval



SPI Frame Interval



SPI State Switching



SPI Write Configuration

CAN Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
f_{CLK_CAN}	CAN Communication Baud Rate	—	5	—	1000	kbps
$V_{O(D)}$	Dominant Level (Logic 0) Output Voltage	R_L=60Ω, CANH	2.45	—	V_{CC}	V
		R_L=60Ω, CANL	0.5	—	1.25	
$V_{O(R)}$	Recessive Level (Logic 1) Output Voltage	R_L=60Ω, CANH	—	2.3	—	V
		R_L=60Ω, CANL	—	2.3	—	
$V_{DIFF(D)}$	Dominant Differential Output Voltage	R_L=60Ω	1.2	2	3	V
$V_{DIFF(R)}$	Recessive Differential Output Voltage	No load	-0.5	—	0.05	
V_X	Maximum Withstand Voltage on Bus Side	—	-36	—	36	V
CAN Bus Interface		Conforming to the ISO 11898-2 standard, twisted pair output				

Functional Description

System Description

The BM42D7611-1 is a professional non-isolated SPI/UART-to-CAN conversion module. Users can use this module to send data by converting from UART or SPI format to CAN format. Various transmission speeds are available, up to 1Mbps for CAN, up to 128kbps for UART, and up to 500kHz for SPI. Three transmission directions are available, bi-directional, SPI/UART to CAN only and CAN to SPI/UART only. Three transmission modes are available, transparent transmission, transparent transmission with ID, and custom protocol transmission.

Operating Mode

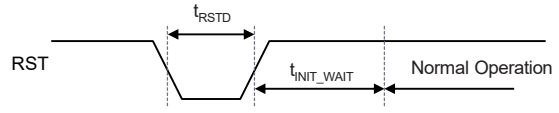
The module operates in the SPI-to-CAN mode when the MODE pin is high, and operates in the UART-to-CAN mode when the MODE pin is low. The CFG pin is used to control whether the module operates in the configuration mode or data transmission mode. The CTL0 and CTL1 pins are used to control the module to switch between several sub-modes in the SPI-to-CAN mode, which are idle, master reading module status, master reading data and master writing data. In addition, the INT pin is used as a feedback pin. The SPI slave notifies the master to read the configuration response frame, hardware identification response frame, configuration frame or data frame by pulling low the INT pin, then automatically pulls the INT pin high after the master finishes reading the corresponding information. After the master finishes sending the configuration command frame, it should reset the module by pulling low the RST pin so as to complete the configuration.

MODE	CFG	Operating Mode
0	0	UART-to-CAN mode, UART configuration
0	1	UART-to-CAN mode, data transmission
1	0	SPI-to-CAN mode, SPI configuration
1	1	SPI-to-CAN mode, data transmission

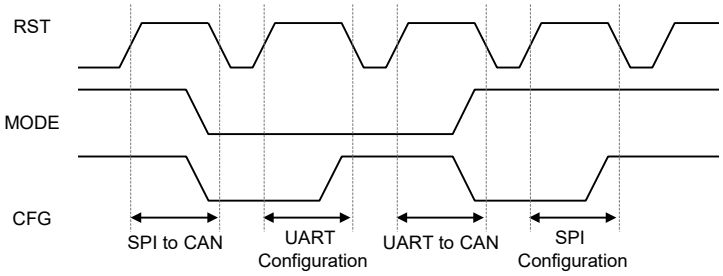
Operating Mode Switching Truth Table

To switch between different operating modes, it is required to pull low the RST pin to reset the module after changing the corresponding pin levels, this makes the module enter the desired operating mode.

To make the module reset successful, pull low the RST pin for at least 100μs, and wait for at least 55ms after reset until the product initialisation is completed. For functional control pins other than the RST pin, wait for at least 50μs after switching the pin state to allow the module to switch the operating state.



Reset Timing



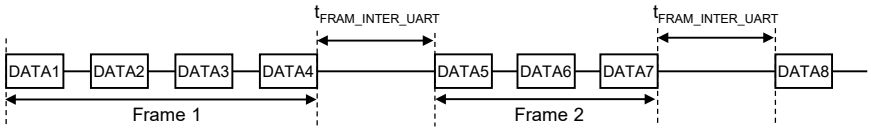
Operating Mode Switching Timing Diagram

UART Communication Mode

In the UART mode, the SPI transmission function is disabled and the SPI interface is invalid. The UART communication format is fixed as 1 start bit, 8 data bits, no parity bit and 1 stop bit. The UART communication rate ranges from 1200bps to 128kbps.

UART Frame

Starting from the first data received by the module UART interface, until a certain time interval has passed and no new data has been received, the data received during this time period is defined as a frame of data and the time interval is defined as the UART frame interval. The frame interval represents the time period for the UART to send n characters, where n ranges from 0x02 to 0x0A and is configured by the user. Under different baud rates, the time period for sending the same n characters will be different. If the the UART frame interval parameter is set to 0x02 representing 2 characters, the baud rate is 9600bps and each character has 10 bits (1 start bit, 8 data bits, 1 stop bit), then the time interval will be 2×10/9600 ≈ 2ms. The UART frame format is shown below:



UART Frame Interval

SPI Communication Mode

This module can only be used as an SPI slave when operating in the SPI mode. The SCK pin is low in the idle state, the data is read at rising edges, the data is 8-bit wide with the MSB first. The maximum communication speed is 500kHz for the transparent conversion and transparent conversion with ID, and the maximum communication speed is 300kHz for the custom protocol conversion.

In the SPI mode, the UART transmission function is disabled so the module cannot send and receive data using the UART-to-CAN mode.

Master Control

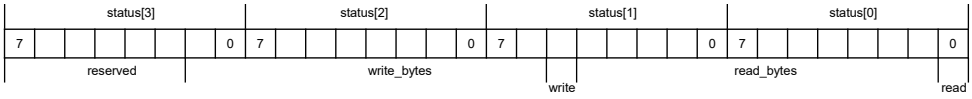
CFG	CTL0	CTL1	Function
0	0	0	SPI configuration, idle
0	0	1	SPI configuration, master reads module status
0	1	0	SPI configuration, master reads response frame and configuration frame
0	1	1	SPI configuration, master writes command
1	0	0	Data transmission, idle
1	0	1	Data transmission, master reads module status
1	1	0	Data transmission, master reads data
1	1	1	Data transmission, master writes data

SPI-to-CAN Mode Master Control Truth Table

In the SPI-to-CAN mode, when the CTL0 pin is low and the CTL1 pin is high, namely the master is in the reading state, the master can obtain the number of bytes that can be read and the number of bytes that can be written. The master can obtain the current module status by reading 4 bytes through the SPI interface.

Bit	Meaning	Symbol	Description
0	Read indication bit	read	read=1 if CAN receiver buffer is not empty, otherwise read=0
12:1	Number of readable bytes	read_bytes	The number of CAN data bytes that can be read by the master
13	Write indication bit	write	write=1 if CAN transmitter buffer is not full, otherwise write=0
25:14	Number of writable bytes	write_bytes	The number of data bytes that can be written into the module
31:26	Reserved bits	reserved	Reserved

SPI Mode Status Code Composition



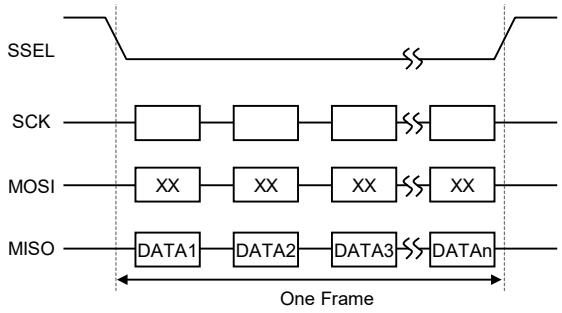
Status Byte Structure

After obtaining 4 status bytes, extract the corresponding bits for judgement and processing, as shown below.

