```c
#include "mc9s12e128.h"
#include "ES_Configure.h"
#include "ES_General.h"
#include "ES_Events.h"
#include "ES_PostList.h"
#include "EventCheckers.h"


// This include will pull in all of the headers from the service modules
// providing the prototypes for all of the post functions
#include "ES_ServiceHeaders.h"
#include "AlignPPService.h"
#include "ArtilleryFSM.h"
#include "ResupplyService.h"
#include "StrategyFSM.h"
#include "DriveTrainService.h"

boolean CheckSPIF(void)
{
    boolean ReturnVal = False;
    ES_Event ThisEvent;
    unsigned char IncomingMessage;

    if (SPISR_SPIF != 0) //SPIF flag set
        {
            //clear flag
            SPISR;
            IncomingMessage = SPIDR;

            //post event to FAC_FSM
            ThisEvent.EventType = SPIF_SET;
            ThisEvent.EventParam = IncomingMessage;
            PostFAC_FSM(ThisEvent);
        }
    return ReturnVal;
}

boolean CheckPPAlignment(void)
{
    ES_Event MtrEvent, StrategyEvent;
    static unsigned char LastPinState = LO;
    unsigned char CurrentPinState;
    boolean ReturnVal = False;

    CurrentPinState = ALIGNPP_PORT; //poll state of pin associated with alignment

    if ( (CurrentPinState != LastPinState) && (CurrentPinState == HI) )
        {
            //Disable Interrupts once aligned _S12_C5I
            EnableIRDetection(False);

            //cancel IRDetect_TIMER
            ES_Timer_StopTimer(ALIGN_TIMER);
            puts("stopped timer\r\n");

            MtrEvent.EventType = STOP_MOTOR;
            MtrEvent.EventParam = 1;
            PostDriveTrainService(MtrEvent);

            StrategyEvent.EventType = DESTINATION_REACHED;
            StrategyEvent.EventParam = 0;
            PostStrategyFSM(StrategyEvent);
            puts("aligned with enemy pp\r\n");

            ALIGNPP_PORT = LO; //reset Port to LO
```

```
            ReturnVal = True;
        }

    LastPinState = CurrentPinState;
    return ReturnVal;
}


boolean CheckBackedUp(void)
{
    ES_Event MtrEvent, StrategyEvent;
    static unsigned char LastPinState = LO;
    unsigned char CurrentPinState;
    boolean ReturnVal = False;

    CurrentPinState = BACKUP_PORT;

    if ( (CurrentPinState != LastPinState) && (CurrentPinState == HI) && (QueryStrategyFSM() ==
Reloading) )
        {
            MtrEvent.EventType = STOP_MOTOR;
            MtrEvent.EventParam = 1;
            PostDriveTrainService(MtrEvent);
            puts("backed up to resupply depot \r\n");
            TurnOffPControl();   //Disable P-Control Interrupt

            StrategyEvent.EventType = READY2RELOAD;
            StrategyEvent.EventParam = 0;
            PostStrategyFSM(StrategyEvent);
            TurnOffPControl();   //Disable P-Control Interrupt

            ReturnVal = True;
        }

    LastPinState = CurrentPinState;
    return ReturnVal;
}

boolean CheckGameActive(void)
{
    ES_Event StrategyEvent;
    static unsigned char LastGameState = STOPPED;
    unsigned char CurrentGameState = STOPPED;
    boolean ReturnVal = False;

    CurrentGameState = QueryGameState();

    if ( (CurrentGameState != LastGameState) && (CurrentGameState == INPLAY) && (QueryStrategyFSM
() == Waiting4GameStart ))
        {

            StrategyEvent.EventType = GAME_START;
            StrategyEvent.EventParam = 0;
            PostStrategyFSM(StrategyEvent);
            puts("now in play \r\n");

            ReturnVal = True;
        }
    else if ( (CurrentGameState != LastGameState) && (CurrentGameState == STOPPED) &&
(QueryStrategyFSM() != Waiting4GameStart) )
        {
            StrategyEvent.EventType = GAME_OVER;
            StrategyEvent.EventParam = 0;
            PostStrategyFSM(StrategyEvent);

            ReturnVal = True;
        }
```

```
    LastGameState = CurrentGameState;
    return ReturnVal;
}

boolean CheckCamouflaged(void)
{
    ES_Event StrategyEvent;
    static unsigned char LastPosition = 1;
    unsigned char CurrentPosition;
    boolean ReturnVal = False;

    CurrentPosition = QueryX(SelfNum);

    if ( (CurrentPosition != LastPosition) && (CurrentPosition == 0) && (QueryStrategyFSM() ==
Reloading ))
        {

            StrategyEvent.EventType = RESUPPLY_COVER_REACHED;
            StrategyEvent.EventParam = 0;
            PostStrategyFSM(StrategyEvent);

            ReturnVal = True;
        }

    LastPosition = CurrentPosition;

    return ReturnVal;
}
```