```
                        /
*****************************************************************************
 Module
   SPIService.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 SPIFSM.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 "ES_Configure.h"
#include "ES_Framework.h"
#include "SPIService.h"
#include "ES_Timers.h"
#include <stdio.h>
#include <hidef.h>
#include <mc9s12e128.h>
#include <s12e128bits.h>
#include <Bin_Const.h>
#include <termio.h>
#include "S12eVec.h"
#include "ME218_E128.h"
#include "ES_Events.h"
#include "CommandSM.h"


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

//Definition of the command from command modulus

#define JP2_SS    PTS_PTS7
#define SPI_FLOW_DEBUG  1                       //Debug flag for SPI Service
#define NO_FAC 0                                        //Override and send simulated
command if FAC is not detected


/*---------------------------- 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 unsigned char command;
static unsigned int counter;
static unsigned int commandidentifier;
static unsigned char XX,YY,RR,KK;


// command is a module level variable

/*---------------------------- Module Code ----------------------------*/
/****************************************************************************
 Function
     InitSPIService

 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
****************************************************************************/
boolean InitSPIService( uint8_t Priority )

{
    ES_Event ThisEvent;

     puts("SPI init \r\n");

    MyPriority = Priority;

        /******************Start of SPI initializtion*********************************/
    SPIBR |= (_S12_SPPR0 | _S12_SPPR1 | _S12_SPPR2 | _S12_SPR0 | _S12_SPR1 |_S12_SPR2); //
Baud rate divisor of 64
    SPICR1 |=_S12_SPE;
    SPICR1 |=_S12_MSTR;
    SPICR1 |= (_S12_CPOL|_S12_CPHA);
    SPICR1 |=_S12_SPIE;         //Enable the SPI
    DDRS |= BIT7HI;
    EnableInterrupts;
        /******************End of SPI initializtion**********************************/

    ThisEvent.EventType = ES_INIT;
    if (ES_PostToService( MyPriority, ThisEvent) == True)
        {
            return True;
```

```
        }
    else
        {
            return False;
        }

}

/*****************************************************************************
 Function
     PostSPIService

 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
*****************************************************************************/
boolean PostSPIService( ES_Event ThisEvent )
{
    return ES_PostToService( MyPriority, ThisEvent);
}

/*****************************************************************************
 Function
    RunSPIService

 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
*****************************************************************************/
 //counter = 0 ;

ES_Event RunSPIService( ES_Event ThisEvent )
{


    ES_Event ReturnEvent;
```

```
    ES_Event PltfmEvent;

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

        // THIS is counter for 5 bytes to be sent to FAC corresponding to each command


    switch (ThisEvent.EventType)

    {

  case ES_STATUS:       //ES_STATUS, the event to trigger the checking of Game Status
            {

               puts("Checking the game statuts \r\n");
               SPICR1 |=_S12_SPE;
               JP2_SS = 0;    // decide the SS line E128 pin here  and pull it LO
               counter=1;       // This is the first command so counter = 1
               commandidentifier = 200;  // This is the commandidentifier which is 200 for
status command and 100 for coordinate command
               SPISR;
               SPIDR = 0x3F;   // First of the Status commands

            }break;

   case ES_Command:             //ES_Command, the evemt to trigger the bot/ships positions

            {
               printf("Checking the ships %d positions \r\n",ThisEvent.EventParam);
               SPICR1 |=_S12_SPE;
               JP2_SS = 0;    // decide the SS line pin here and pull it LO
               counter=1;
               commandidentifier = 100; // This is the commandidentifier which is 200 for
status command and 100 for coordinate command

               SPISR;
               SPIDR = ThisEvent.EventParam;

            }break;


  case ES_Bytereceived:                 // this is byte received event from the interrupt
routine

           {

            if (counter == 2 && commandidentifier ==200 )       // send second command

            {
               SPISR;
               SPIDR = 0x00;
               puts("status byte Received \r\n");
            }
```

```
             else if (counter == 3 && commandidentifier ==200 )      //send third command

             {
                SPISR;
                SPIDR = 0x00;

             }



             else if (counter == 4 && commandidentifier ==200 )

             {
                SPISR;
                SPIDR = 0x00;

            }