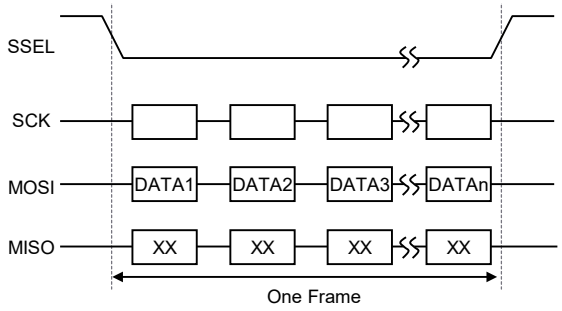
```
read = status[0]&0x01;                          // Read indication bit
read_bytes = ((status[0] & 0xfe)>>1) + ((status[1] & 0x1f)<<7);
                                                // Number of readable bytes
write = (status[1] & (1<<5)) ? 1:0; // Write indication bit
write_bytes = ((status[1]&0xc0)>>6)+((status[2]&0xff)<<2)+((status[3]&0x03)<<10);
                                                // Number of writable bytes
```

SPI Frame

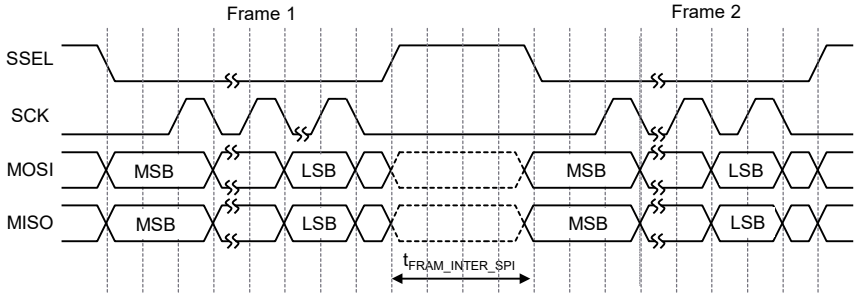
The data between a valid SPI chip selection and a subsequent adjacent invalid chip selection is defined as one data frame. The SPI receiver buffer is able to store up to 124 bytes and the master SPI can send up to 124 bytes to the module for each frame. When the module buffer is full, no more new data will be received until the module has sent out all the data in the SPI receiver buffer via the CAN bus.



SPI Reading Data



SPI Writing Data



SPI Frame Interval

INT Feedback Pin

The module can only be used as an SPI slave and cannot actively control the chip selection pin to send data to the master. Therefore, when the module CAN interface has received data and a certain situation occurs, the module will pull low the INT pin to notify the master to read data. After the master finishes reading the data, the INT pin will return to a high level. There are following two situations in which the INT pin will be pulled low.

- The number of the CAN frames received in the CAN buffer has reached the preset trigger threshold which is set by the configuration frame sent from the master in the configuration mode.

  When this situation occurs, the INT pin will be pulled low to notify the master to read data. The INT pin remains low until all the data in the CAN receiver buffer has been read by the master, then the INT pin returns to high.

- The number of the CAN frames received in the CAN buffer is less than the preset trigger threshold and the master does not read the data for a certain time period after the module CAN receives the data. Here the time period is set by the configuration frame sent from the master in the configuration mode.

  When this situation occurs, the INT pin will be pulled low to notify the master to read data. The INT pin remains low until all the data in the CAN receive buffer has been read by the master, then the INT pin returns to high.

Data Conversion Methods

A data conversion method refers to a basic criteria for data conversion between the serial bus and CAN bus. The module can only operate in one of the data conversion methods at a time. To change the conversion method, users need to reconfigure the module. The module supports transparent conversion, transparent conversion with ID and custom protocol conversion.

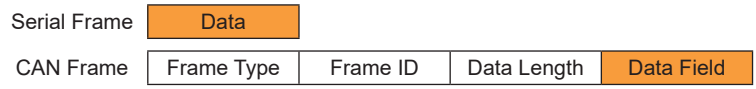
Transparent Conversion

Transparent conversion means that either side of the bus receives data and sends the data to the other side without any processing.

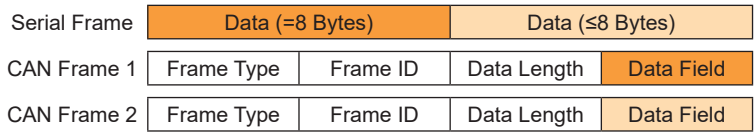
Serial Frame to CAN Frame

The data length of a CAN frame is up to 8 bytes. When the data length of a serial frame is less than or equal to 8 bytes, the data is sent out through a CAN frame. If the data length of a serial frame is greater than 8 bytes, the module each time takes 8 bytes of data from the serial frame and fills them into the CAN frame until all the data have been converted and sent out.

The data length value in the CAN frame is determined by the number of data bytes actually allocated to the CAN frame. The frame ID and frame type are determined by user configuration. To change the frame ID and frame type, users need to reconfigure the module using a configuration command which takes effect after restart. The following are schematic diagrams of data conversion:



Serial Frame to CAN Frame (Transparent Conversion, ≤8 Data Bytes)



Serial Frame to CAN Frame (Transparent Conversion, >8 Data Bytes)

- Conversion Examples

  Example 1: The user-configured CAN frame type is standard frame, the frame ID is 0x0123 (standard frame ID length: 11 bits, range: 0x000~0x7FF), and the data of the serial frame is 0x01~0x08, a total of 8 data bytes. The conversion of a serial frame to a CAN frame is shown in the following table:

Serial Frame	Data							
	01	02	03	04	05	06	07	08

CAN Frame	Frame Type	Frame ID1	Frame ID0	Data Length	Data Field							
	Standard Frame	01	23	08	01	02	03	04	05	06	07	08

Serial Frame to CAN Frame Example (Transparent Conversion, ≤8 Data Bytes)

Example 2: The user-configured CAN frame type is extended frame, the frame ID is 0x00012345 (extended frame ID length: 29 bits, range: 0x00000000~0x1FFFFFFF), and the data of the serial frame is 0x01~0x0C, a total of 12 data bytes. The conversion of a serial frame to two CAN frames is shown in the following table:

Serial Frame	Data								Data			
	01	02	03	04	05	06	07	08	09	0A	0B	0C

CAN Frame 1	Frame Type	Frame ID3	Frame ID2	Frame ID1	Frame ID0	Data Length	Data Field							
	Extended Frame	00	01	23	45	08	01	02	03	04	05	06	07	08

CAN Frame 2	Frame Type	Frame ID3	Frame ID2	Frame ID1	Frame ID0	Data Length	Data Field			
	Extended Frame	00	01	23	45	04	09	0A	0B	0C

Serial Frame to CAN Frame Example (Transparent Conversion, >8 Data Bytes)

CAN Frame to UART Frame

After receving a frame of data, the module immediately transfers it to the UART interface. Therefore each CAN frame data is converted into one UART frame.

