Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 6-3: Tiva TM4C123G – CAN RX TX

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On Tiva TM4C123G CAN RX TX

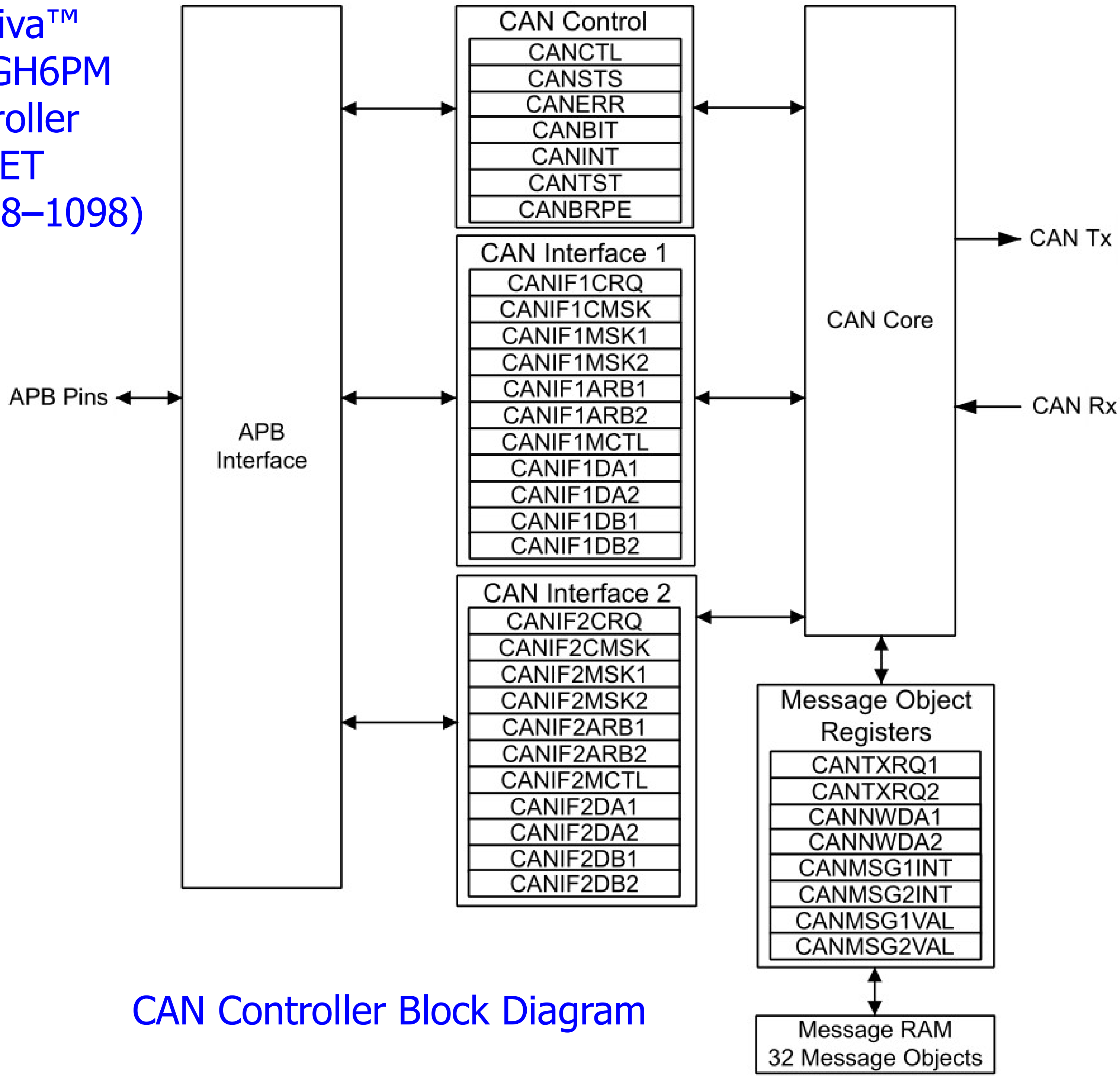
Objectives

The aims of this hands-on session are to

- develop a Tiva TM4C123G project
- implement a CAN (Controller Area Network) application using Tiva C Series LaunchPad EK-TM4C123GXL evaluation kit with TM4C123GH6PMI7 microcontroller
- configure, program, and test the CAN for receiving and transmitting data
- use of a CAN analyzer to evaluate the CAN communication
- build up the development knowledge of CAN applications

Hands-On Tiva TM4C123G CAN RX TX

Refer to Tiva™ TM4C123GH6PM Microcontroller DATA SHEET (page 1048–1098)



