ME218A LAB 4 Pei-Chen, Wu SuNet: pcwu1023

Part 0: Pre-Lab

* What snippet of C code will you use to test if the input signal has changed states?

bool CheckMorseEvents(void)

{

static uint8\_t LastInputState = 0x0;

uint8\_t CurrentInputState;

bool ReturnVal = false;

ES\_Event MorseEvent;

if((HWREG(GPIO\_PORTF\_BASE + (GPIO\_O\_DATA + ALL\_BITS)) & BIT2HI)) CurrentInputState = 1;

else CurrentInputState = 0;

if(CurrentInputState != LastInputState){

ReturnVal = true;

if(CurrentInputState != 0){

MorseEvent.EventType = RisingEdge;

MorseEvent.EventParam = ES\_Timer\_GetTime();

printf("Rising Time : %d", ES\_Timer\_GetTime());

PostMorseElementFSM(MorseEvent);

}

else{

MorseEvent.EventType = FallingEdge;

MorseEvent.EventParam = ES\_Timer\_GetTime();

printf("Falling Time : %d", ES\_Timer\_GetTime());

PostMorseElementFSM(MorseEvent);

}

}

LastInputState = CurrentInputState;

return ReturnVal;

}

* What snippet of C code will you use to test if the length of a high or low interval represents a legal dot-space interval?

static void TestCalibration(void){

uint16\_t SecondDelta;

ES\_Event ThisEvent;

if(FirstDelta == 0) {FirstDelta = TimeOfLastFall - TimeOfLastRise; printf(" FirstDelta : %d", FirstDelta);}

else {

SecondDelta = TimeOfLastFall - TimeOfLastRise;

printf(" SecondDelta : %d", SecondDelta);

if((100\*FirstDelta / SecondDelta) <= 33){

LengthOfDot = FirstDelta;

printf(" LengthOfDot : %d", FirstDelta);

ThisEvent.EventType = CalibrationCompleted;

ThisEvent.EventParam = 1;

PostMorseElementFSM(ThisEvent);

}

else if((100\*FirstDelta / SecondDelta) >= 300){

LengthOfDot = SecondDelta;

ThisEvent.EventType = CalibrationCompleted;

ThisEvent.EventParam = 1;

PostMorseElementFSM(ThisEvent);

printf(" LengthOfDot : %d", SecondDelta);

}

else FirstDelta = SecondDelta;

}

}

* What data structure do you propose to use to represent the morse character as it is being received?

static char MorseString[StringLength];

* What snippet of C code will you use to add a dot to that structure?

static bool MorseStringHasSpace(void){

return StringLength-1-strlen(MorseString);

}

if(MorseStringHasSpace()){

printf("add dash");

strcat(MorseString, "-");

}

* What does this line of C code do? (Assume the GetSignal() returns the state of an input pin) for (signal = GetSignal(); signal == (new\_signal = GetSignal()););
* **The first line will read the state of the input pin, and store it into the variable called “signal”. Then the loop will keep running until the signal changes, it breaks the loop.**
* Propose a more readable and understandable piece of C code do this same function.

signal = GetSignal();

new\_signal=signal;

while(new\_signal == signal)

{

new\_signal = GetSignal();

}

Part 1: A Morse Code Receiver

Circuit for the experiment.



Corner frequency = 0.0144Hz for RC high pass filter.

In transresistive circuit, I use offset about 2.5V, therefore, my hysterisis is 2.4 - 2.6.

For amplifying the output voltage of IR sensor, I use 220k to amplify the output voltage, which in the end is equal to 1.18 at lower peak and 3.11 at higher peak.

The code is presented in the following pages. I include Lab3 LCD.h and LCD.c in the project. I don’t change anything, but I still attached those code in the end of report.

I paste the psoedo – code in Appendix, cause I didn’t do significant modify for that, only one part in MorseElements module.

