```
/*******************************************************************************
 Module
   GameplayHSM.c

 Revision
   2.0.1

 Description
   This is a template for the top level Hierarchical state machine

*******************************************************************************/
/*---------------------------- 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 "GameplayHSM.h"
#include "CalibrationSM.h"
#include "SwitchBranchSM.h"
#include "PIC32PortHAL.h"
#include "PIC32_AD_Lib.h"
#include "MotorService.h"
#include "TapeFollowSM.h"
#include "ButtonModule.h"
#include "terminal.h"
#include "dbprintf.h"
#include "SPILeader.h"


/*---------------------------- Module Defines ----------------------------*/
#define TIMER
static uint16_t ONEMINUTE = 60000;
static uint16_t FINALSECS = 18000;

/*---------------------------- Module Functions ----------------------------*/

static ES_Event_t DuringIdleState( ES_Event_t Event);
static ES_Event_t DuringCalibrationState( ES_Event_t Event);
static ES_Event_t DuringTapeFollowState( ES_Event_t Event);
static ES_Event_t DuringSwitchingBranchesState( ES_Event_t Event);
static bool GameTime( void );

/*---------------------------- Module Variables ----------------------------*/
// everybody needs a state variable, though if the top level state machine
// is just a single state container for orthogonal regions, you could get
// away without it
static GameplayState_t CurrentState;
// with the introduction of Gen2, we need a module level Priority var as well
static uint8_t MyPriority;

//Game Vars
GameSide_t CurrentSide;
static uint8_t Minute = 0;

/*---------------------------- Module Code ----------------------------*/
/*******************************************************************************
 Function
     InitGameplayHSM
```

```
 Parameters
     uint8_t : the priorty of this service

 Returns
     boolean, False if error in initialization, True otherwise

 Description
     Saves away the priority,  and starts
     the top level state machine
 Notes

 Author
     J. Edward Carryer, 02/06/12, 22:06
*****************************************************************************/
bool InitGameplayHSM ( uint8_t Priority )
{

       // save our priority
        //setup start buttons
       //configure AD converter for tape
       //start HSM
}

/*****************************************************************************
 Function
     PostGameplayHSM

 Parameters
     ES_Event_t ThisEvent , the event to post to the queue

 Returns
     boolean False if the post 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 PostGameplayHSM( ES_Event_t ThisEvent )
{
    RunGameplayHSM(ThisEvent);
    return ES_PostToService( MyPriority, ThisEvent);
}

/*****************************************************************************
 Function
    RunGameplayHSM

 Parameters
   ES_Event: the event to process

 Returns
   ES_Event: an event to return

 Description
   the run function for the top level state machine
 Notes
   uses nested switch/case to implement the machine.
```

```
 Author
   J. Edward Carryer, 02/06/12, 22:09
*****************************************************************************/
ES_Event_t RunGameplayHSM( ES_Event_t CurrentEvent )
{
    switch ( CurrentState )
    {
        case IDLE_STATE : //IDLE STATE, wait for start button to start playing the game
             //if button has been pushed
                //Change to calibration state
        case CALIBRATION_STATE :
            //call during calibration state fxn

            //process any events
                   //If an event is active
                         //if event is to push the wall
                                //switch into the tape following state
                   //if timeout evennt

                        //if timeout comes from game timer
                            //if game has ended

                                //renter the idle state
        case TAPE_FOLLOW_STATE:

             //call during tape follow state
                   //If an event is active
                    //If event signals we have left the branch
                          //exited branch, enter switching branches state
                   //if timeout event
                        //if timeout comes from game timer
                          //if game has ended
                                //renter the idle state
        case SWITCHING_BRANCHES_STATE:
             //call during siwthcing branches state
                   //If an event is active

                    //if event is to push the wall
                        //enter the tape following state
                          //if timeout event
                                //if timeout comes from game timer
                                      //if game is ended
                                            //renter the idle state
    //   If we are making a state transition
       //   Execute exit function for current state
      //Modify state variable

       // Execute entry function for new state
       // this defaults to ES_ENTRY

}
/*****************************************************************************
 Function
     StartMasterSM

 Parameters
     ES_Event CurrentEvent

 Returns
     nothing
```

```
 Description
     Does any required initialization for this state machine
 Notes

 Author
     J. Edward Carryer, 02/06/12, 22:15
*****************************************************************************/
void StartGameplayHSM ( ES_Event_t CurrentEvent )
{
  // Begin in the idle state

  // now we need to let the Run function init the lower level state machines
  // use LocalEvent to keep the compiler from complaining about unused var
}


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

static ES_Event_t DuringIdleState( ES_Event_t Event)
{
       // assume no re-mapping or comsumption
    // process ES_ENTRY events

             //Stop Motors
             //turn off light 1
             //turn off light 2
             //set gameplay to false
      // process ES_EXIT events
             // set gameplay to true

             //Initiate Game Timer
}

static ES_Event_t DuringCalibrationState( ES_Event_t Event) //TODO
{
    // assme no re-mapping or comsumption

    // process ES_ENTRY events
      // pass ES_ ENTRY to calibration SM

  // Process ES_EXIT events
             // pass ES_EXIT to calibration SM

    // do the 'during' function for this state
}

static ES_Event_t DuringTapeFollowState( ES_Event_t Event)
{
       // assme no re-mapping or comsumption

    // process ES_ENTRY events
      // pass ES_ ENTRY to tape follow SM

  // Process ES_EXIT events
             // pass ES_EXIT to tape follow SM

    // do the 'during' function for this state
```

```
}

static ES_Event_t DuringSwitchingBranchesState( ES_Event_t Event) //TODO
{
// assme no re-mapping or comsumption

    // process ES_ENTRY events
       // pass ES_ ENTRY to switchingbranches SM

  // Process ES_EXIT events
             // pass ES_EXIT to switchingbranches SM

    // do the 'during' function for this state
}

static bool GameTime( void )
{
    //if one minute has elapsed
              //start timer for 1 minute

       //if two minutes have elapsed

              //start timer for 18 seconds

    //if 2:18 has elapsed
           //return true
}
```