CAN Controller Block Diagram

CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	RW	0x0000.0001	CAN Control	1070
0x004	CANSTS	RW	0x0000.0000	CAN Status	1072
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	1075
0x00C	CANBIT	RW	0x0000.2301	CAN Bit Timing	1076
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	1077
0x014	CANTST	RW	0x0000.0000	CAN Test	1078
0x018	CANBRPE	RW	0x0000.0000	CAN Baud Rate Prescaler Extension	1080
0x020	CANIF1CRQ	RW	0x0000.0001	CAN IF1 Command Request	1081
0x024	CANIF1CMSK	RW	0x0000.0000	CAN IF1 Command Mask	1082
0x028	CANIF1MSK1	RW	0x0000.FFFF	CAN IF1 Mask 1	1085
0x02C	CANIF1MSK2	RW	0x0000.FFFF	CAN IF1 Mask 2	1086
0x030	CANIF1ARB1	RW	0x0000.0000	CAN IF1 Arbitration 1	1088
0x034	CANIF1ARB2	RW	0x0000.0000	CAN IF1 Arbitration 2	1089
0x038	CANIF1MCTL	RW	0x0000.0000	CAN IF1 Message Control	1091
0x03C	CANIF1DA1	RW	0x0000.0000	CAN IF1 Data A1	1094
0x040	CANIF1DA2	RW	0x0000.0000	CAN IF1 Data A2	1094

CAN Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x044	CANIF1DB1	RW	0x0000.0000	CAN IF1 Data B1	1094
0x048	CANIF1DB2	RW	0x0000.0000	CAN IF1 Data B2	1094
0x080	CANIF2CRQ	RW	0x0000.0001	CAN IF2 Command Request	1081
0x084	CANIF2CMSK	RW	0x0000.0000	CAN IF2 Command Mask	1082
0x088	CANIF2MSK1	RW	0x0000.FFFF	CAN IF2 Mask 1	1085
0x08C	CANIF2MSK2	RW	0x0000.FFFF	CAN IF2 Mask 2	1086
0x090	CANIF2ARB1	RW	0x0000.0000	CAN IF2 Arbitration 1	1088
0x094	CANIF2ARB2	RW	0x0000.0000	CAN IF2 Arbitration 2	1089
0x098	CANIF2MCTL	RW	0x0000.0000	CAN IF2 Message Control	1091
0x09C	CANIF2DA1	RW	0x0000.0000	CAN IF2 Data A1	1094
0x0A0	CANIF2DA2	RW	0x0000.0000	CAN IF2 Data A2	1094
0x0A4	CANIF2DB1	RW	0x0000.0000	CAN IF2 Data B1	1094
0x0A8	CANIF2DB2	RW	0x0000.0000	CAN IF2 Data B2	1094
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	1095
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	1095
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	1096
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	1096
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	1097
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	1097
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	1098
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	1098

Controller Area Network Signals

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CAN0Rx	28 58 59	PF0 (3) PB4 (8) PE4 (8)	I	TTL	CAN module 0 receive.
CAN0Tx	31 57 60	PF3 (3) PB5 (8) PE5 (8)	O	TTL	CAN module 0 transmit.
CAN1Rx	17	PA0 (8)	I	TTL	CAN module 1 receive.
CAN1Tx	18	PA1 (8)	O	TTL	CAN module 1 transmit.

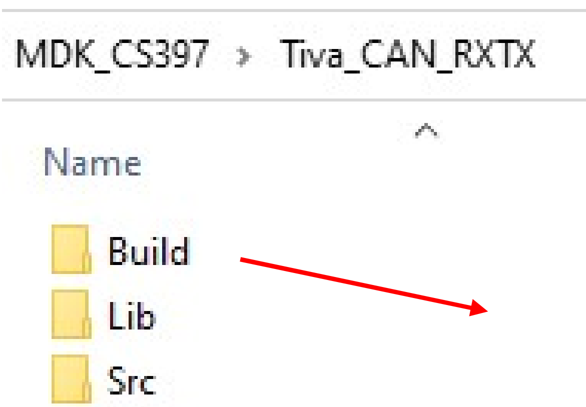
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

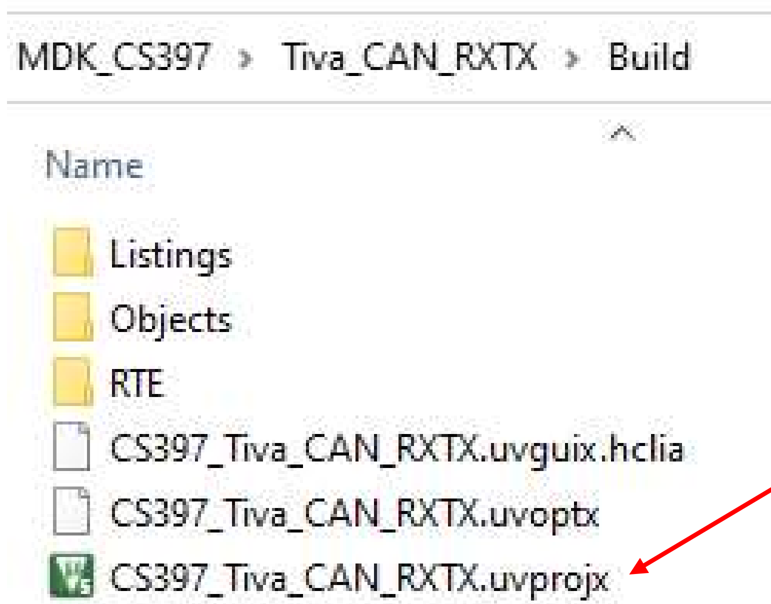
Note: The number in parentheses is the encoding that must be programmed into the PMCn field in the GPIO Port Control (GPIOPCTL) register (page 688) to assign the CAN signal to the specified GPIO port pin.

