Application Report
SPRACF7–April 2018

Adding CAN-FD Tx and Rx to an Existing mmWave Project

Raghunandan Kamath

ABSTRACT

This application note describes the steps required to integrate the usage of the CAN-FD interface on the mmWave devices.

Contents

List of Figures

Trademarks

1 Initializing the Driver

The first step is to add code to include and initialize the CANFD driver. This driver is required for transmitting and receiving from the CANFD interface. The following is C code that initializes the CANFD driver. This tested code may be copied into the project.

```
#include <ti/drivers/canfd/canfd.h>
/**************************************************************************
 ************************** Global Definitions **************************
 **************************************************************************/
volatile uint32_t       gTxDoneFlag = 0, gRxDoneFlag = 0, gParityErrFlag = 0;
volatile uint32_t       gTxPkts = 0, gRxPkts = 0, gErrStatusInt = 0;
volatile uint32_t               iterationCount = 0U;
uint32_t                dataLength = 0U;
uint32_t                msgLstErrCnt = 0U;
uint32_t                gDisplayStats = 0;
uint8_t                 rxData[64U];
uint32_t                txDataLength, rxDataLength;
CANFD_MCANFrameType     frameType = CANFD_MCANFrameType_FD;
static void MCANAppInitParams(CANFD_MCANInitParams* mcanCfgParams);
CANFD_Handle                canHandle;
CANFD_MsgObjHandle          txMsgObjHandle;
CANFD_MCANMsgObjCfgParams   txMsgObjectParams;

/**************************************************************************
 ************************** CAN Driver Initialize Function **********************
 **************************************************************************/
void Can_Initalize(void)
{
  int32_t                    errCode = 0;
     int32_t                    retVal = 0;
     CANFD_MCANInitParams       mcanCfgParams;
  CANFD_MCANBitTimingParams   mcanBitTimingParams;
  CANFD_MCANMsgObjCfgParams   rxMsgObjectParams;
     CANFD_MsgObjHandle         rxMsgObjHandle;
     gTxDoneFlag = 0;
  gRxDoneFlag = 0;
/* Setup the PINMUX to bring out the XWR16xx CAN pins */
 Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINE14_PADAE, PINMUX_OUTEN_RETAIN_HW_CTRL,
PINMUX_INPEN_RETAIN_HW_CTRL);
 Pinmux_Set_FuncSel(SOC_XWR16XX_PINE14_PADAE, SOC_XWR16XX_PINE14_PADAE_CANFD_TX);

 Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PIND13_PADAD, PINMUX_OUTEN_RETAIN_HW_CTRL,
PINMUX_INPEN_RETAIN_HW_CTRL);
 Pinmux_Set_FuncSel(SOC_XWR16XX_PIND13_PADAD, SOC_XWR16XX_PIND13_PADAD_CANFD_RX);

 /* Configure the divide value for MCAN source clock */
 SOC_setPeripheralClock(gMmwMssMCB.socHandle, SOC_MODULE_MCAN, SOC_CLKSOURCE_VCLK, 4U, &errCode);
/* Initialize peripheral memory */
 SOC_initPeripheralRam(gMmwMssMCB.socHandle, SOC_MODULE_MCAN, &errCode);
 MCANAppInitParams (&mcanCfgParams);
 /* Initialize the CANFD driver */
 canHandle = CANFD_init(&mcanCfgParams, &errCode);
 if (canHandle == NULL)
 {
     System_printf ("Error: CANFD Module Initialization failed [Error code %d]\n", errCode);
      return ;
  }
  /* Configuring 1Mbps and 5Mbps as nominal and data bit-rate respectively
        Prop seg: 8
        Ph seg 1: 6
        Ph Seg2 : 5
        Sync jump: 1
        BRP(Baud rate Prescaler): 2

        Nominal Bit rate = (40)/(((8+6+5)+1)*BRP) = 1Mhz
```

```
            Timing Params for Data Bit rate:
            Prop seg: 2
            Ph seg 1: 2
            Ph Seg2 : 3
            Sync jump: 1
            BRP(Baud rate Prescaler): 1
            Nominal Bit rate = (40)/(((2+2+3)+1)*BRP) = 5Mhz
        */
        mcanBitTimingParams.nomBrp      = 0x2U;
        mcanBitTimingParams.nomPropSeg  = 0x8U;
        mcanBitTimingParams.nomPseg1    = 0x6U;
        mcanBitTimingParams.nomPseg2    = 0x5U;
        mcanBitTimingParams.nomSjw      = 0x1U;

        mcanBitTimingParams.dataBrp     = 0x1U;
        mcanBitTimingParams.dataPropSeg = 0x2U;
        mcanBitTimingParams.dataPseg1   = 0x2U;
        mcanBitTimingParams.dataPseg2   = 0x3U;
        mcanBitTimingParams.dataSjw     = 0x1U;