In CharacterizeSpace, when it detects EOC, I don’t post the event to both services, instead I use following code.

if(CurrentState != EOC\_WaitFall) ES\_PostList02(ThisEvent); //decode

PostMorseElementFSM(ThisEvent);

printf("EOCDetected");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

AllEventCheckers.h

Description

header file for the event checking functions

Notes

History

When Who What/Why

-------------- --- --------

08/06/13 14:37 jec started coding

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef AllEventCheckers\_H

#define AllEventCheckers\_H

// prototypes for event checkers

#include "EventCheckers.h"

#include "ButtonFSM.h"

#endif /\* EventCheckers\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

ES\_Configure.h

Description

This file contains macro definitions that are edited by the user to

adapt the Events and Services framework to a particular application.

Notes

History

When Who What/Why

-------------- --- --------

10/21/13 20:54 jec lots of added entries to bring the number of timers

and services up to 16 each

08/06/13 14:10 jec removed PostKeyFunc stuff since we are moving that

functionality out of the framework and putting it

explicitly into the event checking functions

01/15/12 10:03 jec started coding

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef CONFIGURE\_H

#define CONFIGURE\_H

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// The maximum number of services sets an upper bound on the number of

// services that the framework will handle. Reasonable values are 8 and 16

// corresponding to an 8-bit(uint8\_t) and 16-bit(uint16\_t) Ready variable size

#define MAX\_NUM\_SERVICES 16

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// This macro determines that nuber of services that are \*actually\* used in

// a particular application. It will vary in value from 1 to MAX\_NUM\_SERVICES

#define NUM\_SERVICES 3

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 0, the lowest priority service.

// Every Events and Services application must have a Service 0. Further

// services are added in numeric sequence (1,2,3,...) with increasing

// priorities

// the header file with the public function prototypes

#define SERV\_0\_HEADER "ButtonFSM.h"

// the name of the Init function

#define SERV\_0\_INIT InitButtonFSM

// the name of the run function

#define SERV\_0\_RUN RunButtonFSM

// How big should this services Queue be?

#define SERV\_0\_QUEUE\_SIZE 5

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// The following sections are used to define the parameters for each of the

// services. You only need to fill out as many as the number of services

// defined by NUM\_SERVICES

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 1

#if NUM\_SERVICES > 1

// the header file with the public function prototypes

#define SERV\_1\_HEADER "MorseElementFSM.h"

// the name of the Init function

#define SERV\_1\_INIT InitMorseElementFSM

// the name of the run function

#define SERV\_1\_RUN RunMorseElementFSM

// How big should this services Queue be?

#define SERV\_1\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 2

#if NUM\_SERVICES > 2

// the header file with the public function prototypes

#define SERV\_2\_HEADER "MorseDecodeFSM.h"

// the name of the Init function

#define SERV\_2\_INIT InitMorseDecodeFSM

// the name of the run function

#define SERV\_2\_RUN RunMorseDecodeFSM

// How big should this services Queue be?

#define SERV\_2\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 3

#if NUM\_SERVICES > 3

// the header file with the public function prototypes

#define SERV\_3\_HEADER "TestHarnessService3.h"

// the name of the Init function

#define SERV\_3\_INIT InitTestHarnessService3

// the name of the run function

#define SERV\_3\_RUN RunTestHarnessService3

// How big should this services Queue be?

#define SERV\_3\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 4

#if NUM\_SERVICES > 4

// the header file with the public function prototypes

#define SERV\_4\_HEADER "TestHarnessService4.h"

// the name of the Init function

#define SERV\_4\_INIT InitTestHarnessService4

// the name of the run function

#define SERV\_4\_RUN RunTestHarnessService4

// How big should this services Queue be?

#define SERV\_4\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 5

#if NUM\_SERVICES > 5

// the header file with the public function prototypes

#define SERV\_5\_HEADER "TestHarnessService5.h"

// the name of the Init function

#define SERV\_5\_INIT InitTestHarnessService5

// the name of the run function

#define SERV\_5\_RUN RunTestHarnessService5

// How big should this services Queue be?

#define SERV\_5\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 6

#if NUM\_SERVICES > 6

// the header file with the public function prototypes

#define SERV\_6\_HEADER "TestHarnessService6.h"

// the name of the Init function

#define SERV\_6\_INIT InitTestHarnessService6

// the name of the run function

#define SERV\_6\_RUN RunTestHarnessService6

// How big should this services Queue be?