The project file

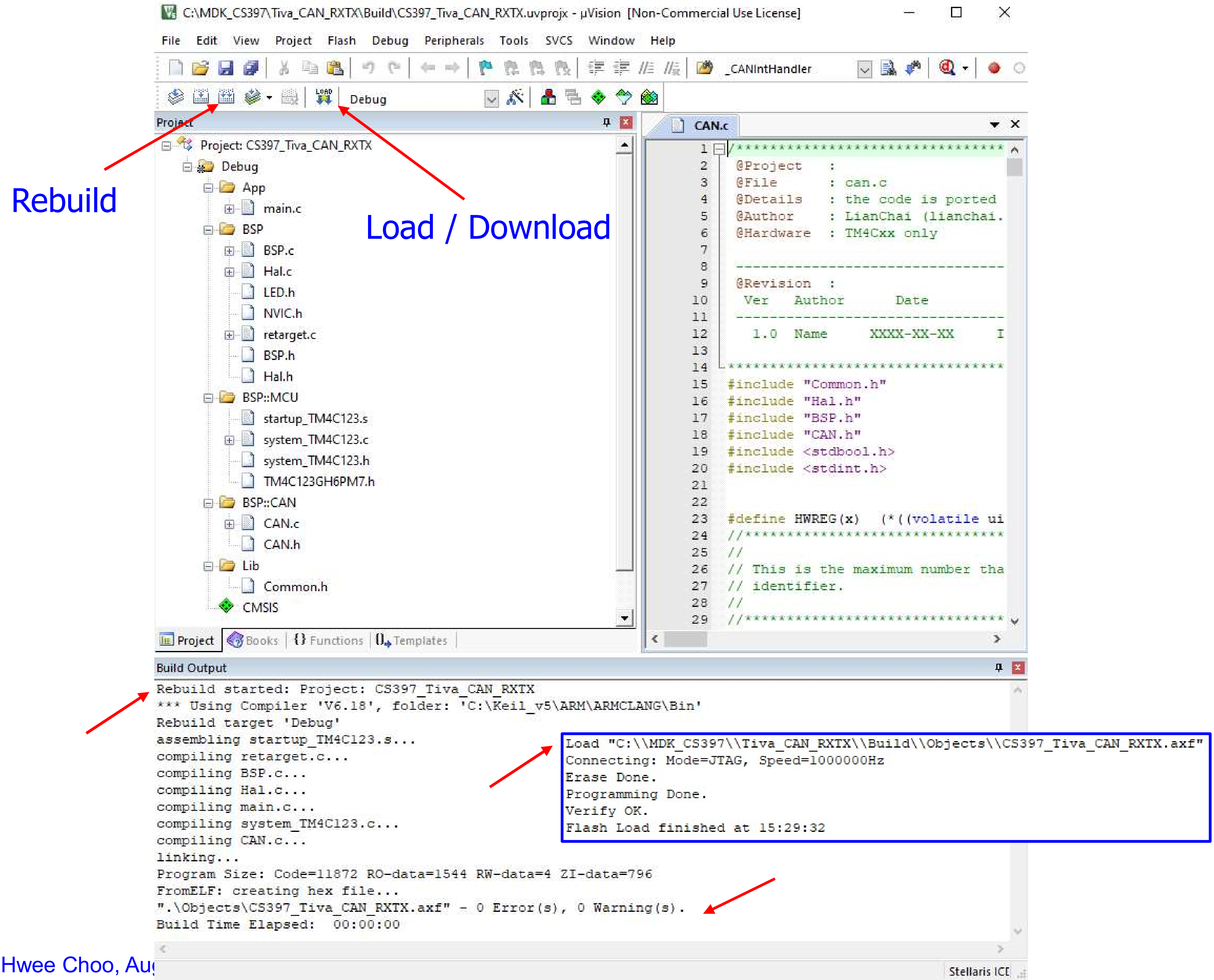
Unzip **13_CS397_Hands-On_6-3_Tiva_CAN_RXTX.zip** and copy the contents to a folder, e.g.,

C:\MDK_CS397\Tiva_CAN_RXTX





Double-click to run this project and build


C:\MDK_CS397\Tiva_CAN_RXTX\Build\CS397_Tiva_CAN_RXTX.uvprojx - µVision [Non-Commercial Use License]
Rebuild
Load / Download
Project: CS397_Tiva_CAN_RXTX
CAN.c
Build Output
Rebuild started: Project: CS397_Tiva_CAN_RXTX
*** Using Compiler 'V6.18', folder: 'C:\Keil_v5\ARM\ARMCLANG\Bin'
Rebuild target 'Debug'
assembling startup_TM4C123.s...
compiling retarget.c...
compiling BSP.c...
compiling Hal.c...
compiling main.c...
compiling system_TM4C123.c...
compiling CAN.c...
linking...
Program Size: Code=11872 RO-data=1544 RW-data=4 ZI-data=796
FromELF: creating hex file...
".\Objects\CS397_Tiva_CAN_RXTX.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00
Load "C:\\MDK_CS397\\Tiva_CAN_RXTX\\Build\\Objects\\CS397_Tiva_CAN_RXTX.axf"
Connecting: Mode=JTAG, Speed=1000000Hz
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 15:29:32