      /* Configure the CAN driver */
         retVal = CANFD_configBitTime (canHandle, &mcanBitTimingParams, &errCode);
        if (retVal < 0)
        {
            System_printf ("Error: CANFD Module configure bit time failed [Error code %d]\n",
            errCode);
            return ;
        }
       /* Setup the transmit message object */
        txMsgObjectParams.direction = CANFD_Direction_TX;
        txMsgObjectParams.msgIdType = CANFD_MCANXidType_29_BIT;
        txMsgObjectParams.msgIdentifier = 0xD1;
           txMsgObjHandle = CANFD_createMsgObject (canHandle, &txMsgObjectParams, &errCode);
           if (txMsgObjHandle == NULL)
           {
            System_printf ("Error: CANFD create Tx message object failed [Error code %d]\n",
            errCode);
            return ;
           }
        /* Setup the receive message object */
        rxMsgObjectParams.direction = CANFD_Direction_RX;
        rxMsgObjectParams.msgIdType = CANFD_MCANXidType_29_BIT;
        rxMsgObjectParams.msgIdentifier = 0xD1;

        rxMsgObjHandle = CANFD_createMsgObject (canHandle, &rxMsgObjectParams, &errCode);
        if (rxMsgObjHandle == NULL)
        {
            System_printf ("Error: CANFD create Rx message object failed [Error code %d]\n",
            errCode);
            return ;
        }
        }
    /**************************************************************************
     ******************* CAN Parameters initialize Function *****************
     *************************************************************************/

    static void MCANAppInitParams(CANFD_MCANInitParams* mcanCfgParams)
    {
        /*Intialize MCAN Config Params*/
        memset (mcanCfgParams, sizeof (CANFD_MCANInitParams), 0);

        mcanCfgParams->fdMode          = 0x1U;
        mcanCfgParams->brsEnable       = 0x1U;
        mcanCfgParams->txpEnable       = 0x0U;
        mcanCfgParams->efbi            = 0x0U;
        mcanCfgParams->pxhddisable     = 0x0U;
```

```
    mcanCfgParams->darEnable       = 0x1U;
    mcanCfgParams->wkupReqEnable   = 0x1U;
    mcanCfgParams->autoWkupEnable  = 0x1U;
    mcanCfgParams->emulationEnable = 0x0U;
    mcanCfgParams->emulationFAck   = 0x0U;
    mcanCfgParams->clkStopFAck     = 0x0U;
    mcanCfgParams->wdcPreload      = 0x0U;
    mcanCfgParams->tdcEnable       = 0x1U;
    mcanCfgParams->tdcConfig.tdcf  = 0U;
    mcanCfgParams->tdcConfig.tdco  = 8U;
    mcanCfgParams->monEnable       = 0x0U;
    mcanCfgParams->asmEnable       = 0x0U;
    mcanCfgParams->tsPrescalar     = 0x0U;
    mcanCfgParams->tsSelect        = 0x0U;
    mcanCfgParams->timeoutSelect   = CANFD_MCANTimeOutSelect_CONT;
    mcanCfgParams->timeoutPreload  = 0x0U;
    mcanCfgParams->timeoutCntEnable= 0x0U;
    mcanCfgParams->filterConfig.rrfe        = 0x1U;
    mcanCfgParams->filterConfig.rrfs        = 0x1U;
    mcanCfgParams->filterConfig.anfe        = 0x1U;
    mcanCfgParams->filterConfig.anfs        = 0x1U;
    mcanCfgParams->msgRAMConfig.lss         = 127U;
    mcanCfgParams->msgRAMConfig.lse         = 64U;
    mcanCfgParams->msgRAMConfig.txBufNum    = 32U;
    mcanCfgParams->msgRAMConfig.txFIFOSize  = 0U;
    mcanCfgParams->msgRAMConfig.txBufMode   = 0U;
    mcanCfgParams->msgRAMConfig.txEventFIFOSize         = 0U;
    mcanCfgParams->msgRAMConfig.txEventFIFOWaterMark    = 0U;
    mcanCfgParams->msgRAMConfig.rxFIFO0size             = 0U;
    mcanCfgParams->msgRAMConfig.rxFIFO0OpMode           = 0U;
    mcanCfgParams->msgRAMConfig.rxFIFO0waterMark        = 0U;
    mcanCfgParams->msgRAMConfig.rxFIFO1size             = 64U;
    mcanCfgParams->msgRAMConfig.rxFIFO1waterMark        = 64U;
    mcanCfgParams->msgRAMConfig.rxFIFO1OpMode           = 64U;
    mcanCfgParams->eccConfig.enable         = 1;
    mcanCfgParams->eccConfig.enableChk      = 1;
    mcanCfgParams->eccConfig.enableRdModWr  = 1;
    mcanCfgParams->errInterruptEnable   = 1U;
    mcanCfgParams->dataInterruptEnable  = 1U;
    mcanCfgParams->appErrCallBack       = MCANAppErrStatusCallback;
    mcanCfgParams->appDataCallBack      = MCANAppCallback;
}
```