             else if (counter == 5 && commandidentifier ==200 )

             {

               SPISR;
               SPIDR = 0x00;


             }


              else if (counter == 6 && commandidentifier ==200 )          // after 5th
command received

             {

               SPISR;
               command=SPIDR;

               switch(command)

               {

               printf("Command is %d \r\n",command);


               case 0x00:                //Status is Game stop

                {
                   if (SPI_FLOW_DEBUG)
                   {
```

```
      puts ("Game Stop \r\n");
     }

  }break;

 case 0b01000000 :         //Status is Red Supply Station is offline
 {

      PltfmEvent.EventType = ES_REDREPAIR;
       PostPltfmFSM(PltfmEvent);
     puts("REDREPAIR \r\n");
 }break;

  case 0b00100000 :        //Status is Blue Supply Station is offline
 {
        PltfmEvent.EventType = ES_REDREPAIR;
       PostPltfmFSM(PltfmEvent);
        puts("BLUEREPAIR \r\n");
 }break;

case 0x0A   :                      //Status is game in progress
 {
        PltfmEvent.EventType = ES_GAMEPLAY;
        puts("FAC: GamePlay@!\r\n");
        PostCommandSM(PltfmEvent);
         //puts("GAMEPLAY \r\n");
 }break;

case 0x0B:                 //Status is Gmae over and Red Win

 {
       PltfmEvent.EventType = ES_GAMEOVERRED;
       PostPltfmFSM(PltfmEvent);
        puts("GAMEOVERED \r\n");
 }break;

case 0x0C:                 //Status us Game over and Blue win
 {
      PltfmEvent.EventType = ES_GAMEOVERBLUE;
       PostPltfmFSM(PltfmEvent);
     puts("GAMEOVERBLUE \r\n");
 }break;

 default:
 {
  puts("FAC:Illegal game status \r\n");
 }
 break;

 }



}
```

```
            else if (counter == 2 && commandidentifier ==100 )     // This is for 5 bytes
corresponding to coordinate command having identifier = 100

            {

               puts("FAC first POS byte received!\r\n");
               SPISR;
               SPIDR = 0x00;

            }



            else if (counter == 3 && commandidentifier ==100 )

            {
               SPISR;
               KK = SPIDR;

               SPISR;
               SPIDR = 0x00;

            }



            else if (counter == 4 && commandidentifier ==100 ) // send 4th command i.e 3rd
byte is received

            {
               SPISR;
               XX = SPIDR;           // 3rd byte received is the xx coordinate
               printf ("XX =%u \r\n",XX) ;
               SPISR;
               SPIDR = 0x00;


           }
           //break;


            else if (counter == 5 && commandidentifier ==100 )

            {

              SPISR;

              YY = SPIDR;         // 4th byte received is the yy coordinate
              printf ("YY = %u \r\n",YY) ;
              SPISR;
              SPIDR = 0x00;
```

```
            }

                else if (counter == 6 && commandidentifier ==100 )

            {

              SPISR;

              RR = SPIDR;                    // make this global ?
               printf ("RR=%u\r\n",RR) ;

                          if (NO_FAC)            //Debug for FAC offline
             {
              XX =10;
              YY = 20;
              RR = 30;
             }

              printf ("Xpos: %d Ypos: %d: Rot: %d First: %d \r\n",XX,YY,RR,KK) ;

              SPICR1 &=(~(_S12_SPE));

               PltfmEvent.EventType = ES_TGT_FOUND_XX;
               PltfmEvent.EventParam = XX;
               PostCommandSM(PltfmEvent);                                          //Post
corresponding Events and the XX position is included in EventParam

               PltfmEvent.EventType = ES_TGT_FOUND_YY;
               PltfmEvent.EventParam = YY;
               PostCommandSM(PltfmEvent);                                          //Post
corresponding Events and the YY position is included in EventParam

               PltfmEvent.EventType = ES_TGT_FOUND_RR;
               PltfmEvent.EventParam = RR;
               PostCommandSM(PltfmEvent);                                          //Post
corresponding Events and the Rotation is included in EventParam


             }


           } break;

    }



    return ReturnEvent;


}
```

```
/*****************************************************************************
 private functions
 *****************************************************************************/


void interrupt _Vec_spi commandread(void)  //SPI interrupt routine

{

    static unsigned long dummy;
    ES_Event ThisEvent;
    if (SPISR_SPIF !=0)

        {

          SPISR;
          dummy=SPIDR;
          counter=counter+1;            //counter which bytes is received

          if (counter == 6)
          {
              JP2_SS = 1;               //if all 6 bytes received, toggle the JP2_SS
          }

          ThisEvent.EventType = ES_Bytereceived;
          ThisEvent.EventParam=1;
          PostSPIService (ThisEvent);           //Self post the event, byte recieved


        }



}
```