Code in Hal.c (1/3)

```
/* Hal.c */
#include <Common.h>
#include "Hal.h"

void Port_Init( void )
{
    /* enable GPIO port and clock (pg 340)*/
    SYSCTL->RCGCGPIO   |= SYSCTL_RCGCGPIO_R0  /* enable Port A clock*/
                        | SYSCTL_RCGCGPIO_R1  /* enable Port B clock*/
                        | SYSCTL_RCGCGPIO_R4  /* enable Port E clock*/
                        | SYSCTL_RCGCGPIO_R5; /* enable Port F clock */

    /* enable clock to UART0   */
    SYSCTL->RCGCUART |= SYSCTL_RCGCUART_R0;

    /* Wait for GPIO port to be ready */
    while( 0 == (SYSCTL->PRGPIO & SYSCTL_PRGPIO_R0) );/* port A */
    while( 0 == (SYSCTL->PRGPIO & SYSCTL_PRGPIO_R1) );/* port B */
    while( 0 == (SYSCTL->PRGPIO & SYSCTL_PRGPIO_R4) );/* port E */
    while( 0 == (SYSCTL->PRGPIO & SYSCTL_PRGPIO_R5) ){};/* port F */

    /* Wait for UART to be ready */
    while( 0 == (SYSCTL->PRUART & SYSCTL_PRUART_R0) );

    /* Unlock GPIO PF[0] */
    GPIOF->LOCK = GPIO_LOCK_KEY;/* Unlock Port F (pg 684)*/
    GPIOF->CR |= BIT(PF_SW2);/* Set Commit Control register for PF[0] */
```

Code in Hal.c (2/3)

```
/* Initialize RGB LED  */
GPIOF->DIR |= BIT(PF_LED_RED) | BIT(PF_LED_BLUE) | BIT(PF_LED_GREEN);
GPIOF->DEN |= BIT(PF_LED_RED) | BIT(PF_LED_BLUE) | BIT(PF_LED_GREEN);
GPIOF->AFSEL &=~( BIT(PF_LED_RED) | BIT(PF_LED_BLUE) | BIT(PF_LED_GREEN) );

/* Buzzer */
GPIOA->DIR |= BIT( PA_BUZZER);
GPIOA->DEN |= BIT (PA_BUZZER);

/* initialize GPIO PA0 (UART0_RX) & PA1 (UART0_TX)   */
GPIOA->AFSEL |= BIT(PA_UART0_RX) | BIT(PA_UART0_TX);
GPIOA->DEN   |= BIT(PA_UART0_RX) | BIT(PA_UART0_TX);
GPIOA->AMSEL &= ~( BIT(PA_UART0_RX) | BIT(PA_UART0_TX) );
GPIOA->PCTL  &= ~( GPIO_PCTL_PA0_M | GPIO_PCTL_PA1_M ); /* clear Port C config bits  */
GPIOA->PCTL  |= GPIO_PCTL_PA0_U0RX | GPIO_PCTL_PA1_U0TX;

/** initialize UART0   **/
UART0->CTL &= ~UART_CTL_UARTEN; /* disable UART during initialization  */
UART0->CC  &= ~UART_CC_CS_M;     //  clock source mask
UART0->CC  |= UART_CC_CS_SYSCLK; // set to system clock

UART0->CTL &= ~UART_CTL_HSE;/* use 16X CLKDIV  */
UART0->IBRD = 5; /* int (80,000,000 / (16 * 921,600)) = 5.425347    */
UART0->FBRD = 27;  /* int (5.425347 * 64 + 0.5 = int (27.72) = 27   */

UART0->LCRH &= ~UART_LCRH_WLEN_M;
UART0->LCRH |= UART_LCRH_WLEN_7 | UART_LCRH_PEN; /* 7 data bits, parity enable  */
UART0->LCRH |= UART_LCRH_EPS ;   /* even parity  */
UART0->LCRH &= ~UART_LCRH_STP2;  /* 1 stop bit */
```