2 Register Callbacks

2.1 Tx Complete and Rx Interrupt Callback

The application must implement a callback function to handle the transmit complete and receive interrupts.

```
static void MCANAppCallback(CANFD_MsgObjHandle handle, CANFD_Reason reason)
{
    int32_t                 errCode, retVal;
    uint32_t                id;
    CANFD_MCANFrameType     rxFrameType;
    CANFD_MCANXidType       rxIdType;

    if (reason == CANFD_Reason_TX_COMPLETION)
    {
        {
            gTxPkts++;
            gTxDoneFlag = 1;
            return;
        }
    }
    if (reason == CANFD_Reason_RX)
    {
        {
            /* Reset the receive buffer */
            memset(&rxData, 0, sizeof (rxData));
            dataLength = 0;

            retVal = CANFD_getData (handle, &id, &rxFrameType, &rxIdType, &rxDataLength,
&rxData[0], &errCode);
            if (retVal < 0)
            {
                System_printf ("Error: CAN receive data for iteration %d failed [Error
                code %d]\n", iterationCount, errCode);
                return;
            }
            if (rxFrameType != frameType)
            {
                System_printf ("Error: CAN received incorrect frame type Sent %d Received
                 %d for iteration %d failed\n", frameType, rxFrameType, iterationCount);
                return;
            }
            /* Validate the data */
            gRxPkts++;
            gRxDoneFlag = 1;
            return;
        }
    }
    if (reason == CANFD_Reason_TX_CANCELED)
    {
        {
            gTxPkts++;
            gTxDoneFlag = 1;
            gRxDoneFlag = 1;
            return;
        }
    }
}
```

2.2 Error and Status Interrupt Callback

The application must implement a callback function to handle error and status interrupts.

```
static void MCANAppErrStatusCallback(CANFD_Handle handle, CANFD_Reason reason,
CANFD_ErrStatusResp* errStatusResp)
{
       /*Record the error count */
    gErrStatusInt++;
    return;
}
```

3 CAN-FD Transmit

The CANFD can work both in the FD mode and classic CAN mode, based on the frame type used during the transmission.

The following code can be used to transmit CAN data, based on the frame type defined during the initialization and the length message.

```
if(frameType == CANFD_MCANFrameType_FD)
    {
        Task_sleep(1);
        while(len > 64U)
        {
retVal = CANFD_transmitData (txMsgObjHandle, msg_id, CANFD_MCANFrameType_FD, 64U, &txmsg[index],
&errCode);
                index = index + 64U;
                len = len - 64U;
                Task_sleep(1);
        }
retVal = CANFD_transmitData (txMsgObjHandle, msg_id, CANFD_MCANFrameType_FD, len, &txmsg[index],
&errCode);
    }
    else
    {
        while(len > 8U)
        {
   retVal = CANFD_transmitData (txMsgObjHandle, msg_id,
     CANFD_MCANFrameType_CLASSIC, 8U, &txmsg[index], &errCode);
            if (retVal < 0)
            {
                continue;
            }
            index = index + 8U;
            len = len - 8U;
           }
            retVal = CANFD_transmitData (txMsgObjHandle, msg_id, CANFD_MCANFrameType_CLASSIC,
len, &txmsg[index], &errCode);

        while(retVal < 0)
        {
retVal = CANFD_transmitData (txMsgObjHandle, msg_id, CANFD_MCANFrameType_CLASSIC, len,
&txmsg[index], &errCode);
        }
    }
```

4 Linking the CAN-FD Driver

The final step is to build the executable by linking with the CAN-FD drivers. If using a CCS project, the CAN-FD drivers can be added to the project's linker properties, as shown in Figure 1.

