```
/*******************************************************************************
 Module
   SCI_Receive.c

 Revision
   1.0.1

 Description
   This is a template file for implementing a simple service under the
   Gen2 Events and Services Framework.

 Notes

 History
 When            Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
*******************************************************************************/
/*---------------------------- Include Files ----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include <stdio.h>
//#include "ME218_C32.h"
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "Coach.h"
#include "ES_Timers.h"
//#include "timers12.h"
#include "ADS12.h"
#include <mc9s12e128.h>     /* derivative information */
#include <s12E128bits.h>
#include <s12vec.h>
#include "SCI_Receive.h"
#include "SCI_Send.h"
#include "Ultrason.h"

#define WaitPariTime 1000
#define HeartBeatTime 3000
#define SendCtrlTime 200

#define STEERING_NEUTRAL 768 //Accelerometer A/D reading with control held horizontal
#define MAX_STEERING_LEFT 922 //Accelerometer A/D reading that maps to max left turn
#define MAX_STEERING_RIGHT 614 //Accelerometer A/D reading that maps to max right turn



/*---------------------------- Module Defines ----------------------------*/

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/

/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
static CoachState_t CurrentState;
static ES_Event PostEvent;
```

```
static unsigned char Data, RID, KICK;
static unsigned char AddMSB, AddLSB, AUX, WHIM, TRANS;    // To be query by send service
static signed char SPD, DIR;
//static unsigned char CurrentAux, LastAux;
static unsigned short TeamN;
static signed int Raw_Throttle_Command;
static short Current_Throttle_AD, Current_Steering_AD;
signed long Raw_Steering_Command;

/*----------------------------- Module Code -----------------------------*/
/****************************************************************************
 Function
      Init_SCI_Receive

 Parameters
      uint8_t : the priorty of this service

 Returns
      boolean, False if error in initialization, True otherwise

 Description
      Saves away the priority, and does any
      other required initialization for this service
 Notes

 Author
      J. Edward Carryer, 01/16/12, 10:00
****************************************************************************/
bool Init_Coach ( uint8_t Priority )
{
    ES_Event ThisEvent;

    MyPriority = Priority;

    CurrentState = IDLE;



    Data = 0x00;    // Initialize the Data


    SPD = 0x00;
    DIR = 0x00;
    AUX = 0x00;
    WHIM = 0x00;
    TRANS = 0x00;


    //Initialize the AD pins Port 0 and 1

    if (ADS12_Init("IIIIAAAA")==ADS12_OK)
        {
            puts("Initilized AD Port\n\r");
        }



    printf("COACH INIT");
```

```
    PTT = 0x00;
    PTU = 0x00;
    //LastAux = 0x00;
    //CurrentAux = 0x00;

    ThisEvent.EventType = ES_INIT;
    if (ES_PostToService( MyPriority, ThisEvent) == true)
        {
            return true;
        }
    else
        {
            return false;
        }


}

/****************************************************************************
 Function
     Post_SCI_Receive

 Parameters
     EF_Event ThisEvent ,the event to post to the queue

 Returns
     boolean False if the Enqueue operation failed, True otherwise

 Description
     Posts an event to this state machine's queue
 Notes

 Author
     J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool Post_Coach( ES_Event ThisEvent )
{
    return ES_PostToService( MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    Run_SCI_Receive

 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes

 Author
   J. Edward Carryer, 01/15/12, 15:23
****************************************************************************/
ES_Event Run_Coach( ES_Event ThisEvent )
{
    ES_Event ReturnEvent;
```

```
ReturnEvent.EventType = ES_NO_EVENT; // assume no errors


switch ( CurrentState )
    {
    case IDLE:

        //printf("Idle");



        if (ThisEvent.EventType == PairButtonPress)
            {


                PTU = 0x00;     //Clear all LED


                TeamN = ADS12_ReadADPin(3)/4;    // Read from the AD pin

                if (0 <= TeamN && TeamN <= 23)
                    {
                        RID = 1;
                    }

                else if (24 <= TeamN && TeamN <= 41)
                    {
                        RID = 2;
                    }

                else if (42 <= TeamN && TeamN <= 59)
                    {
                        RID = 3;
                    }

                else if (60 <= TeamN && TeamN <= 77)
                    {
                        RID = 4;
                    }

                else if (78 <= TeamN && TeamN <= 95)
                    {
                        RID = 5;
                    }

                else if (96 <= TeamN && TeamN <= 113)
                    {
                        RID = 6;
                    }

                else if (114 <= TeamN && TeamN <= 131)
                    {
                        RID = 7;
                    }

                else if (132 <= TeamN && TeamN <= 149)
                    {
                        RID = 8;
                    }
```

```
				else if (150 <= TeamN && TeamN <= 167)
					{
						RID = 9;
					}

				else if (168 <= TeamN && TeamN <= 185)
					{
						RID = 10;
					}

				else if (186 <= TeamN && TeamN <= 203)
					{
						RID = 11;
					}

				else if (204 <= TeamN && TeamN <= 221)
					{
						RID = 12;
					}

				else if (222 <= TeamN && TeamN <= 255)
					{
						RID = 13;
					}

				//printf("TeamN now is: %d\n\r", TeamN);

				printf("Rid: %x\n\r", RID);
				Data = RID;

				if ((PTAD & BIT7HI) == BIT7HI)
					{
						printf("Green!\n\r");
						Data = Data | BIT4HI;	// If COL is set, it is green, otherwise it is red
					}

				printf("%d\n\r", Data);

				//Data = 0x1C;
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!




