Freescale MQX RTOS Example Guide

Low Power CAN wakeup example

This document explains the low power CAN wakeup example for imx6sx board; the example consists of two applications to demonstrate how to wake up MQX in low power mode (with CAN in stop mode) when it detects some CAN message from the CAN bus.

The example

The example contains two applications: one for low power CAN receiver and the other for CAN trigger. The CAN receiver runs in low power and support the whole SoC to enter SUSPEND mode (depending on the other core's status). And the CAN trigger could wake up the CAN receiver at 5 seconds interval.

Running the example

You need to connect two pairs of pins between two boards: CANH to CANH and CANL to CANL. And the CAN instance number is specified with "CAN_DEVICE" in can_wakeup.h.

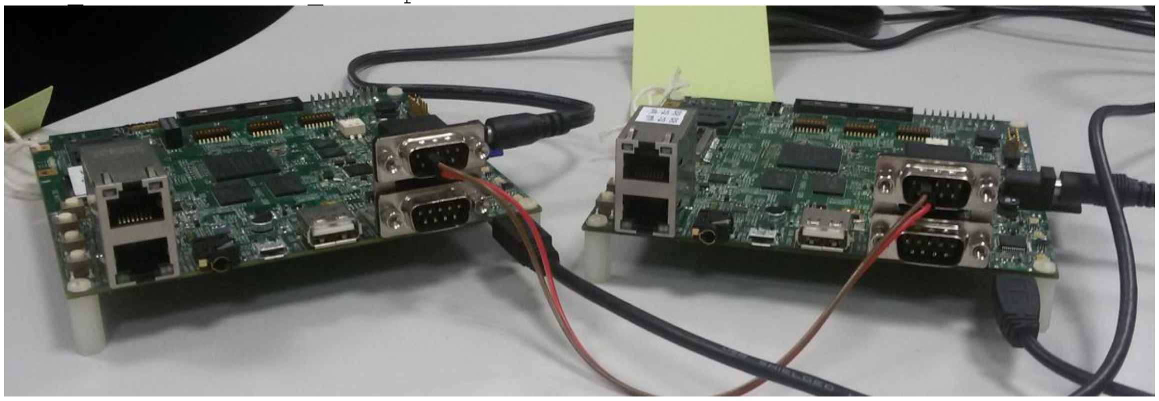


Figure 1
Connect instance 1 and 2

To run the example, the corresponding compiler and a terminal program are needed. First start the CAN receiver program, and when "Begin to receive CAN message" appears on the terminal, you can start the CAN trigger program.
To verify SoC wakeup from SUSPEND mode, you can run CAN receiver (on Cortex-M4) and Linux (on Cortex-A9) on same board, and run CAN trigger on the other one. After CAN receiver and CAN trigger start, make linux kernel suspend at any time to see the system (both A9 and M4) could be woken up by the CAN trigger.

Explaining the example

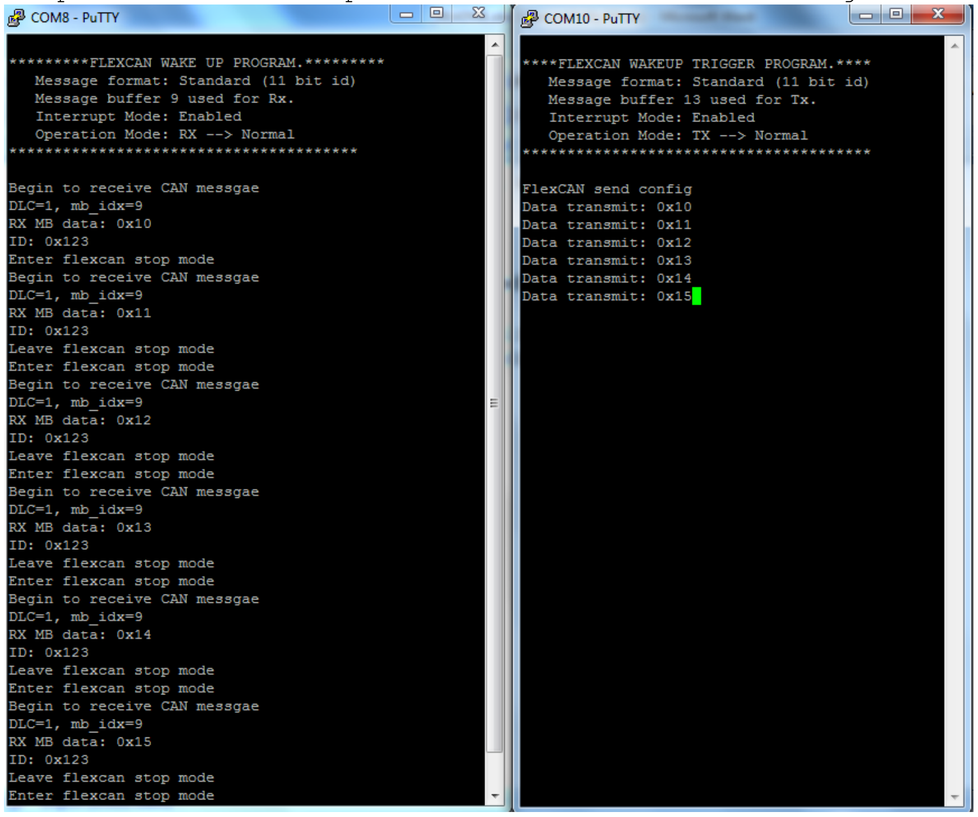
The CAN receiver has only one task:

- **Main_Task:** this task initializes and starts waiting on the CAN message in a loop. When succeeding to register wakeup source at Linux peer, it will set CAN device to STOP mode and make MQX enter low power mode. And after message is got, it will print out the message information and go into another loop. When registering wakeup source fails (because Linux peer is not running), it will not enter low power mode and just wait for the next CAN message.

The CAN trigger has only one task:

- Main_Task: This task initializes and starts sending out CAN message every 5 seconds in a loop.

The expected console output will like below when running with Linux:



COM8 - PuTTY

```
*********FLEXCAN WAKE UP PROGRAM.*********
   Message format: Standard (11 bit id)
   Message buffer 9 used for Rx.
   Interrupt Mode: Enabled
   Operation Mode: RX --> Normal
****************************************

Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x10
ID: 0x123
Enter flexcan stop mode
Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x11
ID: 0x123
Leave flexcan stop mode
Enter flexcan stop mode
Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x12
ID: 0x123
Leave flexcan stop mode
Enter flexcan stop mode
Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x13
ID: 0x123
Leave flexcan stop mode
Enter flexcan stop mode
Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x14
ID: 0x123
Leave flexcan stop mode
Enter flexcan stop mode
Begin to receive CAN messgae
DLC=1, mb_idx=9
RX MB data: 0x15
ID: 0x123
Leave flexcan stop mode
Enter flexcan stop mode
```

COM10 - PuTTY

```
****FLEXCAN WAKEUP TRIGGER PROGRAM.****
   Message format: Standard (11 bit id)
   Message buffer 13 used for Tx.
   Interrupt Mode: Enabled
   Operation Mode: TX --> Normal
****************************************

FlexCAN send config
Data transmit: 0x10
Data transmit: 0x11
Data transmit: 0x12
Data transmit: 0x13
Data transmit: 0x14
Data transmit: 0x15
```

Figure 2
Expected console output result