CAN Multiple ID Transmit Receive Support – (F2837x/F28004x)

This document will briefly summarize how to configure a CAN mailbox to support multiple CAN identifiers from Simulink® using the *Embedded Coder® Hardware Support Package for Texas Instruments (TI) C2000™ Processors:*
https://www.mathworks.com/matlabcentral/fileexchange/43096-embedded-coder-support-package-for-texas-instruments-c2000-processors

The below demo models are supported for F2837xD and F28004x targets:

i. 'CAN_MultiID_Transmit_Receive_F2837xD.slx' tested on F28379D Launchpad.

ii. 'CAN_MultiID_Transmit_Receive_F28004x.slx' tested on F280049C Launchpad.

OVERVIEW

Currently, the 'eCAN Transmit' and the 'eCAN Receive' blocks in the 'Embedded Coder Support Package for Texas Instruments C2000 Processors' support only single message identifier i.e., using 'eCAN Transmit' block a mailbox can be configured to send a single CAN message identifier and using 'eCAN Receive' block a mailbox can be configured to receive only a single CAN message identifier. Due to this limitation, we are only able to send and receive up to 32 CAN message IDs. In real time applications, there may be more than 32 message IDs which need to be transmitted and received. Hence, this workaround is proposed to overcome this limitation.

Refer to the demo video on this example here:

https://www.youtube.com/watch?v=-bf1tFDmiUQ&t=88s

Note: This reference example is applicable only for the MATLAB releases R2021a and below.

This reference example demonstrates the below workflows on F2837xD and F28004x MCUs:

1. Use CAN Pack blocks with CAN Tx and transmit multiple message IDs from a single mailbox.
2. To configure a mailbox as CAN Rx mailbox and receive multiple message IDs from a single mailbox.
3. To enable the loop back mode for self-test.
4. To read the CAN message only when a new message is received by using interrupts.
5. To read the CAN IDs using Memory copy block.

DETAILED DESIGN:

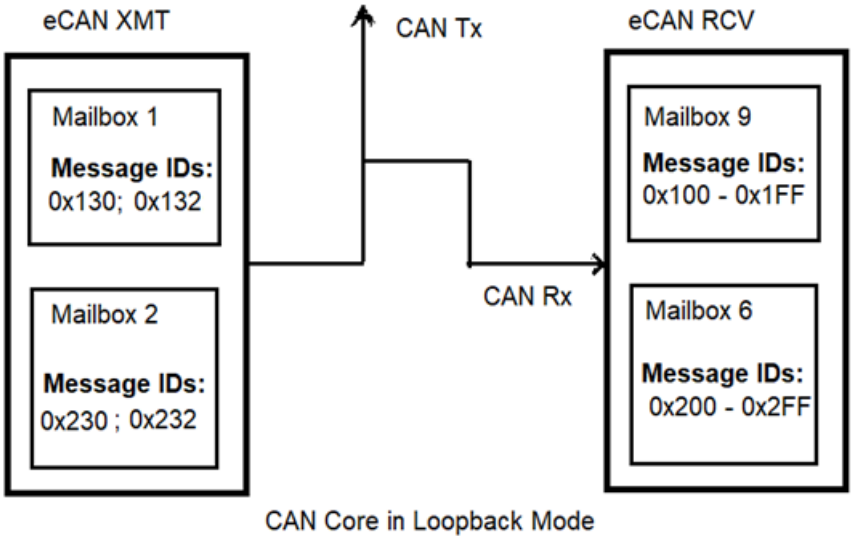
The below picture shows the CAN mailbox configuration to transmit and receive multiple CAN message IDs:

i. Mailbox 1 is used to transmit the message IDs 0x130 and 0x132 while mailbox 2 is used to transmit the message IDs 0x230 and 0x232 using CAN Pack blocks.

ii. Mailbox 9 is reconfigured to receive message IDs from 0x100 to 0x1FF while mailbox 6 is reconfigured to receive message IDs from 0x200 to 0x2FF.