#define SERV\_6\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 7

#if NUM\_SERVICES > 7

// the header file with the public function prototypes

#define SERV\_7\_HEADER "TestHarnessService7.h"

// the name of the Init function

#define SERV\_7\_INIT InitTestHarnessService7

// the name of the run function

#define SERV\_7\_RUN RunTestHarnessService7

// How big should this services Queue be?

#define SERV\_7\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 8

#if NUM\_SERVICES > 8

// the header file with the public function prototypes

#define SERV\_8\_HEADER "TestHarnessService8.h"

// the name of the Init function

#define SERV\_8\_INIT InitTestHarnessService8

// the name of the run function

#define SERV\_8\_RUN RunTestHarnessService8

// How big should this services Queue be?

#define SERV\_8\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 9

#if NUM\_SERVICES > 9

// the header file with the public function prototypes

#define SERV\_9\_HEADER "TestHarnessService9.h"

// the name of the Init function

#define SERV\_9\_INIT InitTestHarnessService9

// the name of the run function

#define SERV\_9\_RUN RunTestHarnessService9

// How big should this services Queue be?

#define SERV\_9\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 10

#if NUM\_SERVICES > 10

// the header file with the public function prototypes

#define SERV\_10\_HEADER "TestHarnessService10.h"

// the name of the Init function

#define SERV\_10\_INIT InitTestHarnessService10

// the name of the run function

#define SERV\_10\_RUN RunTestHarnessService10

// How big should this services Queue be?

#define SERV\_10\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 11

#if NUM\_SERVICES > 11

// the header file with the public function prototypes

#define SERV\_11\_HEADER "TestHarnessService11.h"

// the name of the Init function

#define SERV\_11\_INIT InitTestHarnessService11

// the name of the run function

#define SERV\_11\_RUN RunTestHarnessService11

// How big should this services Queue be?

#define SERV\_11\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 12

#if NUM\_SERVICES > 12

// the header file with the public function prototypes

#define SERV\_12\_HEADER "TestHarnessService12.h"

// the name of the Init function

#define SERV\_12\_INIT InitTestHarnessService12

// the name of the run function

#define SERV\_12\_RUN RunTestHarnessService12

// How big should this services Queue be?

#define SERV\_12\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 13

#if NUM\_SERVICES > 13

// the header file with the public function prototypes

#define SERV\_13\_HEADER "TestHarnessService13.h"

// the name of the Init function

#define SERV\_13\_INIT InitTestHarnessService13

// the name of the run function

#define SERV\_13\_RUN RunTestHarnessService13

// How big should this services Queue be?

#define SERV\_13\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 14

#if NUM\_SERVICES > 14

// the header file with the public function prototypes

#define SERV\_14\_HEADER "TestHarnessService14.h"

// the name of the Init function

#define SERV\_14\_INIT InitTestHarnessService14

// the name of the run function

#define SERV\_14\_RUN RunTestHarnessService14

// How big should this services Queue be?

#define SERV\_14\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for Service 15

#if NUM\_SERVICES > 15

// the header file with the public function prototypes

#define SERV\_15\_HEADER "TestHarnessService15.h"

// the name of the Init function

#define SERV\_15\_INIT InitTestHarnessService15

// the name of the run function

#define SERV\_15\_RUN RunTestHarnessService15

// How big should this services Queue be?

#define SERV\_15\_QUEUE\_SIZE 3

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Name/define the events of interest

// Universal events occupy the lowest entries, followed by user-defined events

typedef enum { ES\_NO\_EVENT = 0,

ES\_ERROR, /\* used to indicate an error from the service \*/

ES\_INIT, /\* used to transition from initial pseudo-state \*/

ES\_TIMEOUT, /\* signals that the timer has expired \*/

/\* User-defined events start here \*/

ES\_NEW\_KEY, /\* signals a new key received from terminal \*/

ES\_LOCK,

ES\_UNLOCK,

BUTTON\_DOWN,

BUTTON\_UP,

DBButtonUp,

DBButtonDown,

RisingEdge,

FallingEdge,

CalibrationCompleted,

EOCDetected,

EOWDetected,

BadSpace,

DotDetected,

DashDetected,

BadPulse,

} ES\_EventTyp\_t ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for the Distribution lists. Each definition

// should be a comma separated list of post functions to indicate which

// services are on that distribution list.

