****
****

Pseudo-code for CommMasterSM (An HSM to handle SSI communication with the LOC and pass information to the other services)

Module variables: CurrentCommand, LastCommand, ThisMessageType, LastMessageType

Events: ES_GAME_START, ES_QUERY_GAME_STATUS, ES_SEND_REPORT, ES_ACK, ES_NACK, ES_INACTIVE, ES_RECEIVE_REPORT

Defines: Clock defines, pin defines, Command defines: 0 = Status, 1 = SendReport, 2 = QueryReportResult

InitCommMasterSM
Takes a priority number, returns True.

Initialize the MyPriority variable with the passed in parameter
Set EventType of ThisEvent as ES_ENTRY
Run StartCommMasterSM with parameter ThisEvent
Return True

RunCommMasterSM
Takes ES_Event, returns ES_Event

Assign false to MakeTransition
Set Currentstate equal to the CommMasterState NextState
Assign ES_Event EntryEvenKind the EventType of ES_Entry and param of 0, which defaults to normal entry to new state
Assign ES_Event ReturnEvent the EventType of ES_NO_EVENT and param of 0, which assumes no error

```
Switch
     Case QueryGameStart
          /*Execute during for state 1 to. ES_ENTRY and ES_EXIT are
          run here to access lower level SM to remap or consume
          event*/
          CurrentEvent equals DuringQueryGameStart
          If current event doesn’t equal ES_NO_EVENT
               Switch
```

```
                        ES_EVENT is ES_GAME_START
                              NextState equals GameChat
                              MakeTransition equals true
                              Break
            End if
            break

      Case GameChat
            CurrentEvent equals DuringGameChat
            If current event doesn't equal ES_NO_EVENT
                  Switch
                        Case: ES_Event is ES_QUERY
                              NextState equals GameChat
                              MakeTransition equals true
                              Break
                        Case: ES_Event is ES_SEND_REPORT
                              //do we deal with these here
                        Case: ES_Event is ES_REPORT_RECEIVED
                              //do we deal with these here

                  Break
            End if
      Break

If MakeTransition equals true
      EventType of CurrentEvent is ES_EXIT
      RunCommMasterSM with parameter CurrentEvent
      CurrentState equals nextstate
      RunCommMasterSM with parameter EntryEventKind
End if

Return ReturnEvent
```

StartMasterSM (ES_Event CurrentEvent)
Takes a ES_Event, returns nothing

Since there is more than 1 state to the top level machine, initialize CurrentState to QueryGameStart
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
Run RunCommMasterSM
Return

Comm_PperiodicQuery_Init
Take nothing return nothing

Start by enabling the clock to the timer (Wide Timer 0)
Kill a few cycles to let the clock get going
Make sure that timer (Timer B) is disabled before configuring
Set it up in 32bit wide (individual, not concatenated) mode
Set up timer B in periodic mode so that it repeats the time-outs
Set timeout to 500mS
Enable a local timeout interrupt
Enable the Timer B in Wide Timer 0 interrupt in the NVIC. it is interrupt number 95 so appears in EN2 at bit 31
Make sure interrupts are enabled globally
Now kick the timer off by enabling it and enabling the timer to stall while stopped by the debugger
Edit interrupt priority register to make the periodic timer interrupt priority 2
End of Comm_Query_Init

Comm_PeriodicQuery_ISR
Takes nothing return nothing

Start by clearing the source of the interrupt
Locally enable interrupts (TXIM in SSIIM)
End of Comm_**Periodic**Query_ISR

QueryGameStatus (a helper function to write query and keep track of last Command)

Takes nothing, returns nothing

LastCommand is 0 for querying game status

Query game status from the slave by writing 0b1100 0000 followed by 4 bytes of 0x00
Locally enable interrupts for EOT
End of QueryGameStatus

SendFreqReport
Takes an unsigned 8 bit int, returns nothing

LastCommand is 1 for sending frequency response
Freq equals the passed in int
Freq equals freq masked to keep only the 4 LSB

Send 0b1000 0000 OR'd with Freq
Locally enable EOT interrupt
End of SendFreqReport

QueryReportResponse
Takes nothing, returns nothing

LastCommand [relates to whether last command was game status, send report, or report response]

Module defines:
GameStatus = 0
SendReport = 1
ReportResponse = 2

**********private functions**********
DuringQueryGameStart
Takes ES_Event, returns static ES_Event

ReturnEvent equals the passed in Event
If EventType of Event is ES_ENTRY or ES_ENTRY_HISTORY
    //Implement any entry actions required for this state machine
    // after that start any lower level machines that run in this state
    (No lower level machines to run)
End if

Else if EventType of Event is ES_EXIT
    No lower level SMs to clean up
    //any local exit functionality for QueryGameStart state
End else if

Else
    //Do the during function for this state
    Query Game Status
    If game status has changed
        Post ES_GAME_START
    End if
End else

Return Return Event

DuringGameChat

```
Takes ES_Event, returns static ES_Event

ReturnEvent equals the passed in Event
If EventType of Event is ES_ENTRY or ES_ENTRY_HISTORY
      //Implement any entry actions required for this state machine
      // after that start any lower level machines that run in this
      state
      Pass Event to StartWaiting2MS
      Pass EVent to StartCommActive
End if

Else if EventType of Event is ES_EXIT
      Pass Event to RunWaiting2MS
      Pass Event to RunCommActive
      //any local exit functionality for GameChat state?
End else if

Else (do the during function for this state)
      Run lower level state machines
            Return Event equals RunWaiting2MS
            Return Event equals RunCommActive
            //Do any activity that is to be repeated as long as we are
            in this state

      End if
End else

Return Return Event
```