				PostEvent.EventType = SendPairReq;
				PostEvent.EventParam = Data;
				Post_SCI_Send (PostEvent);




				Data = 0x00;

				CurrentState = WAIT_PAIR;

				ES_Timer_SetTimer (WaitPair_TIMER, WaitPariTime);		// Start the wait pair timer
				ES_Timer_StartTimer(WaitPair_TIMER);
			}
		break;
```

```
case WAIT_PAIR:

    //printf("In the assembling state");

    if (ThisEvent.EventType == PAIR_RESP)
        {


            printf("Pair RES\n\r");


            if ((Query_SCI_Receive (3) & BIT1HI) == BIT1HI) //CERR is set                    {
                {
                    CurrentState = IDLE;

                    PTU = (PTU | BIT2HI);               //LED

                    printf("Wrong color-back to Idle\n\r");
                }

            else if ((Query_SCI_Receive (3) & BIT2HI) == BIT2HI) //PERR is set                {
                {
                    CurrentState = IDLE;

                    PTU = (PTU | BIT1HI);               //LED

                    printf("Already paired-back to Idle\n\r");
                }



            else if ((Query_SCI_Receive(3) & BIT0HI) == BIT0HI) // ACC is set
                {
                    AddMSB = Query_SCI_Receive(0);
                    AddLSB = Query_SCI_Receive(1);



                    printf("MSB ADD: %x\n\r", AddMSB);
                    printf("LSB ADD: %x\n\r", AddLSB);
                    printf("DT is: %x\n\r", Query_SCI_Receive(2));
                    printf("DV is: %x\n\r", Query_SCI_Receive(3));


                    PTU = (PTU | BIT0HI);               //LED


                    CurrentState = PAIRED;

                    ES_Timer_SetTimer (HeartBeat_TIMER, HeartBeatTime);       //Start the
Heartbeat Timer
                    ES_Timer_StartTimer(HeartBeat_TIMER);



                    ES_Timer_SetTimer (SendControl_TIMER, SendCtrlTime);       //Start the
Send CTRL Timer
                    ES_Timer_StartTimer(SendControl_TIMER);
```

```
                    }

                }


            else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
WaitPair_TIMER)) //Pair Timer time out
                {
                    CurrentState = IDLE;
                    printf("Wait pair timeout-back to Idle\n\r");
                }



            break;







        case PAIRED:

            //printf("In the state of PAIRED");


            if (ThisEvent.EventType == GetHeartBeat)
                {

                    printf("HB\n\r");
                    printf("DT: %x\n\r", Query_SCI_Receive(2));
                    printf("DV: %x\n\r", Query_SCI_Receive(3));

                    ES_Timer_SetTimer (HeartBeat_TIMER, HeartBeatTime);       //Reset the Heartbeat
Timer
                    ES_Timer_StartTimer(HeartBeat_TIMER);
                }



            else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
HeartBeat_TIMER)) //HeartBeat Timer time out
                {
                    CurrentState = IDLE;
                    Write_LED(0);

                    PTU = 0x00;             //Put off all LED

                    printf("No Heart beat..... Paired timeout-back to Idle\n\r");
                }
            else if (ThisEvent.EventType == TaggedOut)   //Status [ACT] = 0
                {
                    CurrentState = IDLE;
                    Write_LED(0);

                    PTU = 0x00;             //Put off all LED

                    printf("Get TAGGED OUT message -back to Idle\n\r");
```

```
                }



            else if ((ThisEvent.EventType == TagOut) && (Query_SendState() == Assembling))    //Send
out tag out command
                {
                    PostEvent.EventType = SendTAGOUT;
                    PostEvent.EventParam = ThisEvent.EventParam;
                    Post_SCI_Send (PostEvent);

                    //printf("Snedout tag out alreday in the coach service");
                }

            else if ((ThisEvent.EventType == Reset) && (Query_SendState() == Assembling))    //Send out
reset command
                {
                    PostEvent.EventType = SendReset;
                    Post_SCI_Send (PostEvent);
                    CurrentState = IDLE;
                    Write_LED(0);

                    PTU = 0x00;             //Put off all LED