- Frame ID and Frame Information Conversion Disabled

  When a CAN frame is converted to a UART frame, the UART frame only contains the data of the CAN frame, without frame information and frame ID bytes, as shown below.

CAN Frame	Frame Type	Frame ID	Data Length	Data Field
UART Frame	Data			

CAN Frame to UART Frame (Transparent Conversion)

  - Conversion Example

    The first CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x06, a total of 6 data bytes. The CAN frame and the converted UART frame are shown in the following table:

CAN Frame	Frame Type	Frame ID1	Frame ID0	Data Length	Data Field					
	Standard Frame	01	23	06	01	02	03	04	05	06

UART Frame	Data					
	01	02	03	04	05	06

CAN Frame to UART Frame Example (Transparent Conversion)

- Frame Information Conversion Enabled

  When the frame information conversion is enabled, the UART frame contains the frame type, data length and data of the CAN frame. After conversion, the UART frame starts with a frame information byte, followed by data. The frame information includes the frame type and data length which are expressed in one byte. The high nibble represents the frame type and the low nibble represents the data length. For the high nibble, “0b1000” represents the extended frame and “0b0000” represents the standard frame. The low nibble has a range of “0b0000~0b1000” which represents “0~8” bytes of data respectively.

  For example:

  If the frame information is 0x06, it indicates that the received CAN frame is a standard frame and the CAN data length is 6 bytes.

If the frame information is 0x87, it indicates that the received CAN frame is an extended frame and the CAN data length is 7 bytes.

Each CAN frame is converted into a UART frame, as shown below:

CAN Frame	Frame Type	Frame ID	Data Length	Data Field
UART Frame	Frame Information	Data		

CAN Frame to UART Frame (Transparent Conversion, Frame Information Enabled)

- Conversion Example

The CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x06, a total of 6 data bytes. The CAN frame and the converted UART frame are shown in the table:

CAN Frame	Frame Type	Frame ID1	Frame ID0	Data Length	Data Field						
	Standard Frame	01	23	06	01	02	03	04	05	06	
UART Frame	Frame Information	Data									
	06	01	02	03	04	05	06				

CAN Frame to UART Frame Example (Transparent Conversion, Frame Information Enabled)

Description:

UART frame information (06) = combination of standard CAN frame (0) and data length (6).

- Frame ID Conversion Enabled

When the frame ID conversion has been enabled by user configuration, the frame information conversion will be enabled by default. Therefore, after a CAN frame is converted to a UART frame, the UART frame includes one frame information byte (including frame type and data length), frame ID bytes (2 bytes for standard frame and 4 bytes for extended frame), and frame data in sequence. The frame ID is converted with its MSB first. The conversion diagram is shown below:

CAN Frame	Frame Type	Frame ID	Data Length	Data Field
UART Frame	Frame Information	Frame ID	Data	

CAN Frame to UART Frame (Transparent Conversion, Frame ID Enabled)

- Conversion Example

When the frame ID conversion has been enabled by user configuration, the frame information conversion will be enabled by default. The CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x07, a total of 7 data bytes. The CAN frame and the converted UART frame are shown in the table:

CAN Frame	Frame Type	Frame ID1	Frame ID0	Data Length	Data Field						
	Standard Frame	01	23	07	01	02	03	04	05	06	07
UART Frame	Frame Information	Frame ID1	Frame ID0	Data							
	07	01	23	01	02	03	04	05	06	07	

CAN Frame to UART Frame Example (Transparent Conversion, Frame ID Enabled)

CAN Frame to SPI Frame

The module SPI awalys acts as a slave. Each frame received by the module CAN bus will be immediately stored into the CAN buffer. Once the number of the received frames reaches the feedback threshold or when the trigger time is reached, the INT pin will be pulled low to notify the master to read data. When the master detects that the INT pin is low, it will retrieve all the data from the CAN buffer with one SPI frame.

There are three data conversion formats for CAN frame to SPI frame conversion. For the first one, when the CAN interface receives a CAN frame, only the CAN data is stored into the buffer. For the second one, when the CAN interface receives a CAN frame, the CAN frame information and data are stored into the buffer. The frame information refers to the frame type and data length of the CAN frame, which occupies one byte, followed by data. This conversion format is called Frame Information Conversion Enabled. For the third one, when the CAN interface receives a CAN frame, the frame information, frame ID and frame data of the CAN frame are all stored into the buffer. The frame information includes the frame type and data length which are expressed in one byte. The frame ID is 2 bytes (standard frame) or 4 bytes (extended frame) long. In the order of frame information, frame ID and data, this conversion format is call Frame ID Conversion Enabled. The following is an introduction to each conversion format.

- Frame ID and Frame Information Conversion Disabled

  When a CAN frame is converted to an SPI frame, the SPI frame only contains the data of the CAN frame, without frame information and frame ID bytes. The conversion diagram is as follows:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field
SPI Frame	Data Segment 1	…		Data Segment n

CAN Frame to SPI Frame (Transparent Conversion)

  - Conversion Example

    The first CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x06, a total of 6 data bytes. The n^{th} CAN frame received is an extended frame, the frame ID is 0x00012345, and the data is 0x00, 0x11~0x16, a total of 7 data bytes. The CAN frames and the converted SPI frame are shown in the following table:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field								
	Standard Frame	0123	06	01	02	03	04	05	06			
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field								
	Extended Frame	00012345	07	00	11	12	13	14	15	16		
SPI Frame	Data Segment 1		…	Data Segment n								
	01	02	03	04	05	06	…	00	11	12	13	14	15	16

CAN Frame to SPI Frame Example (Transparent Conversion)

- Frame Information Conversion Enabled

  When the frame information conversion is enabled, the SPI frame contains the frame type, data length and data of the CAN frame. After conversion, the SPI frame starts with a frame information byte, followed by data. The frame information includes the frame type and data length which are expressed in one byte. The high nibble represents the frame type and the low nibble represents the data length. For the high nibble, “0b1000” represents the extended frame and “0b0000” represents the standard frame. The low nibble has a range of “0b0000~0b1000” which represents “0~8” bytes of data respectively.

For example:

If the frame information is 0x06, it indicates that the received CAN frame is a standard frame and the CAN data length is 6 bytes.

If the frame information is 0x87, it indicates that the received CAN frame is an extended frame and the CAN data length is 7 bytes.

After the frame information conversion is enabled, the conversion from CAN frames to SPI frame is shown in the following table:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field		
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field		
SPI Frame	Frame Information 1	Data Segment 1	…	Frame Information n	Data Segment n	

CAN Frame to SPI Frame (Transparent Conversion, Frame Information Enabled)

- Conversion Example

The first CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x06, a total of 6 data bytes. The n^{th} CAN frame received is an extended frame, the frame ID is 0x00012345, and the data is 0x00, 0x11~0x16, a total of 7 data bytes. The CAN frames and the converted SPI frame are shown in the table:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field										
	Standard Frame	0123	06	01	02	03	04	05	06					
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field										
	Extended Frame	00012345	07	00	11	12	13	14	15	16				
SPI Frame	Frame Information 1	Data Segment 1							…	Frame Information n	Data Segment n			
	06	01	02	03	04	05	06	…	87	00	11	12	13	14	15	16

CAN Frame to SPI Frame Example (Transparent Conversion, Frame Information Enabled)

Description:

SPI frame information 1 (06) = combination of standard CAN frame 1 (0) and data length (6);

SPI frame information n (87) = combination of extended CAN frame n (8) and data length (7).