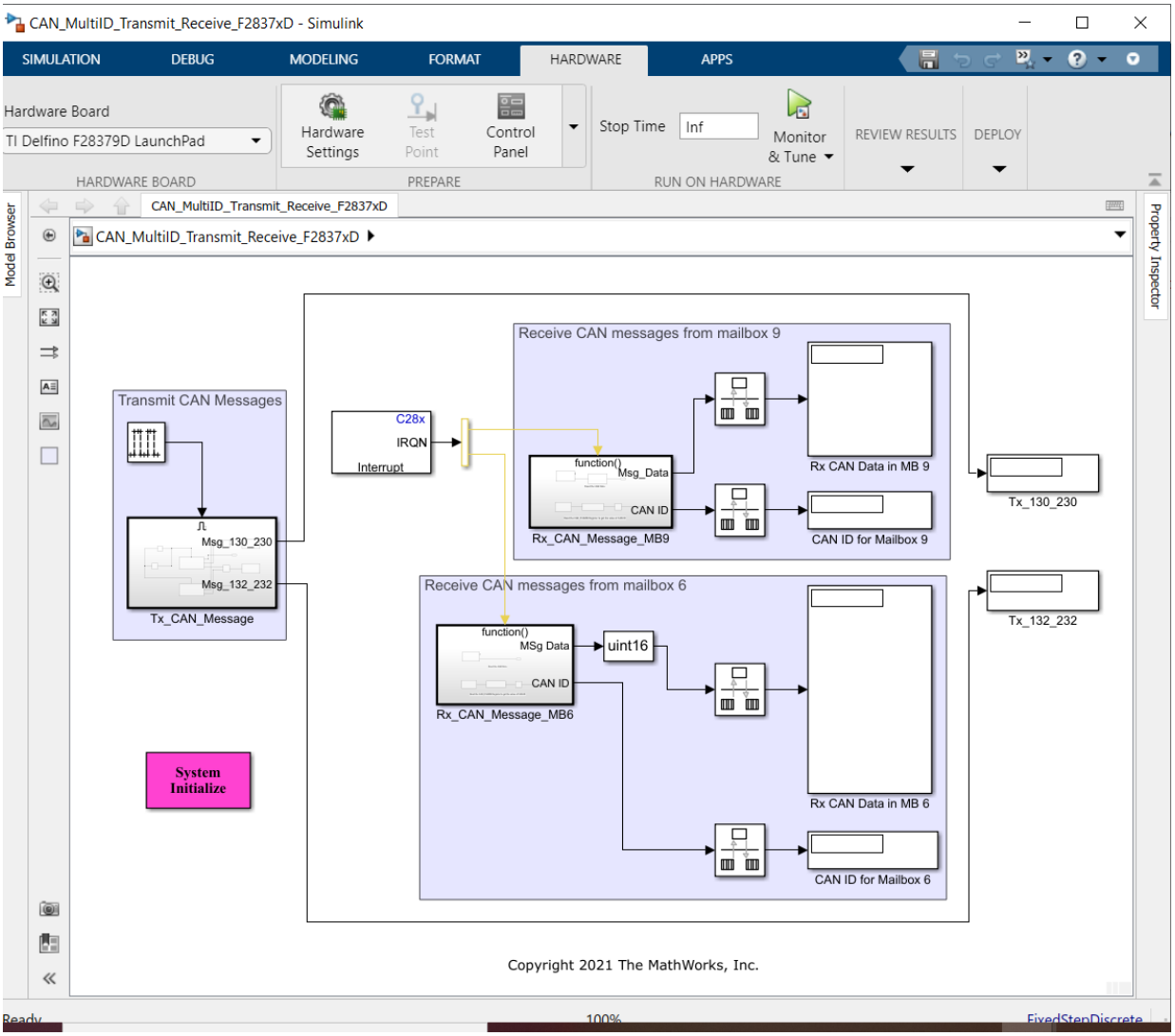
iii. The loopback mode is enabled to perform the self-test if the transmitted message IDs 0x130 and 0x132 are received back by mailbox 9 and the message IDs 0x230 and 0x232 are received back by mailbox 6.

iv. Enable the interrupt on CAN0INT interrupt line for mailbox 9 and on CAN1INT interrupt line for mailbox 6 to read the CAN message only when a new message is received by the mailboxes 9 and 6 respectively.



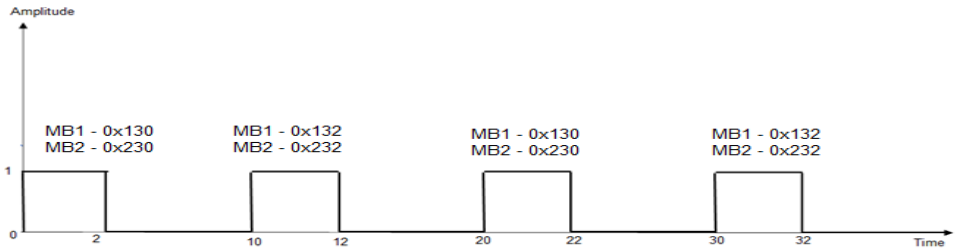
CAN Core in Loopback Mode

IMPLEMENTATION:



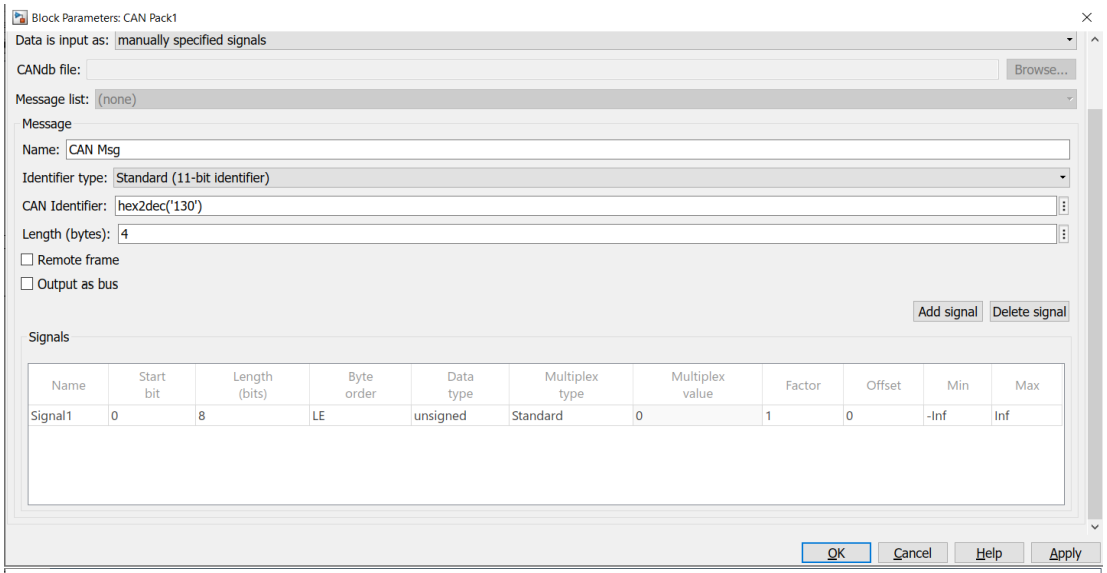
CAN Transmit:

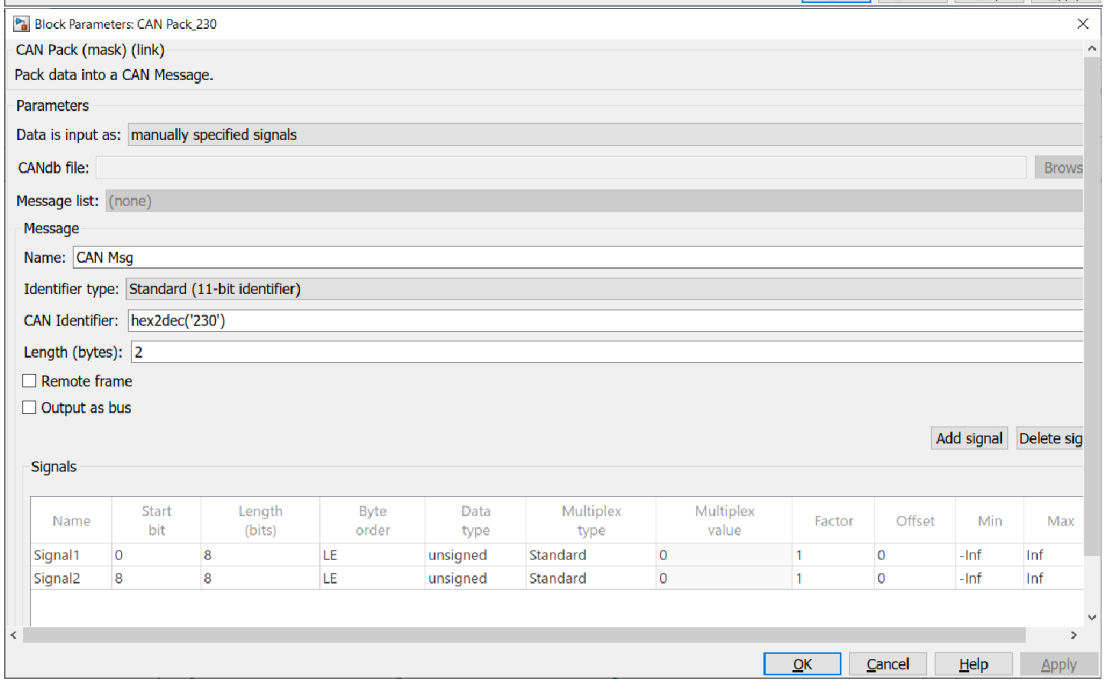
This example uses a 'Pulse Generator' block to enable the 'Tx_CAN_Message' subsystem to transmit the message IDs 0x130 and 0x132 from mailbox 1 and message IDs 0x230 and 0x232 from mailbox 2. The 'Pulse Generator' block is configured to enable the 'Tx_CAN_Message' subsystem for 3 continuous samples and disable the 'Tx_CAN_Message' subsystem for the remaining 7 samples in a period of 10 samples. During the first cycle when the 'Tx_CAN_Message' subsystem is enabled, it allows mailbox 1 to send message ID 0x130 and mailbox 2 to send message ID 0x230 for 3 continuous samples. During the second cycle when the 'Tx_CAN_Message' subsystem is enabled it allows mailbox 1 to send message IDs 0x132 and mailbox 2 to send message ID 0x232 for 3 continuous samples. During the third cycle when the 'Tx_CAN_Message' subsystem is enabled it allows mailbox 1 to send message IDs 0x130 and mailbox 2 to send message ID 0x230 for 3 continuous samples and so on. This is illustrated in the figure below:



Pulse Generator enabling Mailbox 1 and Mailbox 2 to transmit Msg IDs

Every time a new message is sent, the data of message IDs 0x130 and 0x230 is incremented by 1 and that of 0x132 and 0x232 is incremented by 2. The 'CAN Pack' block packs the messages using with the respective message ID before sending the message. The length of the messages for the identifiers 0x130 and 0x132 is configured as 4 bytes and that of the identifiers 0x230 and 0x232 is configured as 2 bytes as shown below:





While packing the data for message IDs 0x230 and 0x232, an additional signal (byte) is also packed whose values are 230 and 232 respectively to easily distinguish the message IDs at the receiving end.

Note: If a 'CAN Pack' block is used along with the 'eCAN Transmit' block, the Message identifier configured in the 'eCAN Transmit' block will be replaced with that of the 'CAN Pack' block. *And that is how sending multiple message ID's from the same Mailbox is achieved.* Since the ID gets replaced, the Message identifier parameter in the 'eCAN Transmit' blocks for mailboxes 1 and 2are given random values.

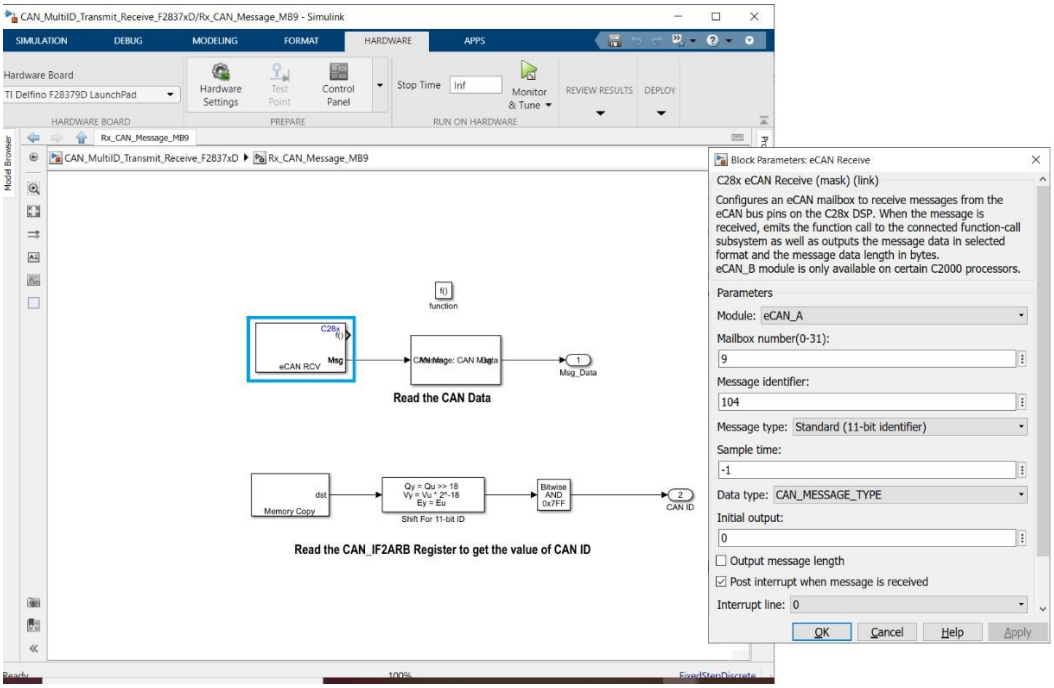
CAN Receive:

The 'eCAN Receive' block in the 'Rx_CAN_Message_MB9' subsystem is configured for the following:

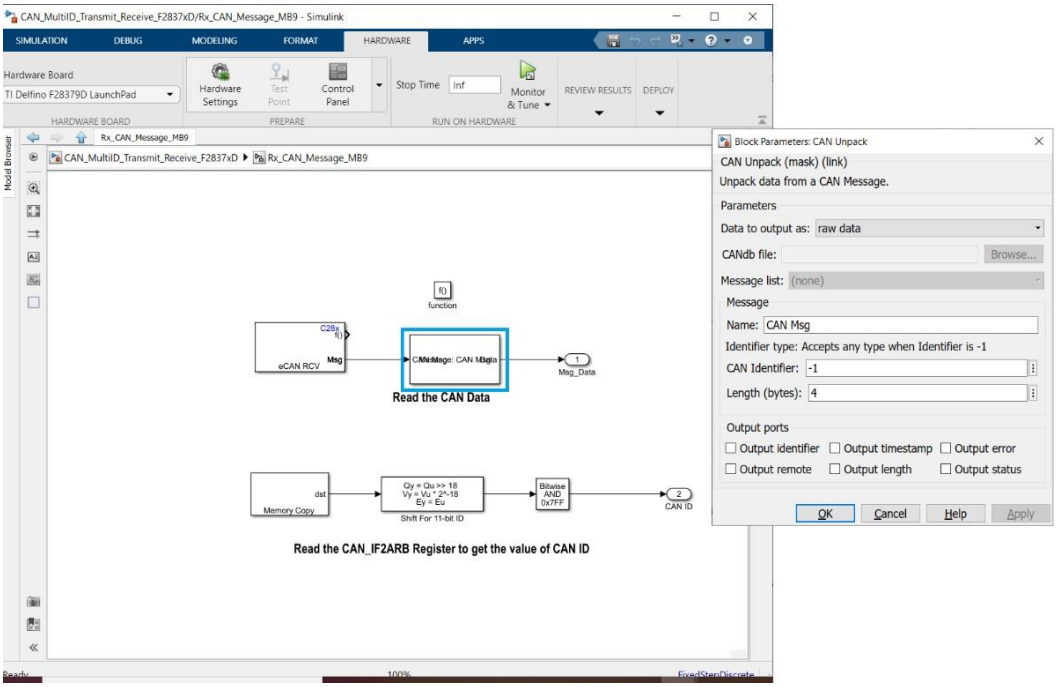
i. To read CAN messages from Mailbox 9 with output data type of CAN_MESSAGE_TYPE.
ii. To enable the interrupt on CAN0INT interrupt line when a new message is received.

Note: 1. The Mailbox 9 is configured to allow multiple identifiers using the 'System Initialize' block. So, the Message identifier and the Message type parameters are given random values as these parameters are re-configured in the custom code.

