```
/*****************************************************************************
 Module
   EventCheckers.c

 Revision
   1.0.1

 Description
   This is the sample for writing event checkers along with the event
   checkers used in the basic framework test harness.

 Notes
   Note the use of static variables in sample event checker to detect
   ONLY transitions.

 History
 When           Who     What/Why
 -------------- ---     --------
 08/06/13 13:36 jec     initial version
*****************************************************************************/

// this will pull in the symbolic definitions for events, which we will want
// to post in response to detecting events
#include "ES_Configure.h"
// this will get us the structure definition for events, which we will need
// in order to post events in response to detecting events
#include "ES_Events.h"
// if you want to use distribution lists then you need those function
// definitions too.
#include "ES_PostList.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"
// this test harness for the framework references the serial routines that
// are defined in ES_Port.c
#include "ES_Port.h"
// include our own prototypes to insure consistency between header &
// actual functionsdefinition
#include "EventCheckers.h"


// This is the event checking function sample. It is not intended to be
// included in the module. It is only here as a sample to guide you in
writing
// your own event checkers



//#include "ES_General.h"


bool Dummy_Check(void)
{
  bool ReturnVal = false;
```

```
  return ReturnVal;
}



/****************************************************************************
 Function
   Check4Keystroke
 Parameters
   None
 Returns
   bool: true if a new key was detected & posted
 Description
   checks to see if a new key from the keyboard is detected and, if so,
   retrieves the key and posts an ES_NewKey event to TestHarnessService1
 Notes
   The functions that actually check the serial hardware for characters
   and retrieve them are assumed to be in ES_Port.c
   Since we always retrieve the keystroke when we detect it, thus clearing
the
   hardware flag that indicates that a new key is ready this event checker
   will only generate events on the arrival of new characters, even though we
   do not internally keep track of the last keystroke that we retrieved.
 Author
   J. Edward Carryer, 08/06/13, 13:48
****************************************************************************/
bool Check4Keystroke(void)
{
    char KeyChar;
    bool ReturnValue = false;


  if ( IsNewKeyReady() ) // new key waiting?
  {
    ES_Event ThisEvent;
    ThisEvent.EventType = ES_NEW_KEY;
    ThisEvent.EventParam = 1;


    ReturnValue = true;

    KeyChar = GetNewKey();
    printf("Key pressed: %c\n\r", KeyChar);

    switch (KeyChar)
    {

                        case 'q' :

                                ThisEvent.EventType = pas_d_armes;

                                PostMasterHSM(ThisEvent);

                                break;

                        case 'w' :
```

```
                ThisEvent.EventType = recess;
                PostMasterHSM(ThisEvent);
                break;
            case 'e' :
                ThisEvent.EventType = end_of_match;
                PostMasterHSM(ThisEvent);
                break;
            case 'r' :
                ThisEvent.EventType = unhorsed;
                PostMasterHSM(ThisEvent);
                break;
            case 't' :
                ThisEvent.EventType = sudden_death;
                PostMasterHSM(ThisEvent);
                break;
            case 'y' :
                ThisEvent.EventType = found_goal;
                PostMasterHSM(ThisEvent);
                break;
            case 'u' :
                ThisEvent.EventType = lost_goal;
                PostMasterHSM(ThisEvent);
                break;
            case 'i' :
                ThisEvent.EventType = found_knight_fwd;
                PostMasterHSM(ThisEvent);
                break;
        case 'o' :
```

```
                ThisEvent.EventType = lost_knight_fwd;
                PostMasterHSM(ThisEvent);
                break;
            case 'p' :
                ThisEvent.EventType = found_knight_rvs;
                PostMasterHSM(ThisEvent);
                break;
            case 'a' :
                ThisEvent.EventType = lost_knight_rvs;
                PostMasterHSM(ThisEvent);
                break;
            case 'j' :
                ThisEvent.EventType = Crossed_Home_Fwd_Offset;
                PostMasterHSM(ThisEvent);
                break;
            case 'k' :
                ThisEvent.EventType = Crossed_Home_Rvs_Offset;
                PostMasterHSM(ThisEvent);
                break;

        case 'l' :
                ThisEvent.EventType = STOP;
                PostMotorService(ThisEvent);
                break;
        case 'z' :
                ThisEvent.EventType = TurnLeft;
                PostMotorService(ThisEvent);
                break;
        case 'x' :
```

```
                ThisEvent.EventType = TurnRight;
                PostMotorService(ThisEvent);
                break;
        case 'c' :
                ThisEvent.EventType = ChargingFieldFwd;
                PostMotorService(ThisEvent);
                break;
        case 'v' :
                ThisEvent.EventType = FindingWallFwd;
                PostMotorService(ThisEvent);
                break;
        case 'b' :
                ThisEvent.EventType = GoingToStartFwd;
                PostMotorService(ThisEvent);
                break;
        case 'n' :
                ThisEvent.EventType = GoingToStartRvs;
                PostMotorService(ThisEvent);
                break;
        case 'm' :
                ThisEvent.EventType = ChargingFieldRvs;
                PostMotorService(ThisEvent);
                break;
        case 'Q' :
                ThisEvent.EventType = VeerLeftFwd;
                PostMotorService(ThisEvent);
                break;
        case 'W' :
                ThisEvent.EventType = VeerRightFwd;
```

```
                    PostMotorService(ThisEvent);
                    break;
            case 'E' :
                    ThisEvent.EventType = VeerLeftRvs;
                    PostMotorService(ThisEvent);
                    break;
            case 'R' :
                    ThisEvent.EventType = VeerRightRvs;
                    PostMotorService(ThisEvent);
                    break;

                default:
                    printf("No corresponding event for key\n\r");
                    break;
    }

  }
  return ReturnValue;
}
```