- Frame ID Conversion Enabled

When the frame ID conversion has been enabled by user configuration, the frame information conversion will be enabled by default. Therefore, after a CAN frame is converted to an SPI frame, the SPI frame includes one byte of frame information (including frame type and data length), frame ID (2 bytes for standard frame and 4 bytes for extended frame), and frame data in sequence. The frame ID is converted with its MSB first. The conversion diagram is shown below:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field				
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field				
SPI Frame	Frame Information 1	Frame 1 ID	Data Segment 1	…	Frame Information n	Frame n ID	Data Segment n	

CAN Frame to SPI Frame (Transparent Conversion, Framed ID Enabled)

- Conversion Example

  When the frame ID conversion has been enabled by user configuration, the frame information conversion will be enabled by default. The first CAN frame received by the CAN interface is a standard frame, the frame ID is 0x0123, and the data is 0x01~0x06, a total of 6 data bytes. The n^{th} CAN frame received is an extended frame, the frame ID is 0x00012345, and the data is 0x00, 0x11~0x16, a total of 7 data bytes. The CAN frames and the converted SPI frame are shown in the table:

	Frame Type	Frame ID	Data Length	Data Field						
CAN Frame 1	Standard Frame	0123	06	01	02	03	04	05	06	
CAN Frame n	Extended Frame	00012345	07	00	11	12	13	14	15	16

SPI Frame	Frame Information 1	Frame ID		Data Segment 1						…	Frame Information n	Frame ID			Data Segment n							
	06	01	23	01	02	03	04	05	06		87	00	01	23	45	00	11	12	13	14	15	16

CAN Frame to SPI Frame Example (Transparent Conversion, Framed ID Enabled)

Transparent Conversion with ID

The transparent conversion with ID means that the serial frame sent or received contains valid CAN frame ID bytes. Before the serial frame to CAN frame conversion, the start address and length of the frame ID has been set by user configuration, according to this the module extracts the frame ID from the serial frame and fills it into the frame ID field of the CAN frame, then the data to the data field. Similarly, for the CAN frame to serial frame conversion, the module extracts the frame ID from the CAN frame and converts it into the corresponding position of the serial frame, and also converts the frame information. In this mode, users can send CAN data with different frame IDs on the same node.

Serial Frame to CAN Frame

The start address and length of the serial frame ID are determined by user configuration. The start address ranges from 0 to 7, the length ranges from 1 to 2 for the standard frame or from 1 to 4 for the extended frame. To change the start address and length of the serial frame ID, users need to reconfigure the module using a configuration command which takes effect after restart.

When the serial frame data length is less than or equal to 8 bytes, the data is converted into one CAN frame and sent out. However, if one CAN frame could not include all the serial frame data, the same frame ID is still used to continue the conversion until the serial frame conversion is completed.

Regarding the transparent conversion with ID, the CAN frame type, the start address and length of the serial frame ID are determined by user configuration and always remain the same unless the user reconfigures the module. The data length of the CAN frame is determined by the number of data bytes actually allocated to the CAN frame.

Serial Frame	Data	Frame ID	Data	
CAN Frame	Frame Type	Frame ID	Data Length	Data Field

Serial Frame to CAN Frame (Transparent Conversion with ID, ≤8 Data Bytes)

Serial Frame	Data 1	Frame ID	Data 2	Data 3	
CAN Frame 1	Frame Type	Frame ID	Data Length	Data 1	Data 2
CAN Frame 2	Frame Type	Frame ID	Data Length	Data 3	

Serial Frame to CAN Frame (Transparent Conversion with ID, >8 Data Bytes)

- Conversion Example

  The CAN frame type is configured to be extended frame, the serial frame ID start address is 2, the serial frame ID length is 3, and the data sent by the serial frame is 0x00~0x0F, a total of 16 data bytes. Since the configured CAN frame type is extended frame, the CAN frame ID will occupy 4 bytes. However, the configured ID start address in the serial frame is 2 and the ID length is 3, the ID extracted from the serial frame and filled into the CAN frame ID field is 0x020304, then complement a "00" to the least low byte, so the CAN frame ID will be 0x02030400. The conversion is shown in the following table:

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Serial Frame	Data		ID			Data										
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

CAN Frame 1	Frame Type	Frame ID				Data Length	Data							
	Extended Frame	02	03	04	00	08	00	01	05	06	07	08	09	0A

CAN Frame 2	Frame Type	Frame ID				Data Length	Data				
	Extended Frame	02	03	04	00	05	0B	0C	0D	0E	0F

Serial Frame to CAN Frame Example (Transparent Conversion with ID)

CAN Frame to UART Frame

After receving a frame of CAN data, the module immediately transfers it to the UART interface. In the transparent conversion with ID mode, the converted UART frame contains the CAN frame information, frame ID and frame data.

The start address and length of the serial frame ID are determined by user configuration. The start address ranges from 0 to 7, the length ranges from 1 to 2 for the standard frame or from 1 to 4 for the extended frame. To change the start address and length of the serial frame ID, users need to reconfigure the module using a configuration command which takes effect after restart.

The frame information includes the frame type and data length which are expressed in one byte. The high nibble represents the frame type and the low nibble represents the data length. For the high nibble, "0b1000" represents the extended frame and "0b0000" represents the standard frame. The low nibble has a range of "0b0000~0b1000" which represents "0~8" bytes of data respectively.

For example:

If the frame information is 0x06, it indicates that the received CAN frame is a standard frame and the CAN data length is 6 bytes.

If the frame information is 0x85, it indicates that the received CAN frame is an extended frame and the CAN data length is 5 bytes.

CAN Frame	Frame Type	Frame ID	Data Length	Data Field
UART Frame	Frame Information	Data	Frame ID	Data

CAN Frame to UART Frame (Transparent Conversion with ID)

- Conversion Example

  The CAN frame type to be received is extended frame, the frame ID start address is 2, the ID length is 3, the CAN ID is 0x10123456, and the data contains 8 bytes from 0x01 to 0x08. The conversion diagram is shown below:

CAN Frame	Frame Type	Frame ID				Data Length	Data							
	Extended Frame	10	12	34	56	08	01	02	03	04	05	06	07	08

Byte Address		0	1	2	3	4	5	6	7	8	9	10
UART Frame	Frame Information	Data		ID			Data					
	88	01	02	10	12	34	03	04	05	06	07	08

CAN Frame to UART Frame Example (Transparent Conversion with ID)

CAN Frame to SPI Frame

In the transparent conversion with ID mode, the module fills the frame ID and data into the SPI frame according to the configured frame ID start address and length, and the CAN frame information is stored in the SPI frame as the first byte. Therefore, after conversion the SPI frame contains the CAN frame information, frame ID (MSB first) and frame data.

The start address and length of the serial frame ID are determined by user configuration. The start address ranges from 0 to 7, the length ranges from 1 to 2 for the standard frame or from 1 to 4 for the extended frame. To change the start address and length of the serial frame ID, users need to reconfigure the module using a configuration command which takes effect after restart.

The frame information includes the frame type and data length which are expressed in one byte. The high nibble represents the frame type and the low nibble represents the data length. For the high nibble, “0b1000” represents the extended frame and “0b0000” represents the standard frame. The low nibble has a range of “0b0000~0b1000” which represents “0~8” bytes of data respectively.

For example:

If the frame information is 0x06, it indicates that the received CAN frame is a standard frame and the CAN data length is 6 bytes.