2. If there is no new message, the message data and the ID of the previous message is repeated.

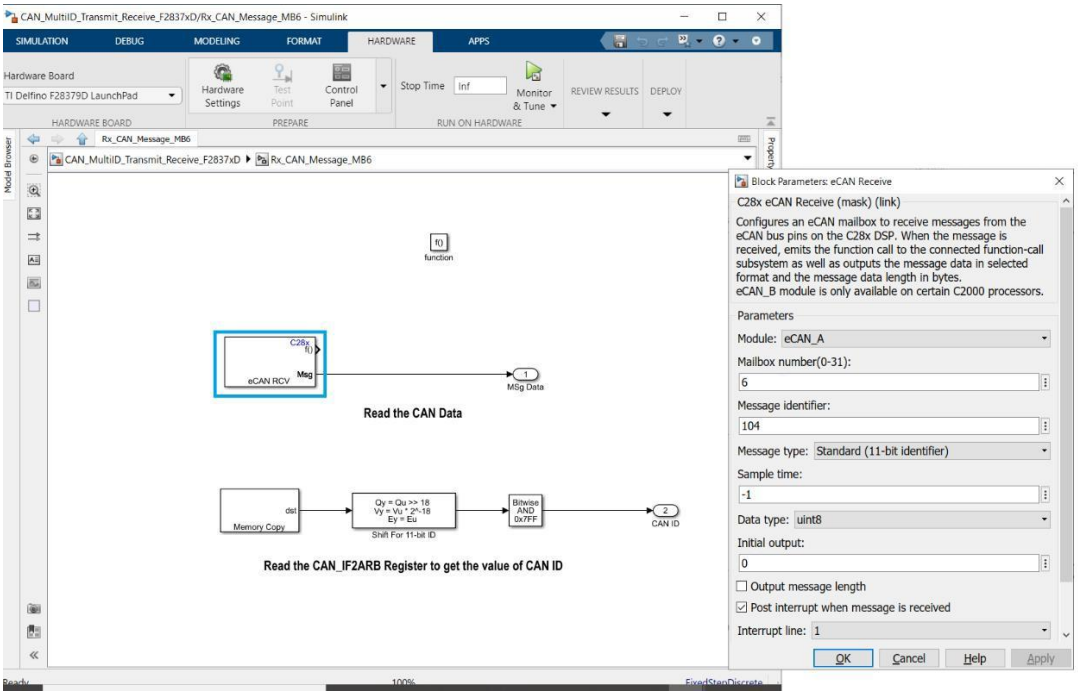


The 'CAN Unpack' block in the 'Rx_CAN_Message_MB9' subsystem is configured to allow any CAN Identifier whose message length is equal to 4 bytes as shown below



The ‘eCAN Receive’ block in the ‘Rx_CAN_Message_MB6’ subsystem is configured for the following:

i. To read CAN messages from Mailbox 6 with output data type unit8.
ii. To enable the interrupt on CAN1INT interrupt line when a new message is received.



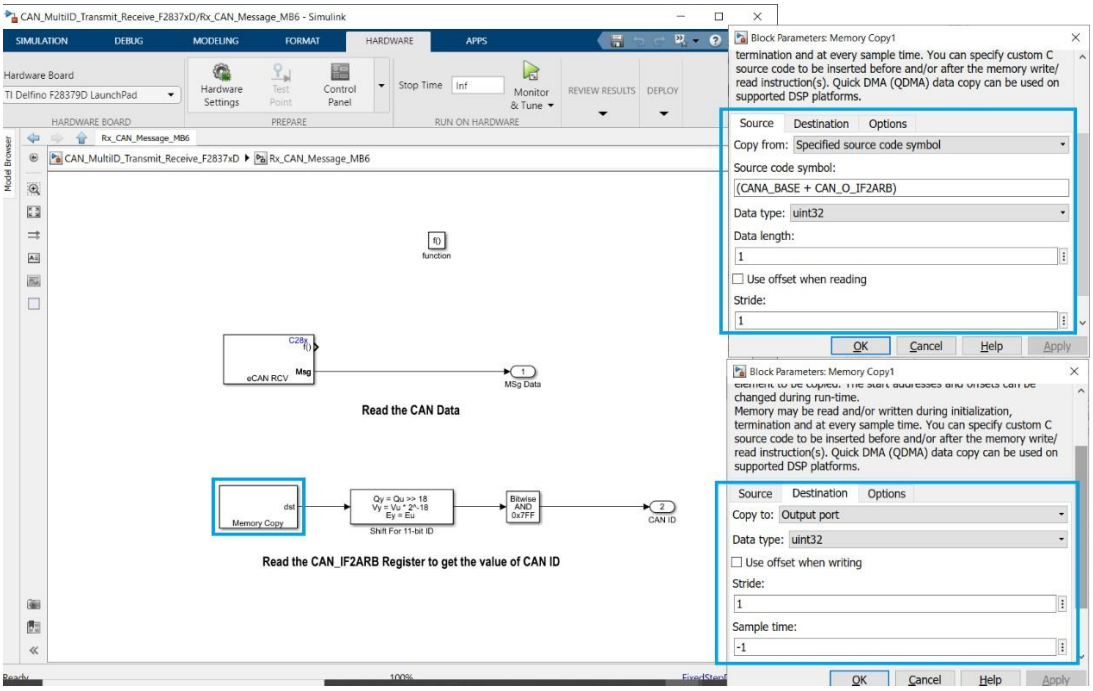
Note: 1. The Mailbox 6 is configured to allow multiple identifiers using the ‘System Initialize’ block. So, the Message identifier and the Message type parameters are given random values as these parameters are re-configured in the custom code.

2. If there is no new message, the message data and the ID of the previous message is repeated.

The ‘Memory Copy’ block is used to read the CAN_IF2ARB Register to display the CAN ID of the current message.

This block is configured to read data from the specified source code symbol (CANA_BASE + CAN_O_IF2ARB) as shown below:

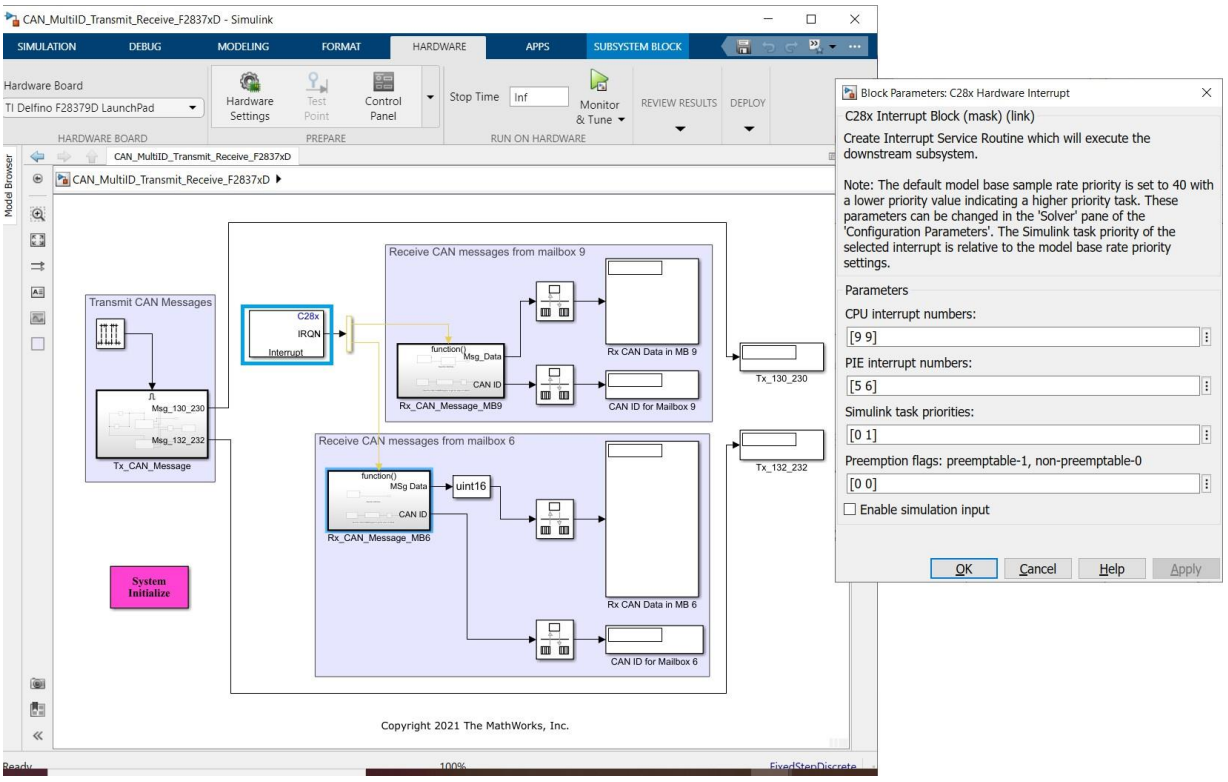
where, CANA_BASE is the start address of the CanaRegs and CAN_O_IF2ARB is the offset of the CAN_IF2ARB Register.



The ‘C28x Hardware Interrupt’ block is configured to generate hardware interrupts for the below events:

i. When a new message is received on mailbox 9, an interrupt is generated on the interrupt line CAN0INT, thereby invoking the ‘Rx_CAN_Message_MB9’ subsystem when the message is received.

ii. When a new message is received on mailbox 6, an interrupt is generated on the interrupt line CAN1INT, thereby invoking the ‘Rx_CAN_Message_MB6’ subsystem when the message is received.



Custom code settings:

The ‘System Initialize’ block reconfigures the mailboxes 9 and 6 with the below custom code to allow receiving multiple message IDs from 0x100 to 0x1FF and 0x200 to 0x2FF respectively.

For F2837xD:

The custom code for F2837xD is as shown below:

```
tCANMsgObject sRXCANMessage;

unsigned char ucRXMsgData[8]= { 0, 0, 0, 0, 0, 0, 0, 0 };

sRXCANMessage.ui32MsgID = 0x1AB;     // CAN message ID

sRXCANMessage.ui32MsgIDMask = 0x100;  // no mask needed for TX

sRXCANMessage.ui32Flags = MSG_OBJ_NO_FLAGS | MSG_OBJ_USE_ID_FILTER | MSG_OBJ_RX_INT_ENABLE;

sRXCANMessage.ui32MsgLen = sizeof(ucRXMsgData);// size of message is 4
```

sRXCANMessage.pucMsgData = ucRXMsgData;// ptr to message content

// Setup the message object being used to receive messages

CANMessageSet(CANA_BASE, 10, &sRXCANMessage, MSG_OBJ_TYPE_RX);

sRXCANMessage.ui32MsgID = 0x2CD; // CAN message ID

sRXCANMessage.ui32MsgIDMask = 0x200; // no mask needed for TX

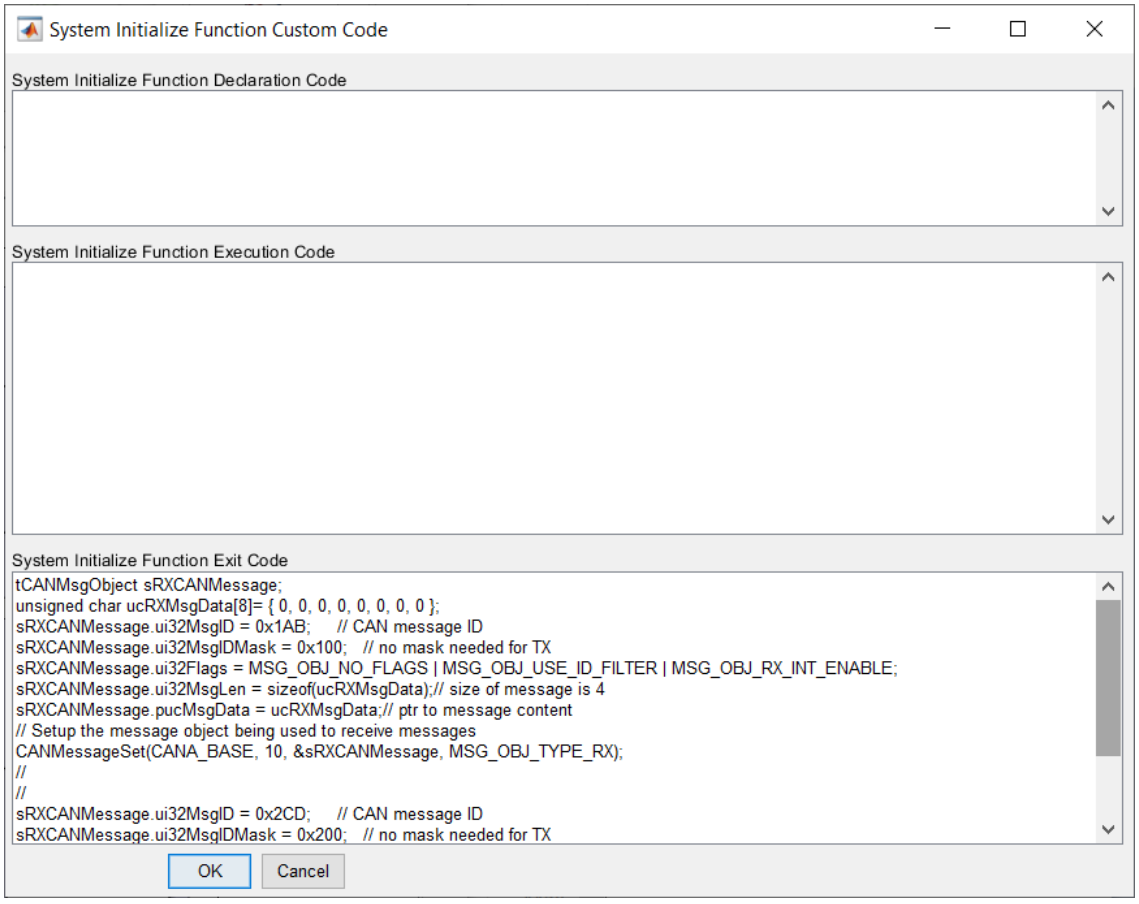
sRXCANMessage.ui32Flags = MSG_OBJ_NO_FLAGS | MSG_OBJ_USE_ID_FILTER | MSG_OBJ_RX_INT_ENABLE;