Code in Hal.c (3/3)

```
        UART0->LCRH |= UART_LCRH_FEN;   /* enable FIFO*/

        UART0->CTL  |= UART_CTL_TXE;    /* transmit enable */
        UART0->CTL  |= UART_CTL_UARTEN; /* enable UART0   */

        /************************
            CAN0
        ************************/
        SYSCTL->RCGCCAN |= SYSCTL_RCGCCAN_R0;
        while( 0 == (SYSCTL->PRCAN & SYSCTL_PRCAN_R0) ){}

        GPIOE->CR    |= BIT(PE_CAN0_RX) | BIT(PE_CAN0_TX);
        GPIOE->DEN   |= BIT(PE_CAN0_RX) | BIT(PE_CAN0_TX);
        GPIOE->AFSEL |= BIT(PE_CAN0_RX) | BIT(PE_CAN0_TX);
        GPIOE->PCTL  |= GPIO_PCTL_PE4_CAN0RX | GPIO_PCTL_PE5_CAN0TX;
    }


    /** Write an ASCII character to UART0**/
    /** character = ASCII to write **/
    void write_ASCII_UART0 (char character )
    {
        while( 0 != (UART0->FR & UART_FR_TXFF) ){}; /* wait if TX FIFO full*/
        UART0->DR = character; /* write character to UART data reg */
    }
```

CR : GPIO Commit
DEN : GPIO Digital Enable
AFSEL : GPIO Alternate Function Select
PCTL : GPIO Port Control

Code in main.c (1/6)

```
/* main.c */

#include "Common.h"
#include "Hal.h"
#include "BSP.h"
#include "LED.h"
#include "NVIC.h"
#include "CAN.h"

/* Local Variables */
static volatile BOOL  g_bSystemTick = FALSE;
static volatile BOOL  g_bLED        = FALSE,  g_bCAN = FALSE;
static int            g_count = 0, g_canCount = 0;

static CAN_HANDLE     g_CAN0Handle;
static tCANMsgObject  g_CAN0RxMsg;
static uint8_t        g_aCAN0RxMsgData[8];
static volatile BOOL  g_bOnCAN0Rx = FALSE;

static tCANMsgObject  g_CAN0TxMsg;
static uint8_t        g_aCAN0TxMsgData[8];

/* Local Functions */
static void main_CAN0Init( void );

/* Callback Functions */
static void main_cbOnCAN0Rx( int nObj );

/* Implementation */
```

Code in main.c (2/6)

```
/* Implementation */
int main()
{
    BSPInit(); /* in BSP.c   */
    BOOL bToggle = TRUE;
    SystemCoreClockUpdate();

    SysTick_Config( SystemCoreClock/1000 );   /* Initialize SysTick ticks every 1 ms */

    /* print to Virtual COM port temrinal */
    printf ("\nHello World! \n\r"); // display to virtual COM port
    printf ("Welcome to CS397!:\n\r");

    main_CAN0Init();

    for(;;)
    {
        if (FALSE != g_bSystemTick) /* check every system tick */
        {
            g_bSystemTick = FALSE;
        }

        if( FALSE != g_bLED )/* Check if LED flag is set */
        {
            /* Clear SysTick flag so we only processes it once */
            g_bLED = FALSE;
            /* Set LED to BLUE if toggle is TRUE(=1), */
            /* otherwise if toggle is FALSE(=0), the LED will be off */
            LED_RGB_SET( RGB_BLUE * bToggle );
            bToggle = !bToggle; /* Inverse toggle, so if 0 it becomes 1, 1 becomes 0 */
        }
```

Code in main.c (3/6)

```
/* CAN0 RX */

if( FALSE != g_bOnCAN0Rx )
{
    g_bOnCAN0Rx = FALSE;
    /* Read data */
    CANMessageGet(&g_CAN0Handle, 1, &g_CAN0RxMsg, 0);

    if ( 0 != g_CAN0RxMsg.ui32MsgLen)
    {
        printf(
        "CAN0 RX: Len: %d ID: 0x%x Data = 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x\r\n",
        g_CAN0RxMsg.ui32MsgLen,  g_CAN0RxMsg.ui32MsgID,
        g_aCAN0RxMsgData[0],g_aCAN0RxMsgData[1],g_aCAN0RxMsgData[2],g_aCAN0RxMsgData[3],
        g_aCAN0RxMsgData[4],g_aCAN0RxMsgData[5],g_aCAN0RxMsgData[6],g_aCAN0RxMsgData[7]);

        g_CAN0RxMsg.ui32MsgLen = 0;
    }
}
```

Code in main.c (4/6)

```
        /* CAN0 TX */

        if( FALSE != g_bCAN )
        {
            g_bCAN = FALSE;
            g_aCAN0TxMsgData[0] = 0xB1;
            g_aCAN0TxMsgData[1] = 0xB2;
            g_aCAN0TxMsgData[2] = 0xB3;
            g_aCAN0TxMsgData[3] = 0xB4;
            g_aCAN0TxMsgData[4] = 0xB5;
            g_aCAN0TxMsgData[5] = 0xB6;
            g_aCAN0TxMsgData[6] = 0xB7;
            g_aCAN0TxMsgData[7] = 0xB8;
            g_CAN0TxMsg.ui32MsgID = 0x389;
            g_CAN0TxMsg.ui32Flags   = MSG_OBJ_TX_INT_ENABLE;
            g_CAN0TxMsg.ui32MsgLen  = sizeof(g_aCAN0TxMsgData);;
            g_CAN0TxMsg.pui8MsgData = g_aCAN0TxMsgData;

            /* CAN0 Tx Msg */
            CANMessageSet(&g_CAN0Handle, 2, &g_CAN0TxMsg, MSG_OBJ_TYPE_TX);

            printf(
              "CAN0 TX: Len: %d ID: 0x%x Data = 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x 0x%x\r\n",
              g_CAN0TxMsg.ui32MsgLen, g_CAN0TxMsg.ui32MsgID,
              g_aCAN0TxMsgData[0],g_aCAN0TxMsgData[1],g_aCAN0TxMsgData[2],g_aCAN0TxMsgData[3],
              g_aCAN0TxMsgData[4],g_aCAN0TxMsgData[5],g_aCAN0TxMsgData[6],g_aCAN0TxMsgData[7]);

        }
    }
}
```