If the frame information is 0x87, it indicates that the received CAN frame is an extended frame and the CAN data length is 7 bytes.

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field				
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field				
SPI Frame	Frame 1 Information	Data 1	Frame 1 ID	Data 1	…	Frame n Information	Data n	Frame n ID	Data n

CAN Frame to SPI Frame (Transparent Conversion with ID)

- Conversion Example

  The CAN frame type to be received is standard frame, the serial frame ID start address is 2 and the frame ID length is 2. The first frame ID is 0x0123 and the data contains 6 bytes from 0x01 to 0x06, the n^{th} frame ID is 0x0345 and the data contains 4 bytes from 0x07 to 0x0A. The conversion diagram is shown below:

CAN Frame 1	Frame Type		Frame ID		Data Length	Data Field												
	Standard Frame		0123		06	01	02	03	04	05	06							
CAN Frame n	Frame Type		Frame ID		Data Length	Data Field												
	Standard Frame		0345		04	07	08	09	0A									
Byte Address		0	1	2	3	4	5	6	7			0	1	2	3	4	5	
SPI Frame	Frame Information 1	Data 1		ID 1		Data 1				…	Frame Information n	Data n		ID n		Data n		
	06	01	02	01	23	03	04	05	06		04	07	08	03	45	09	0A	

CAN Frame to SPI Frame Example (Transparent Conversion with ID)

Custom Protocol Conversion

For the custom protocol conversion, the serial frame consists of frame header, frame length, frame type, frame ID, data and frame tail.

Frame Header: serial frame start byte, 0x00~0xFF configured by user configuration

Frame Length: the total length of frame type, frame ID and data

Frame Type: CAN frame type, 0x00 reperesents standard frame, 0x08 represents extended frame

Frame ID: CAN frame ID, 2 bytes for standard frame ID, 4 bytes for extended frame ID

Data: CAN frame data

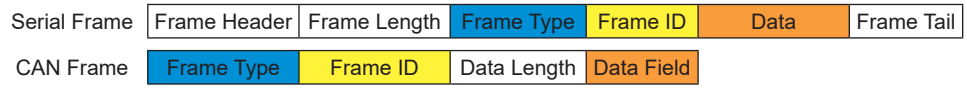
Frame Tail: serial frame end byte, 0x00~0xFF configured by user configuration

Only the serial frame that conforms to the defined format will be received and converted by the module, otherwise it will be discarded directly without any processing.

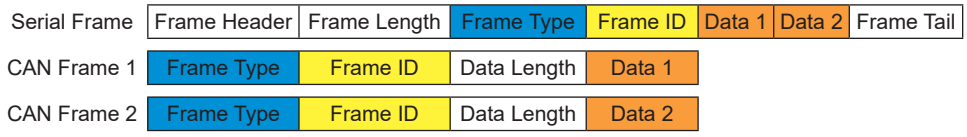
In the custom protocol conversion mode, users can send CAN frames with different IDs on the same node, and can also obtain the CAN frame ID from the serial frame.

Serial Frame to CAN Frame

When the serial frame data length is less than or equal to 8 bytes, the data is converted into one CAN frame and sent out. However, if one CAN frame could not include all the serial frame data, the same frame ID is still used to continue the conversion until the serial frame conversion is completed. The conversion diagram is shown below:



Serial Frame to CAN Frame (Custom Conversion, ≤ 8 Data Bytes)



Serial Frame to CAN Frame (Custom Conversion, > 8 Data Bytes)

- Conversion Example

  For example, by user configuration the serial frame header is 0x10 and the frame tail is 0x1B. The frame type to be transmitted is standard frame (0x00), the frame ID is 0x0123, and the data contains 10 bytes from 0x01 to 0x0A. Therefore, the frame length is the total length of fram type (1 byte), fram ID (2 bytes) and frame data (10 bytes), which is 0x0D. The conversion diagram is shown in the following table:

Serial Frame	Frame Header	Frame Length	Frame Type	Frame ID		Data										Frame Tail
	10	0D	00	01	23	01	02	03	04	05	06	07	08	09	0A	1B

CAN Frame 1	Frame Type	Frame ID		Data Length	Data					
	Standard Frame	01	23	08	01	02	03	04	05	06	07	08

CAN Frame 2	Frame Type	Frame ID		Data Length	Data	
	Standard Frame	01	23	02	09	0A

Serial Frame to CAN Frame Example (Custom Conversion)

CAN Frame to UART Frame

After receving a frame of CAN data, the module immediately converts it to the UART interface to send out. The conversion diagram is shown in the following table:

CAN Frame	Frame Type	Frame ID	Data Length	Data Field		
UART Frame	Frame Header	Frame Length	Frame Type	Frame ID	Data	Frame Tail

CAN Frame to UART Frame (Custom Conversion)

- Conversion Example

  For example, by user configuration the serial frame header is 0x10 and the frame tail is 0x1B. The CAN frame type is extended frame, the frame ID is 0x01234567, and the data contains 6 bytes from 0x01 to 0x06. The conversion diagram is shown in the following table:

CAN Frame	Frame Type	Frame ID				Data Length	Data					
	Extended Frame	01	23	45	67	06	01	02	03	04	05	06

UART Frame	Frame Header	Frame Length	Frame Type	Frame ID				Data						Frame Tail
	10	0B	08	01	23	45	67	01	02	03	04	05	06	1B

CAN Frame to UART Frame Example (Custom Conversion)

Description: Serial frame length (0x0B) = frame type (1 byte) + frame ID (4 bytes) + data (6 bytes).

CAN Frame to SPI Frame

When the CAN interface receives the data, the module converts the received CAN data into serial frame in user-defined format and stores them in the buffer. After receiving the data, if the received frame count meets the trigger threshold or if the the feedback time point is reached, the module will automatically pull low the INT pin to notify the SPI master to read data. The data received by CAN and converted to SPI frame in user-defined format can be read by the master SPI in a single frame. The conversion diagram is shown in the table:

CAN Frame 1	Frame Type	Frame ID	Data Length	Data Field		
CAN Frame n	Frame Type	Frame ID	Data Length	Data Field		

SPI Frame	Frame Header	Frame Length	Frame Type	Frame 1 ID	Data	Frame Tail	…	Frame Header	Frame Length	Frame Type	Frame n ID	Data	Frame Tail

CAN Frame to SPI Frame (Custom Conersion)

- Conversion Example

  For example, by user configuration the serial frame header is 0x10 and the frame tail is 0x0B. The frame type received by the CAN interface is standard frame, the first frame ID is 0x0123 and the data contains 6 bytes from 0x01~0x06, the n^{th} frame ID is 0x0234 and the data contains 4 bytes from 0x09 to 0x0C. The conversion diagram is as follows:

CAN Frame 1	Frame Type	Frame ID		Data Length	Data				
	Standard Frame	01	23	06	01	02	03	04	05	06

CAN Frame n	Frame Type	Frame ID		Data Length	Data			
	Standard Frame	02	34	04	09	0A	0B	0C

SPI Frame

Data Field 1											
Frame Header	Frame Length	Frame Type	Frame 1 ID		Data						Frame Tail
10	09	00	01	23	01	02	03	04	05	06	0B
	Data Field n										
…	Frame Header	Frame Length	Frame Type		Frame n ID		Data				Frame Tail
…	10	07	00		02	34	09	0A	0B	0C	0B

CAN Frame to SPI Frame Example (Custom Conersion)

Description:

Frame length of Data Field 1 (0x09) = frame type (1 byte) + frame 1 ID (2 bytes) + data (6 bytes).