sRXCANMessage.ui32MsgLen = sizeof(ucRXMsgData);// size of message is 4

sRXCANMessage.pucMsgData = ucRXMsgData;// ptr to message content

// Setup the message object being used to receive messages

CANMessageSet(CANA_BASE, 7, &sRXCANMessage, MSG_OBJ_TYPE_RX);



The mailboxes 9 and 6 are reconfigured for the below changes using the custom code above:

i. The mailbox 9 is reconfigured with message identifier 0x1AB and mailbox 6 is reconfigured with message identifier 0x2CD.

ii. The message identifier mask value of 0x100 is used for mailbox 9 to allow messages from 0x100 to 0x1FF and that of 0x200 is used for mailbox 6 to allow messages from 0x200 to 0x2FF.

iii. The below flags are enabled for both the mailboxes 9 and 6:
MSG_OBJ_USE_ID_FILTER - To enable filtering based on the identifier mask specified by the message identifier mask value.
MSG_OBJ_RX_INT_ENABLE - To enable interrupt on receipt.

iv. The message length of 4 bytes.

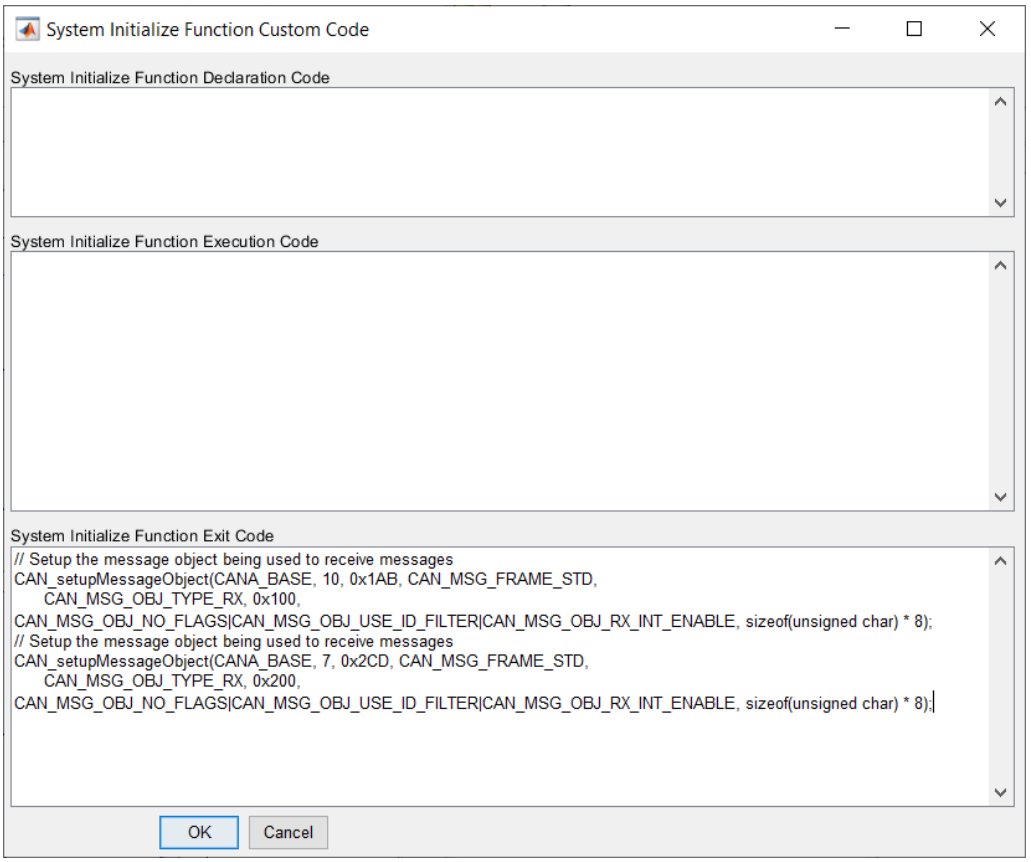
For F28004x:

The custom code for F28004x is as shown below:

```
// Setup the message object being used to receive messages

CAN_setupMessageObject(CANA_BASE,10,0x1AB,CAN_MSG_FRAME_STD,CAN_MSG_OBJ_TYPE_
RX,0x100,
CAN_MSG_OBJ_NO_FLAGS|CAN_MSG_OBJ_USE_ID_FILTER|CAN_MSG_OBJ_RX_INT_ENABLE,
sizeof(unsigned char) * 8);

// Setup the message object being used to receive messages

CAN_setupMessageObject(CANA_BASE,7,0x2CD,CAN_MSG_FRAME_STD,CAN_MSG_OBJ_TYPE_R
X,0x200,
CAN_MSG_OBJ_NO_FLAGS|CAN_MSG_OBJ_USE_ID_FILTER|CAN_MSG_OBJ_RX_INT_ENABLE,
sizeof(unsigned char) * 8);
```