Code in main.c (5/6)

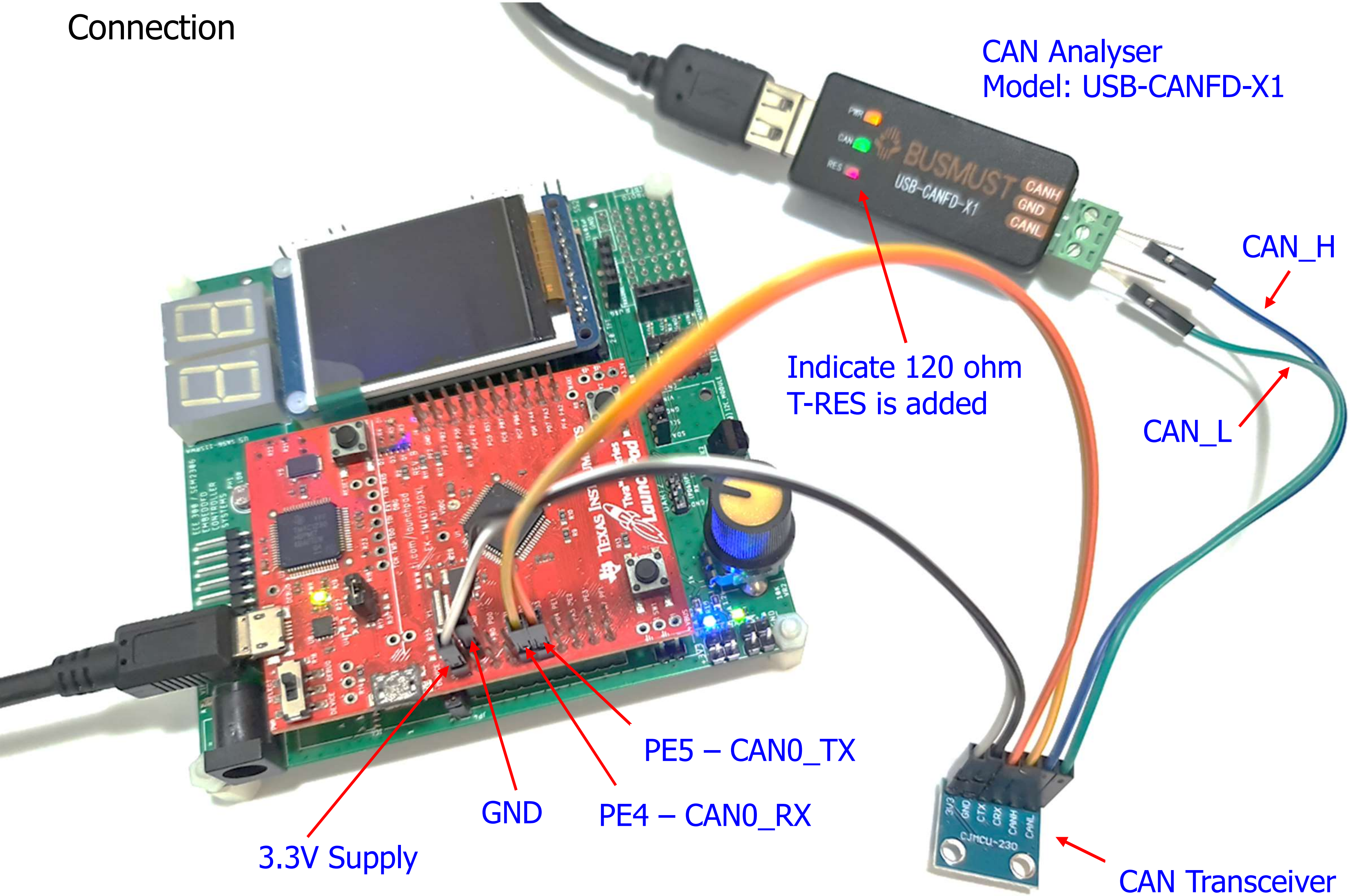
```
/* Callback functions */
void SysTick_Handler( void )
{
    g_bSystemTick = TRUE;
    g_count++;
    g_canCount++;

    if (g_count%500 == 0) /* set flag every 500ms */
    {
        g_bLED = TRUE;
    }

    if (g_canCount%1000 == 0) /* set flag every 1000ms */
    {
        g_bCAN = TRUE;
    }
}

static void main_cbOnCAN0Rx( int nObj )
{
    g_bOnCAN0Rx = TRUE;
}
```