Configuration Description

Conversion Parameters

Data Conversion Method

The data conversion method refers to a basic criteria for data conversion between the serial bus and CAN bus. The module can only operate in one data conversion method at a time. The module supports transparent conversion, transparent conversion with ID and custom protocol conversion, which have been described in their corresponding section.

Conversion Direction

This parameter refers to the direction in which the data is allowed to be converted. The module supports three conversion directions: bi-direction between SPI/UART and CAN, uni-direction from SPI/UART to CAN, and uni-direction from CAN to SPI/UART.

CAN Frame Information Converted to Serial Frame

This parameter is only available in the transparent conversion mode. After this option is enabled, the frame type and data length of CAN frames will be simultaneously converted to SPI/UART frames. For more details, refer to the Frame Information Conversion Enabled sections of the Transparent Conversion chapter.

CAN Frame ID Converted to Serial Frame

This parameter is only available in the transparent conversion mode. After this option is enabled, the frame type, data length and frame ID will be simultaneously converted to SPI/UART frames. For more details, refer to the Frame ID Conversion Enabled sections of the Transparent Conversion chapter.

CAN Frame ID Position in Serial Frame

This includes two parameters, the start address and length of the CAN frame ID, which are only available in the transparent conversion with ID mode. For more details, refer to the Transparent Conversion with ID chapter.

Frame Header and Frame Tail

These parameters which defines the head and tail of the frame are only available in the custom protocol conversion mode. For more details, refer to the Custom Protocol Conversion chapter.

SPI Parameters

Frame Count to Trigger Feedback

This parameter is only available in the SPI-to-CAN mode. As an SPI slave, the module cannot actively send data. When the module has received a certain number of CAN frames, it will inform the master to read the data by pulling low the INT pin. This parameter is in the unit of the received CAN frames. When the number of the received CAN frames reaches the preset value, the INT pin will be pulled low to trigger the feedback.

Time to Trigger Feedback

This parameter is only available in the SPI-to-CAN mode. As an SPI slave, the module cannot actively send data. When the number of the received CAN frames does not reach the preset value and the received data has not been read for a preset time period, the module will also pull low the INT pin to trigger the feedback.

UART Parameters

Baud Rate

This parameter refers to the UART baud rate for sending data. The valid baud rates of the serial port are listed in the UART and CAN baud rate code table.

Frame Interval

This parameter refers to the time inverval between UART communication frames. For more details, refer to the UART Communication Mode section.

CAN Parameters

Baud Rate

This parameter refers to the CAN operating baud rate. The valid CAN baud rates are listed in the UART and CAN baud rate code table.

Transmit Frame Type

This parameter is available only in the transparent conversion mode and transparent conversion with ID mode. It indicates the type of CAN frame that is sent, including standard frame and extended frame.

Transmit ID

This parameter is available only in the transparent conversion mode. The ID of the CAN frames is determined by configuration. If the CAN frame to be transmitted is a standard frame, the CAN frame ID ranges from 0x000 to 0x7FF. If the CAN frame to be transmitted is an extended frame, the CAN frame ID ranges from 0x00000000 to 0x1FFFFFFF. From left to right, the frame IDs are ID3, ID2, ID1 and ID0, where ID3 is the highest byte. If a standard frame with a frame ID 0x0123 is sent during transparent conversion, the Transmit ID parameter should be set to 0x00000123.

Receive Filtering Enable

When the receive filtering is enabled, the module CAN interface filters the received CAN frames according to the filtering mode, mask code and acceptance code. When the receive filtering is disabled, the module CAN interface will receive all CAN frames on the bus.

Receive Filtering Mode

The receive filtering is divided into extended frame filtering and standard frame filtering.

Mask Code and Acceptance Code

The mask code and acceptance code are used together to implement the desired filtering mode.

When a bit of the mask code is 1, only when the corresponding bit of both the received CAN frame ID and the acceptance code are the same will this CAN frame ID bit be accepted. Otherwise, it will be ignored.

When a bit of the mask code is 0, the corresponding bit of the received CAN frame ID will be accepted whether it is the same as the corresponding bit of the acceptance code or not.

Mask Bit	Acceptance Bit	Frame ID	Receve or Not
0	X	X	Receive
1	0	0	Receive
1	0	1	Not receive
1	1	0	Not receive
1	1	1	Receive

Truth Table of Mask Code and Acceptance Code

- Example 1

  The module is configured to filter the extended frames using a mask code of 0x1FFFFFFF and an acceptance code of 0x00002345.

  Mask code: 0b 0001 1111 1111 1111 1111 1111 1111 1111

  Acceptance code: 0b 0000 0000 0000 0000 0010 0011 0100 0101

  Because all bits of the mask code are 1, only the extended frame whose frame ID is the same as the acceptance code can be received, which means only the extended frame with a frame ID of 0x00002345 can be received by the module.

- Example 2

  The module is configured to filter the standard frames using a mask code of 0x7F0 and an acceptance code of 0x345.

  Mask code: 0b 0111 1111 0000

  Acceptance code:0b 0011 0100 0101

  All standard frames with an ID of 0b 0011 0100 XXXX can be received, which means the standard frames with an ID in the range of 0x340~0x34F can be received, where X represents 0 or 1.

- Example 3

  The module is configured to filter the standard frames using a mask code of 0x1312 and an acceptance code of 0x0301.

  Mask code: 0b 0001 0011 0001 0010

  Acceptance code: 0b 0000 0011 0000 0001

  Because the module is configured to filter standard frames and the standard frame ID ranges from 0x000 to 0x7FF, the ID bits out of the range will not be compared with the acceptance code. Therefore, any standard frame on the bus with a frame ID of 0b 0X11 XXX0 XX0X can be received, where X represents 0 or 1.

- Example 4

  The module is configured to filter standard frames, the mask code is 0x00 00 00 00 and the acceptance code is 0x0301. Since all bits of the mask code are 0, the standard frames on the bus will be received regardless of their frame ID. This means the module will receive all standard frames on the bus, which is the same as the situation that the receive filtering is disabled.

Write Configuration Parameters

To configure the conversion function for the corresponding mode, users need to send the following command frame consisting of multiple bytes in the fixed frame format. After the configuration has been completed, the configuration data is saved even when the power is off. To change the conversion function, users can reconfigure the module. The configuration command is introduced in the following.