                    printf("Alreday reset go back to Idle\n\r");
                }



            else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
SendControl_TIMER)) //Send Control Timer time out
                {

                    AUX = 0x00;
                    WHIM = 0x00;

                    Current_Throttle_AD = ADS12_ReadADPin(1);    // Speed

                    //`````````````````````````````````` Get Speed value
                    Raw_Throttle_Command = (signed int)Current_Throttle_AD/4 - 128;

                    //Bound and set thresholds on signal value
                    if (Raw_Throttle_Command < -118)
                        {
                            Raw_Throttle_Command = -128;
                        }
                    else if (Raw_Throttle_Command > 117)
                        {
                            Raw_Throttle_Command = 117;
                        }

                    SPD = (signed char)(Raw_Throttle_Command);



                    if (SPD <= -10)
                        {
                            AUX = 0xFF;     // Send out breaking value
```

```
                    }



                    //`````````````````````````````````````` Get Steering value
                    Current_Steering_AD = ADS12_ReadADPin(2);

                    Raw_Steering_Command = ((signed long)Current_Steering_AD -
STEERING_NEUTRAL);

                    if (Raw_Steering_Command < 0)
                        {
                            Raw_Steering_Command =
(Raw_Steering_Command*128)/(STEERING_NEUTRAL - MAX_STEERING_RIGHT);
                        }
                    else
                        {
                            Raw_Steering_Command =
(Raw_Steering_Command*127)/(MAX_STEERING_LEFT - STEERING_NEUTRAL);
                        }

                    //Bound and set thresholds on signal value
                    if (Raw_Steering_Command < -118)
                        {
                            Raw_Steering_Command = -128;
                        }
                    else if (Raw_Steering_Command > 117)
                        {
                            Raw_Steering_Command = 127;
                        }

                    DIR = (signed char)(Raw_Steering_Command);




                    //AUX = ADS12_ReadADPin(3);    // Breaking


                    printf("ultrais:%d\n\r",Query_Ultrason());
                    if (Query_Ultrason() > 250)
                        {
                            KICK = 0x00;                        // Kicking
                            //printf("1-%i\n\r", Query_Ultrason());
                        }
                    else
                        {
                            KICK = 0x0f;                        // Not Kicking
                            //printf("0-%i\n\r", Query_Ultrason());
                        }

                    //printf("Bef%d\n\r", AUX);
                    AUX =    ((AUX<<4) | KICK);
                    //printf("Aft%d\n\r", AUX);
```

```
                    if ((PTU & BIT7HI) == BIT7HI)
                        {
                            WHIM = 0xFF;
                        }

                    TRANS = 0x00;

                    PostEvent.EventType = SendCtrl;
                    Post_SCI_Send (PostEvent);


                    ES_Timer_SetTimer (SendControl_TIMER, SendCtrlTime);        //Start the Send CTRL
Timer
                    ES_Timer_StartTimer(SendControl_TIMER);
                }



            break;
        }

    return ReturnEvent;
}



unsigned char Query_AddMSB ( void )
{
    return(AddMSB);
}

unsigned char Query_AddLSB ( void )
{
    return(AddLSB);
}

unsigned char Query_SPD ( void )
{
    return(SPD);
}

unsigned char Query_DIR ( void )
{
    return(DIR);
}

unsigned char Query_AUX ( void )
{
    return(AUX);
}

unsigned char Query_WHIM ( void )
{
    return(WHIM);
}

unsigned char Query_TRANS ( void )
{
```

```
    return(TRANS);
}
/*****************************************************************************
 private functions
 *****************************************************************************/



/*------------------------------ Footnotes -------------------------------*/
/*------------------------------ End of file -----------------------------*/
```