****

Pseudo-code for GameChatSM (A flat SM to manage the communication and 2ms waiting time)

```
Switch based on CurrentState
           Case GameStatus
                 Pull off first two bytes of 0x00 and 0xFF
                 Save 3rd byte off FIFO as Status Byte 1
                 Save 4rd byte off FIFO as Status Byte 2
                 Save 5rd byte off FIFO as Status Byte 3
```

```
            Case SendReport
                  Pull off all 5 bytes, which have no meaning
            Case ReportResponse
                  Pull off first two bytes of 0x00 and 0xFF
                  Save 3rd byte to Response Ready
                  Save 4th byte to Report Status
                  Pull 5th byte of 0x00 off the stack
```

RunGameChatSM

```
Takes ann ES_Event CurrentEvent and returns an ES_Event

Set make transition equals false
Set GameChatState NextState to CurrentState
Initialize EntryEventKind to normal entry to state
Initialize ReturnEvent to Current Event, assuming no consuming

Switch based on CurrentState
      Case Waiting2MS
            Current event is result of CurrentEvent passed to
            DuringWaiting2MS
            If EventType of CurrentEvent is not ES_NO_EVENT
                  Switch based on CurrentEvent.EventType
                        Case 2ms_Timer_Expired
                              NextState equals CommActive
                              MakeTransition equals true
                              EntryEventKind

                              Consume event by setting EventType of
                              ReturnEvent to ES_NO_EVENT
                              break
            End if
            Else
                  Return event equals currentstate because current
                  event is now ES_NO_EVENT
            End else
      Break
If make transition equals true
      EventType of CurrentEvent is ES_EXIT
      RunGameChatSM with parameter as CurrentEvent
      CurrentState equals NextState
      RunGameChatSM with parameter as EntryEventKind
End if
```

```
Return ReturnEvent
```

StartGameChatSM

```
Takes an ES_Event CurrentEvent, returns nothing

If ES_ENTRY_HISTORY not equal to EventType of CurrentEvent
     Set CurrentState equal to ENTRY_STATE
End if

RunGameChatSM with parameter as CurrentEvent
End of StartGameChatSM
```

DuringWaiting2MS

```
Takes an ES_Event Event, returns ES_EVENT

Set ES_Event ReturnEvent equal to Event
If EventType of Event is not ES_ENTRY or ES_ENTRY_HISTORY
     //implement entry actions for this state machine
     (no lower level SM to run)
End if

Else if EventType of Event is ES_EXIT
     (no lower level states to run clean up)
     //Do any local exit functionality
End if

Else do the during function for this state
     (no lower level SM to run)
     //do any activity to is repeated as long as we are in this
state
End if

Return ReturnEvent
```

DuringCommActive

```
Takes an ES_Event Event, returns ES_EVENT

Set ES_Event ReturnEvent equal to Event
If EventType of Event is not ES_ENTRY or ES_ENTRY_HISTORY
     //implement entry actions for this state machine
     (no lower level SM to run)
End if
```

```
Else if EventType of Event is ES_EXIT
      (no lower level states to run clean up)
      //Do any local exit functionality
End if

Else do the during function for this state
      (no lower level SM to run)
      //do any activity to is repeated as long as we are in this
state

      //add content here




End if

Return ReturnEvent
```

2 ms one shot safety timer init

```
Takes nothing and returns nothing
      start by enabling the clock to the timer (Wide Timer 1)
       Loop until timer hardware is ready
       Disable timer A before configuring
       Configure timer for 32bit (individual instead of concatenated)
       Macro define 16bit refers to individual timer rather than
actual 16bit
       Set Timer a into one-shot mode (mask bits 0:1 and write value
for 1-shot mode = 0x01)
      set timeout to 2 ms
      Enable local timeout interrupt. (TATOIM = bit 0 maybe not?)
      Enable interrupt in NVIC register; we have interrupt 96 so
nvic_en3
      Change priority of one-shot to 0; we are using pri 24
      Turn on interrupts globally
     set timer to stall in debugging. We will wait until the start
function to start the timer
```

2 ms one shot ISR

```
      clear source of interrupt
      Go to CommActive state by posting the 2ms_Timer_Expired event
      to comm service (
```

GetGreenCheckInShoot
Return bit 7 of Status Byte 1
End of GetGreenCheckInShoot

GetGreenActiveArea
Return bits 4-6 of Status Byte 1
End of GetGreenActiveArea

GetGreenScore
Return bits 0-5 of Status Byte 2
End of GetGreenScore

GetRedActiveArea
Return bit 0-2 of Status Byte 1
End of GetRedActiveArea

GetRedCheckInShoot
Return bit 3 of Status Byte 1

End of GetRedCheckInShoot

GetRedScore
Return Bits 0-5 of Status Byte 3
End of GetRedScore

GetGameStatus
Return by 7 of Status Byte 3
End of GetGameStatus

GetResponseReady
Return bits 0-7 of Response Ready Byte
End of GetResponseReady

GetReportResponse
Return bits 6-7 of Report Status Byte
End of GetReportResponse

GetNextLocation
Return bits 0-3 of Report Status Byte
End of GetNextLocation