Write Configuration Command Frame Format

Frame Start	Command Code	Data Length	Data Field	Checksum
2 bytes	1 byte	1 byte	60 bytes	1 byte
0xF7, 0xF8	0x01	0x3C	Defined in the following table	XOR result of all preceding bytes

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x01

Data Length: 1 byte, indicating the number of data bytes in the data field, fixed at 60 (0x3C)

Data Field: 60 bytes, configuration information

Checksum: 1 byte, indicating the XOR result of all preceding bytes

No.	Parameter	Data Range	Default Value	Description
0	UART buad rate	0x00~0x10	0x0C (128kbps)	To change the UART baud rate, refer to the UART and CAN baud rate code table.
1	UART data bit	0x08	0x08	Fixed at 8 data bits
2	UART stop bit	0x01	0x01	Fixed at 1 stop bit
3	UART parity bit	0x00	0x00	No parity function
4	CAN baud rate	0x00~0x0F	0x08 (125kbps)	To change the CAN baud rate, refer to the UART and CAN baud rate code table.
5~7	Reserved bytes	—	0x00 00 00	Not used, invalid
8	CAN receive filtering enable	0x00/0x01	0x00	0x00: disable 0x01: enable
9	Receive filtering mode	0x08/0x00	0x08	0x00: only filter standard frames 0x08: only filter extended frames
10~13	Mask code	0x00 00 00 00 ~ 0xFF FF FF FF	0xFF FF FF FF	
14~17	Acceptance code 0	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
18~21	Acceptance code 1	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
22~25	Reserved bytes	—	0x00 00 00 00	Not used, invalid
26~29	Acceptance code 2	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
30~33	Acceptance code 3	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
34~37	Acceptance code 4	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
38~41	Acceptance code 5	0x00 00 00 00 ~ 0xFF FF FF FF	0x00 00 00 00	
42	Conversion mode	0x01~0x03	0x02	0x01: transparent conversion 0x02: transparent conversion with ID 0x03: custom protocol conversion
43	Conversion direction	0x00~0x02	0x00	0x00: bi-direction 0x01: only SPI/UART to CAN 0x02: only CAN to SPI/UART
44	UART frame interval	0x02~0x0A	0x02 (2 characters)	UART frame interval =10×n/f_{CLK_UART}, f_{CLK_UART}=1200bps~128kbps

No.	Parameter	Data Range	Default Value	Description
45	CAN fram information converted to serial frame	0x00/0x01	0x00	0x00: disable 0x01: enable This parameter is only used for transparent conversions.
46	CAN frame ID converted to serial frame	0x00/0x01	0x00	0x00: disable 0x01: enable This parameter is only used for transparent conversions.
47	CAN transmit frame type	0x00/0x08	0x08	0x00: standard frame 0x08: extended frame This parameter is only used for transparent conersions and transparent conversions with ID.
48~51	CAN transmit ID	0x00 00 00 00 ~ 0x1F FF FF FF	0x00 00 00 00	This parameter is only used for transparent conversions and consists of ID3~ID0. Standard frame: 0x000~0x7FF valid Extended frame: 0x00000000~0x1FFFFFFF valid
52	Length of CAN ID in serial frame	0x01~0x04	0x04	In bytes. This parameter is only used for transparent conversions with ID.
53	Start address of CAN ID in serial frame	0x00~0x07	0x00	In bytes. This parameter is only used for transparent conversions with ID.
54	Frame header	0x00~0xFF	0x40	One byte, user-defined
55	Frame tail	0x00~0xFF	0x1A	One byte, user-defined
56	Frame count to trigger feedback	0x04~0x0B	0x07	In the unit of one CAN frame, this parameter is only used in the SPI-to-CAN mode.
57	Time to trigger feedback	0x01~0xFF	0x05	In the unit of 100ms, this parameter is only used in SPI-to-CAN mode.
58	CPOL	0x00	0x00	Fixed
59	CPHA	0x00	0x00	Fixed

Data Field Definition of Write Configuration Command Frame

No.	Code	Serial Port Baud Rate (bps)	CAN Baud Rate (bps)
1	0x01	—	5k
2	0x02	57600	10k
3	0x03	38400	20k
4	0x04	19200	40k
5	0x05	14400	50k
6	0x06	9600	80k
7	0x07	4800	100k
8	0x08	2400	125k
9	0x09	1200	200k
10	0x0A	—	250k
11	0x0B	—	400k
12	0x0C	128000	500k
13	0x0D	—	666k
14	0x0E	—	800k
15	0x0F	—	1M
16	0x10	—	—

UART and CAN Baud Rate Code Table

Write Configuration Command Response Frame Format

After the module receives the Write Configuration Command frame, it will update the current configuration and send a response frame to the master no matter the update is successful or not. The format of the write configuration response frame is shown in the following table.

Frame Start	Command Code	State Code	Checksum
2 bytes	1 byte	1 byte	1 byte
0xF7, 0xF8	0x01	0x13 or 0x07	XOR result of all preceding bytes

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x01

State Code: 0x13 means configuration succeeded, 0x07 means configuration failed

Checksum: 1 byte, indicating the XOR result of all preceding bytes

Obtain Hardware Information Command Frame Format

Before configuring the module, this command is used to obtain the hardware information so as to verify whether the hardware information is correct or not. The specific command frame format is shown in the following table.

Frame Start	Command Code	Data Length	Data Field	Checksum
2 bytes	1 byte	1 byte	4 bytes	1 byte
0xF7, 0xF8	0x02	0x04	0x0A, 0x15, 0x12, 0x03	0x07

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x02

Data Length: 1 byte, indicating the number of data bytes in the data field, fixed at 4 (0x04)

Data Field: 4 bytes, product hardware identification information

Checksum: 1 byte, indicating the XOR result of all preceding bytes, 0x07

Hardware Information Response Frame Format

Frame Start	Command Code	State Code	Checksum
2 bytes	1 byte	1 byte	1 byte
0xF7, 0xF8	0x02	0x13 or 0x07	XOR result of all preceding bytes

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x02

State Code: 0x13 means matching succeeded, 0x07 means matching failed

Checksum: 1 byte, indicating the XOR result of all preceding bytes

Read Configuration Command Frame Format

This command is used to read the configuration parameters and its frame format is shown in the following table.

Frame Start	Command Code	Data Length	Data Field	Checksum
2 bytes	1 byte	1 byte	0 bytes	1 byte
0xF7, 0xF8	0x03	0x00	No data	0x0C

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x03

Data Length: 1 byte, fixed at 0x00 because there is no data for this command

Data Field: 0 bytes, no data

Checksum: 1 byte, indicating the XOR result of all preceding bytes, 0x0C

Read Configuration Command Response Frame Format

After receiving the Read Configuration Command frame, the module will send the configuration parameters to the main control MCU with a response frame. The response frame format is shown below.

Frame Start	Command Code	Data Length	Data Field	Checksum
2 bytes	1 byte	1 byte	60 bytes	1 byte
0xF7, 0xF8	0x03	0x3C	Refer to the write configuration frame definition	XOR result of all preceding bytes

Frame Start: 2 bytes, 0xF7 and 0xF8 in sequence

Command Code: 1 byte, fixed at 0x03

Data Length: 1 byte, fixed at 0x3C

Data Field: 60 bytes, refer to the write configuration command frame section

Checksum: 1 byte, indicating the XOR result of all preceding bytes

Configuration Steps

Configure via SPI

1. Pull high the MODE pin to select the SPI mode and pull low the CFG pin to enter the configuration state. Next, pull low the RST pin for at least 100μs to reset the module. Then pull high the RST pin and wait for at least 55ms for the module to actually enter the configuration mode.
2. Pull high both CTL0 and CTL1 pins to make the module enter the write command state, and wait for 50μs.
3. Pull low the SSEL pin then the SPI master sends the configuration command frame. After this pull high the SSEL pin and detect the INT pin state. If the INT pin is low, pull high CTL0 and pull low CTL1 to make the module enter the read state. Wait for at least 50μs then pull low the SSEL pin again, after which the master can send 5 invalid data bytes to the module to read the response frame, and then pull high the SSEL pin.
4. The master receives the response frame and determines whether the configuration is successful or not.
5. After the configuration command has been successfully sent, pull high the CFG pin to make the module enter the SPI-to-CAN data transmission mode. Pull low the RST pin for at least 100μs, then pull high the RST pin to reset the module and wait for at least 55ms, then the configuration command will be actually written into the module. After this data conversion and transmission can start. Pull high both CTL0 and CTL1 pins, then the master writes data to the module via SPI. Pull high CTL0 and pull low CTL1, then the master reads module data via SPI.
6. Steps 1~4 are also applicable for the Read Configuration command and Obtain Hardware Information command.