Code in main.c (6/6)

```
/* Local functions */
static void main_CAN0Init( void )
{
    CANInit( &g_CAN0Handle, 0 ); /* Initializes CAN0 */

    CANBitRateSet( &g_CAN0Handle, SystemCoreClock, 1000000 ); /* Set CAN0 to 1Mbps */

    /* Enable CAN0 Interrupts */
    CANIntEnable( &g_CAN0Handle, CAN_INT_MASTER | CAN_INT_ERROR | CAN_INT_STATUS );

    CANEnable( &g_CAN0Handle ); /* Enable CAN0 */

    /* Add callback to get data received notification */
    CANIntRegister( &g_CAN0Handle, main_cbOnCAN0Rx );

    g_CAN0RxMsg.ui32MsgID = 0;
    g_CAN0RxMsg.ui32MsgIDMask = 0;
    g_CAN0RxMsg.ui32Flags = MSG_OBJ_RX_INT_ENABLE | MSG_OBJ_USE_ID_FILTER;
    g_CAN0RxMsg.ui32MsgLen = 8;
    g_CAN0RxMsg.pui8MsgData = g_aCAN0RxMsgData;

    /* Now load the message object into the CAN peripheral.  Once loaded the
    CAN will receive any message on the bus, and an interrupt will occur.
    Use message object 1 for receiving messages (this is not the same as
    the CAN ID which can be any value in this example). */

    CANMessageSet(&g_CAN0Handle, 1, &g_CAN0RxMsg, MSG_OBJ_TYPE_RX);
}
```

