

StageSM pseudo code

```
/*----- Include Files -----*/
// Basic includes for a program using the Events and Services Framework
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "StageSM.h"
#include "Location.h"
#include "MasterVehicle.h"
#include "LOCMaster.h"

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

/*----- Module Variables -----*/
set up a variable to store the state

set up variables to store destinations
/*----- Module Code -----*/
/*****

*****/
ES_Event RunStageSM( ES_Event CurrentEvent )
{
    default to not make a transition
    default to normal entry to new state
    default to not consume events

    switch ( CurrentState )
    {
        case STAGE_WAITING :
            execute during function for "STAGE_WAITING"

            If an event is active
            {
                switch EventType
                {
                    case STAGE_ACTIVE :
                        create a local variable to store the destination index
                        (for our array of coordinates), dest_index
                        query the active stage and update dest_index
                        if the dest_index indicates a valid location (1,2,or 3)
                            update the current X and Y destination that we are going to
                            set NextState to STAGE_MOVE_Y, start with first
                            adjusting current robot Y position
                            mark that we are making a transition
                            consume event

                        default:
                            break;
                    }
                }
            }
            break;

        case STAGE_MOVE_Y :
            execute during function "STAGE_MOVE_Y"

            If an event is active
            {
                switch Event Type
                {
                    case SHOOT_ACTIVE_4://all the stations are active for shooting
```

```

        set next state to STAGE_WAITING
        mark that we are making a transition
        do not consum ethis event
        break;

    case Y_REACHED :
        set NextState to STAGE_MOVE_X to start moving in X direction
        mark that we are making transition
        consume the event
        break;

        case CONSTRUCTION_END:
            set the NextState to be STAGE_WAITING
            mark that we are making a transition
            do not consum this event
            break;
        default:
            break;
    }
}
break;

case STAGE_MOVE_X :
    execute during function for "STAGE_MOVE_X"

    If an event is active
    {
        switch Event Type
        {
            case SHOOT_ACTIVE_4:
                set NextState to STAGE_WAITING
                mark that we are taking a transition
                do not consume the event
                break;

            case X_REACHED:
                if our Y location is still good
                    set NextState to STAGE_VERIFICATION indicating that we are ready for
handshake
                else
                    set NextState to STAGE_MOVE_Y to adjust Y position again

                    mark that we are making a transition
                    consume the event
                    break;

                    case CONSTRUCTION_END:
                        set NextState to STAGE_WAITING
                        mark that we are taking a transition
                        do no consume the event
                        break;

                    default:
                        break;
                }
            }
        }
    break;

case STAGE_VERIFICATION :
    execute during function for "STAGE_VERIFICATION"
    If an event is active

```

```

        {
            switch Event Type
            {
                case ES_TIMEOUT:
                    if the stage_timer times out,
                    indicating we got stuck during the handshake
                        set NextState to be STAGE_WAITING
                        mark that we are making a transition
                        do not consume the event
                        post a "restart verify frequency" event to the LOCMaster
                        break

                case FINISHED_STAGING:
                    set NextState to STAGE_WAITING
                    mark that we are taking a transition
                    do not consume the event
                    break;

                case CONSTRUCTION_END:
                    set NextState to STAGE_WAITING
                    mark that we are taking a transition
                    do not consume the event
                    break;

                default:
                    break;
            }
        }
        break;

    default:
        break;
}

If we are making a state transition
{
    Execute exit function for current state
    Modify state variable
    Execute entry function for new state
}

return(ReturnEvent);
}
/*****/
void StartStageSM ( ES_Event CurrentEvent )
{
    set the first state to the entry state
    run the state machine with an entry event
}

/*****/
StagingState_t QueryStageSM ( void )
{
    return(CurrentState);
}

```

```

/*****
static ES_Event DuringWaiting( ES_Event Event)
{
    ES_Event ReturnEvent = Event; // assume no re-mapping or consumption
    return(ReturnEvent);
}

static ES_Event DuringMoveY( ES_Event Event)
{
    ES_Event ReturnEvent = Event; // assume no re-mapping or consumption

    // process ES_ENTRY, ES_ENTRY_HISTORY & ES_EXIT events
    if we have ES_ENTRY or ES_ENTRY_HISTORY
        move the Y to the current destination
    else if we have ES_EXIT
        stop the motor

    return(ReturnEvent);
}

static ES_Event DuringMoveX( ES_Event Event)
{
    ES_Event ReturnEvent = Event; // assume no re-mapping or consumption

    // process ES_ENTRY, ES_ENTRY_HISTORY & ES_EXIT events
    if we have ES_ENTRY or ES_ENTRY_HISTORY
        move the X to the current destination
    else if we have ES_EXIT
        stop the motor

    return(ReturnEvent);
}

static ES_Event DuringFreqMeasurement( ES_Event Event)
{
    ES_Event ReturnEvent = Event; // assume no re-mapping or consumption

    // process ES_ENTRY, ES_ENTRY_HISTORY & ES_EXIT events
    if we have ES_ENTRY or ES_ENTRY_HISTORY
        post an "ARRIVED_AT_STAGING" event to LOCMaster
        start a 3 sec "STAGE_TIMER" (if we do not finish handshaking during this time, we will
restart the handshaking

    return(ReturnEvent);
}

static ES_Event DuringWaitingForResponse( ES_Event Event)
{
    ES_Event ReturnEvent = Event; // assume no re-mapping or consumption
    return(ReturnEvent);
}

```