Configure via UART

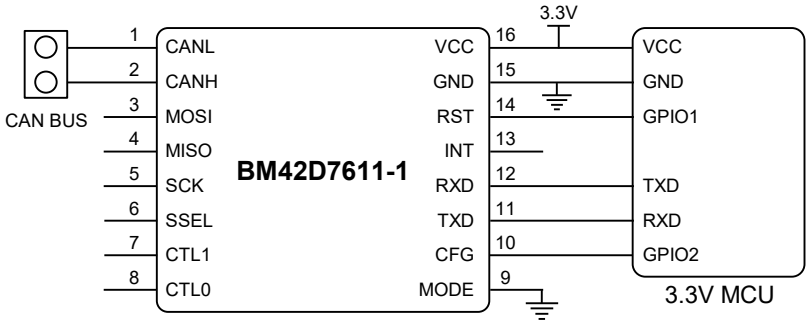
It should be noted that when the CFG pin is pulled low to enter the configuraiton mode, the serial port baud rate is fixed at 9600bps.

1. Pull low the MODE pin to enter the UART mode and pull low the CFG pin to enter the configuration state. Pull low the RST pin for at least 100μs, then pull high RST and wait for 55ms for the module reset to complete.
2. Now the master can send the configuration command frame. The module will actively send a response frame to the master after receving the configuration command frame.
3. The master receives the response frame and determines whether the configuration is successful or not.

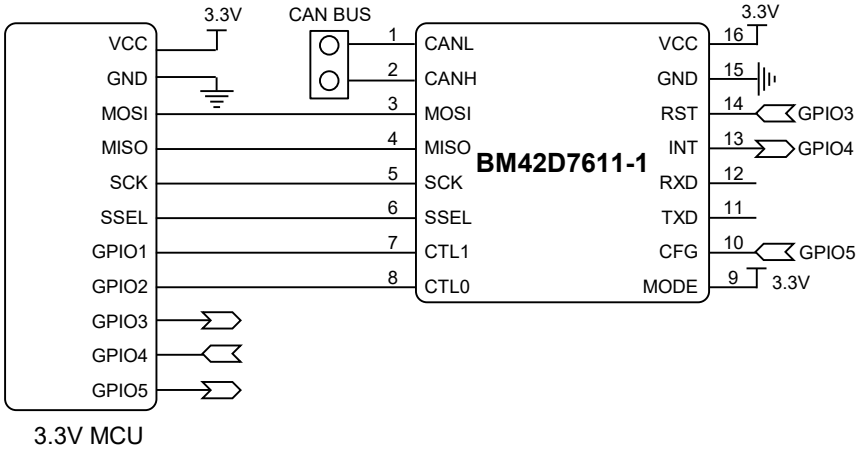
4. After the configuration is successful, pull high the CFG pin, pull low the RST pin and wait for at least 100μs, then pull high the RST pin and wait for 55ms for the module to reset and write the configuration information into the corresponding configuration port. After this data conversion and transmission can start.

Application Circuits

Hardware Circuits



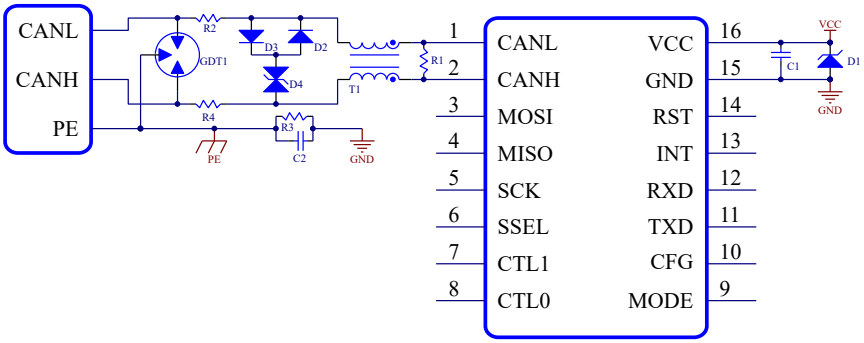
UART to CAN



SPI to CAN

Port Protection Circuit

When the module is used in harsh environments such as high voltage power or lightning strike environments, it is recommended to add a protection circuitry to the CANH/CANL end to prevent damage to the module. The following figure provides a civil protection circuit design for a large number of lightning surges, and the parameter description table lists a set of recommended circuit parameters. Users can adjust the parameters according to the actual situations.

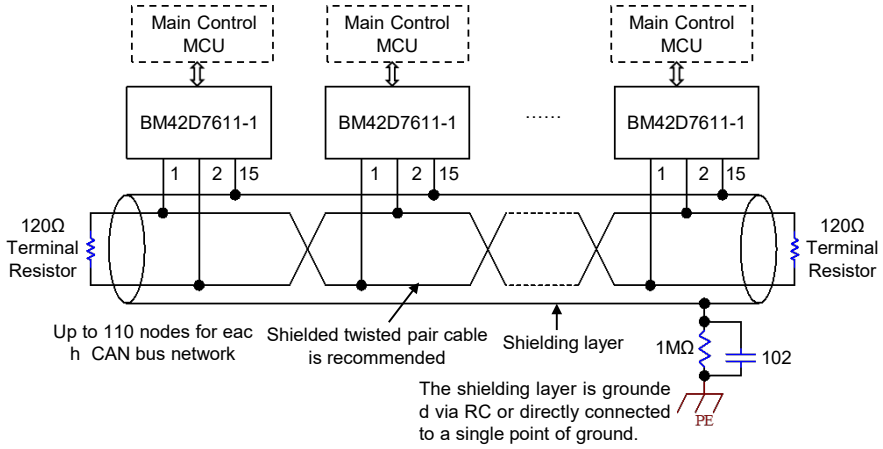


Parameter description:

Component	Recommended Parameter
C1	10μF, 35V
D1	SMBJ5.0A
R1	120Ω
R3	1MΩ
C2	1nF, 2kV
R2, R4	2.7Ω/2W
D2, D3	1N4007
D4	SMBJ30CA
GDT1	B3D090L
T1	ACM2520-301-2P

Recommended Networking Method

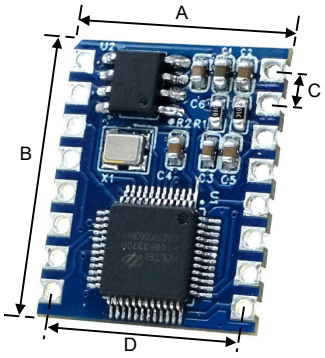
The CAN bus usually uses straight wiring and the number of notes that can be connected to each network is up to 110. The longest communication distance is10km for general-purpose modules when the CAN baud rate is 5kbps.



Layout Description

The CANL and CANH pins should use differential signal routing to improve anti-interference performance.

Dimensions



Symbol \ Unit	mm	inch
A (Product Length)	17.5	0.7
B (Product Width)	18.9	0.74
C (Pin Pitch)	2.54	0.1
D (Pin Pitch)	15.24	0.6

Note: Unit: mm (inch); tolerance: ±0.25 (±0.010).

Reference Information

Revision History

Date	Author	Version	Modification Description
20241127	王靖鈞	V1.0	First version

Buy Online

Best Modules