Hands-On Tiva TM4C123G CAN RX TX

Connection

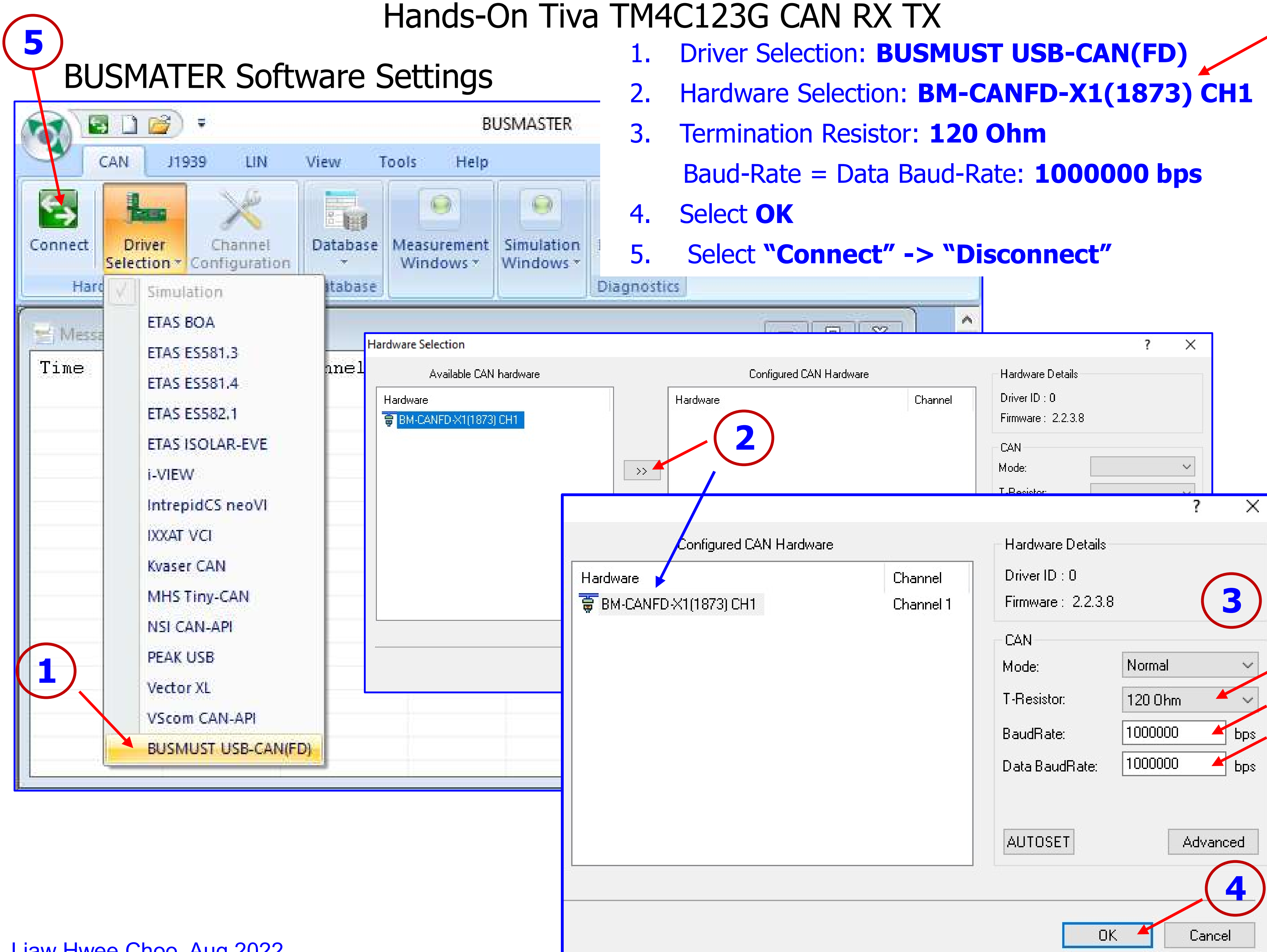


Hands-On Tiva TM4C123G CAN RX TX

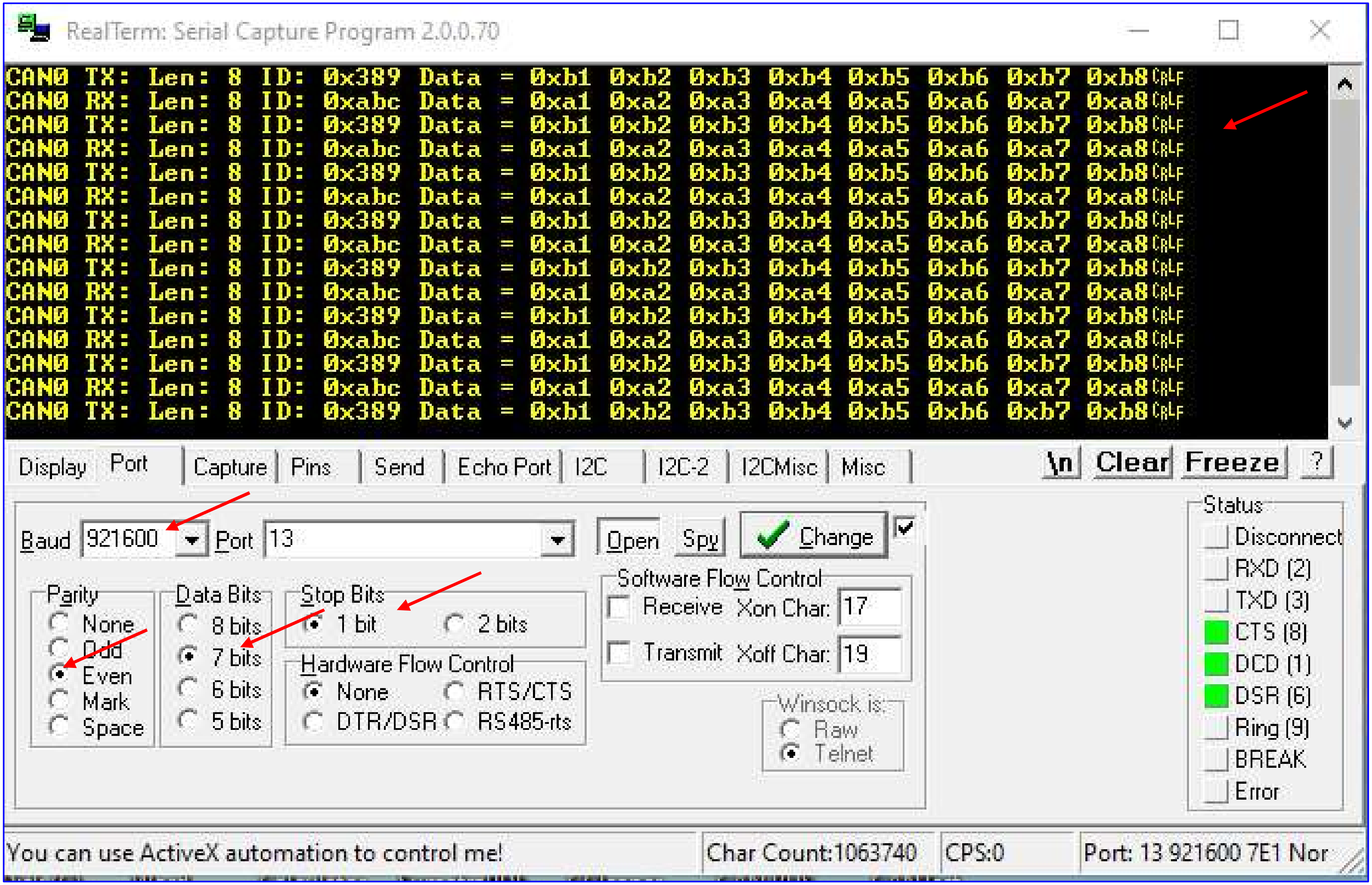
BUSMATER Software Settings

1. Driver Selection: **BUSMUST USB-CAN(FD)**
2. Hardware Selection: **BM-CANFD-X1(1873) CH1**
3. Termination Resistor: **120 Ohm**

   Baud-Rate = Data Baud-Rate: **1000000 bps**
4. Select **OK**
5. Select **"Connect" -> "Disconnect"**



Tiva CAN0 RX & TX Results on RealTerm



Hands-On Tiva TM4C123G CAN RX TX

Tiva CAN0 RX & TX Results on Analyzer