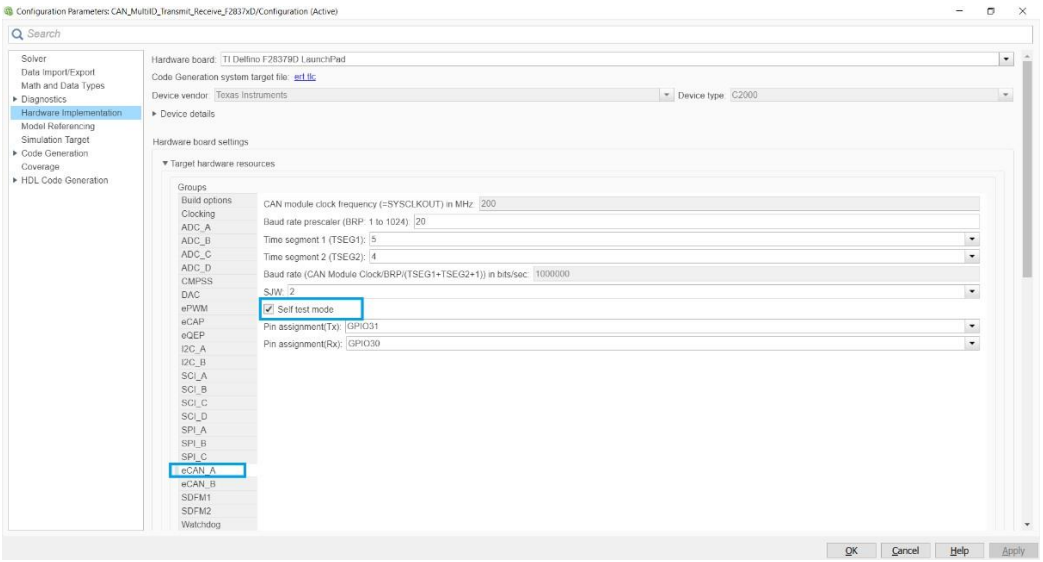
The mailboxes 9 and 6 are reconfigured for the below changes using the custom code above:

i. The mailbox 9 is reconfigured with message identifier 0x1AB and mailbox 6 is reconfigured with message identifier 0x2CD.
ii. The message identifier mask value of 0x100 is used for mailbox 9 to allow messages from 0x100 to 0x1FF and that of 0x200 is used for mailbox 6 to allow messages from 0x200 to 0x2FF.
iii. The below flags are enabled for both the mailboxes 9 and 6:
CAN_MSG_OBJ_USE_ID_FILTER - To enable filtering based on the identifier mask specified by the message identifier mask value.
CAN_MSG_OBJ_RX_INT_ENABLE - To enable interrupt on receipt.
iv. The message length of 8 bytes.



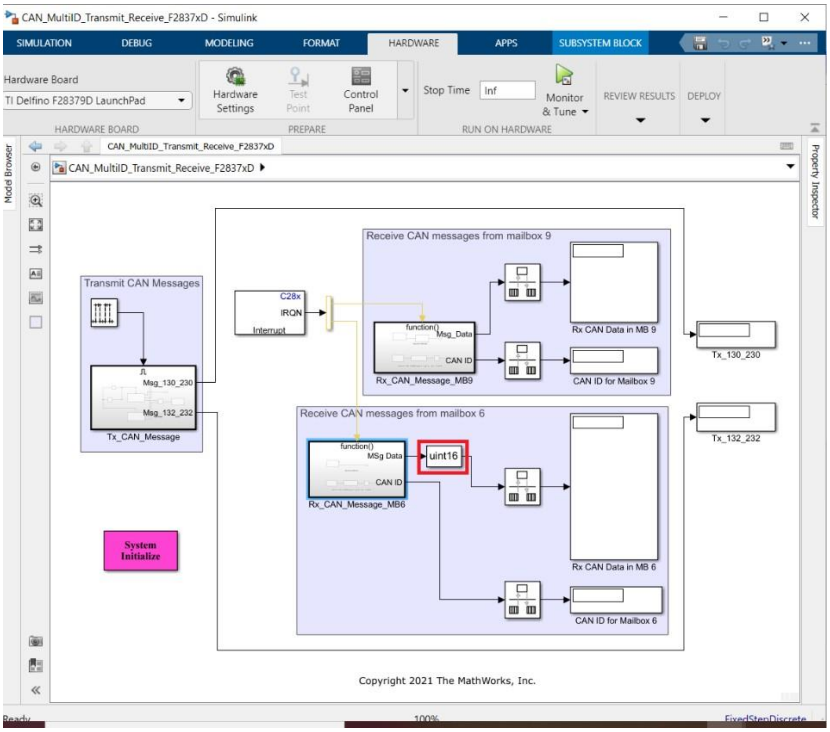
Configuration settings:

Enable the 'Self-test mode' parameter under Hardware Implementation ➔ Target hardware resources ➔ eCAN_A in the configuration parameters to enable the loop back mode for self-test.



Run the model in external mode

1. Connect the target device and run the model in external mode by clicking Hardware ➔ Monitor & Tune. The model will be built and loaded on the target.
2. Observe that the CAN message data and the respective CAN ID are displayed.



3. Note: The above data type conversion block is applicable only when the model is run on external mode as the 8-bit data is read as 16 bit in external mode.