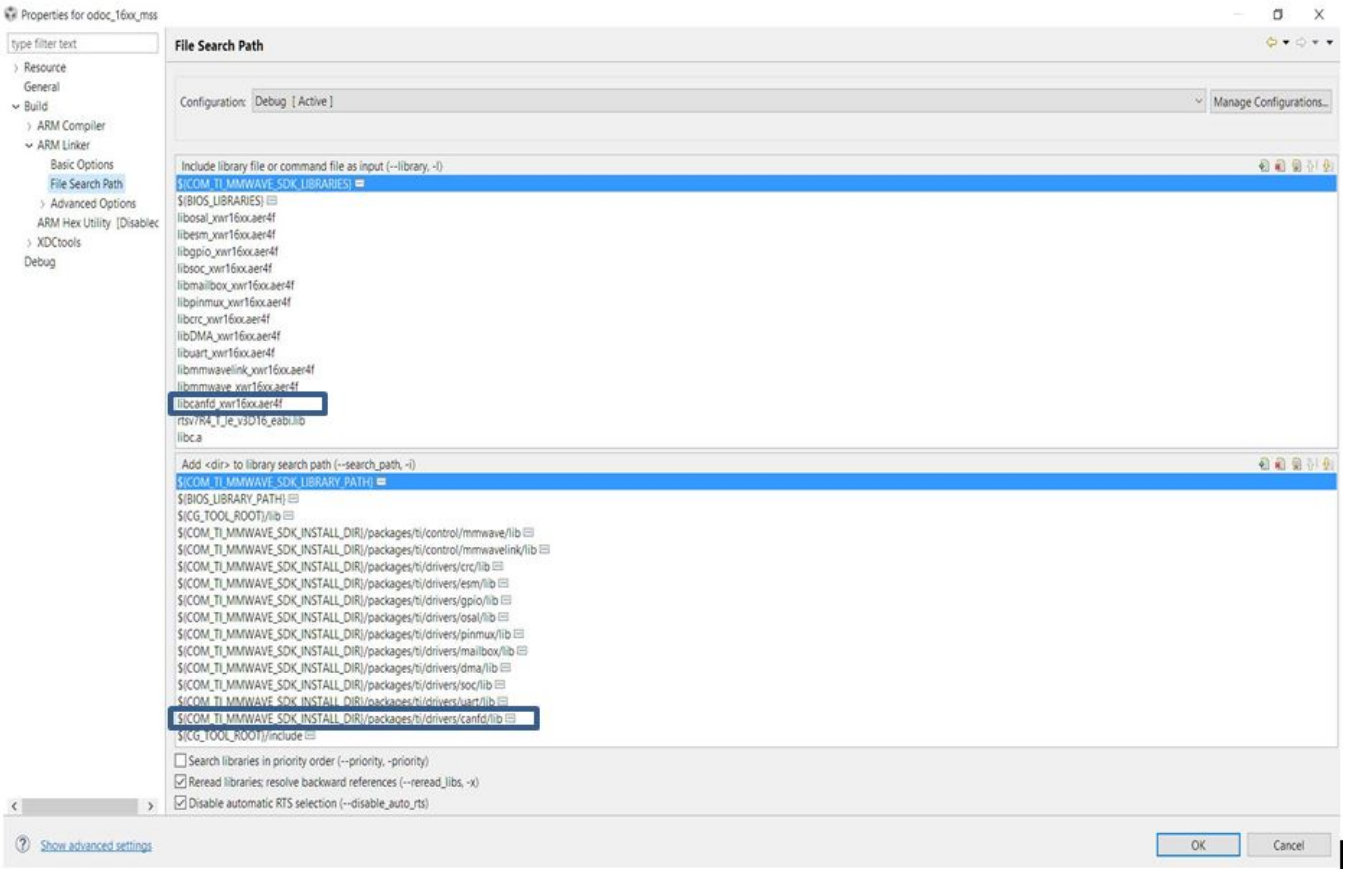

Figure 1. CCS Project Linker Properties

If using the makefile, perform the same procedure.

```
################################################################################
# Additional libraries which are required to build the DEMO:
################################################################################
MSS_MMW_DEMO_STD_LIBS = $(R4F_COMMON_STD_LIB)                                  \
            -llibpinmux_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)        \
            -llibdma_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)           \
            -llibcrc_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)           \
            -llibuart_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)          \
            -llibgpio_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)          \
            -llibmailbox_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)       \
            -llibmmwavelink_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)    \
            -llibmmwave_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)        \
            -llibcli_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)       \
            -llibcanfd_$(MMWAVE_SDK_DEVICE_TYPE).$(R4F_LIB_EXT)
MSS_MMW_DEMO_LOC_LIBS = $(R4F_COMMON_LOC_LIB)                              \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/pinmux/lib      \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/uart/lib        \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/dma/lib         \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/crc/lib         \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/gpio/lib        \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/mailbox/lib     \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/control/mmwavelink/lib  \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/control/mmwave/lib      \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/utils/cli/lib   \
            -i$(MMWAVE_SDK_INSTALL_PATH)/ti/drivers/canfd/lib
```

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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