#define NUM\_DIST\_LISTS 3

#if NUM\_DIST\_LISTS > 0

#define DIST\_LIST0 PostButtonFSM

#endif

#if NUM\_DIST\_LISTS > 1

#define DIST\_LIST1 PostMorseElementFSM, PostMorseDecodeFSM

#endif

#if NUM\_DIST\_LISTS > 2

#define DIST\_LIST2 PostMorseDecodeFSM

#endif

#if NUM\_DIST\_LISTS > 3

#define DIST\_LIST3 PostTemplateFSM

#endif

#if NUM\_DIST\_LISTS > 4

#define DIST\_LIST4 PostTemplateFSM

#endif

#if NUM\_DIST\_LISTS > 5

#define DIST\_LIST5 PostTemplateFSM

#endif

#if NUM\_DIST\_LISTS > 6

#define DIST\_LIST6 PostTemplateFSM

#endif

#if NUM\_DIST\_LISTS > 7

#define DIST\_LIST7 PostTemplateFSM

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// This are the name of the Event checking funcion header file.

#define EVENT\_CHECK\_HEADER "AllEventCheckers.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// This is the list of event checking functions

#define EVENT\_CHECK\_LIST CheckButtonEvents, CheckMorseEvents

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These are the definitions for the post functions to be executed when the

// corresponding timer expires. All 16 must be defined. If you are not using

// a timer, then you should use TIMER\_UNUSED

// Unlike services, any combination of timers may be used and there is no

// priority in servicing them

#define TIMER\_UNUSED ((pPostFunc)0)

#define TIMER0\_RESP\_FUNC PostButtonFSM

#define TIMER1\_RESP\_FUNC TIMER\_UNUSED

#define TIMER2\_RESP\_FUNC TIMER\_UNUSED

#define TIMER3\_RESP\_FUNC TIMER\_UNUSED

#define TIMER4\_RESP\_FUNC TIMER\_UNUSED

#define TIMER5\_RESP\_FUNC TIMER\_UNUSED

#define TIMER6\_RESP\_FUNC TIMER\_UNUSED

#define TIMER7\_RESP\_FUNC TIMER\_UNUSED

#define TIMER8\_RESP\_FUNC TIMER\_UNUSED

#define TIMER9\_RESP\_FUNC TIMER\_UNUSED

#define TIMER10\_RESP\_FUNC TIMER\_UNUSED

#define TIMER11\_RESP\_FUNC TIMER\_UNUSED

#define TIMER12\_RESP\_FUNC TIMER\_UNUSED

#define TIMER13\_RESP\_FUNC TIMER\_UNUSED

#define TIMER14\_RESP\_FUNC TIMER\_UNUSED

#define TIMER15\_RESP\_FUNC TIMER\_UNUSED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Give the timer numbers symbolc names to make it easier to move them

// to different timers if the need arises. Keep these definitions close to the

// definitions for the response functions to make it easier to check that

// the timer number matches where the timer event will be routed

// These symbolic names should be changed to be relevant to your application

#define SERVICE0\_TIMER 15

#endif /\* CONFIGURE\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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"

#include <stdbool.h>

#include "termio.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

//#include "inc/hw\_sysctl.h"

#include "driverlib/gpio.h"

#include "ES\_Timers.h"

#define ALL\_BITS (0xff << 2)

// 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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

CheckMorseEvents

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 TestHarnessService0

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 CheckMorseEvents(void)

{

static uint8\_t LastInputState = 0x0;

uint8\_t CurrentInputState;

bool ReturnVal = false;

ES\_Event MorseEvent;

if((HWREG(GPIO\_PORTF\_BASE + (GPIO\_O\_DATA + ALL\_BITS)) & BIT2HI)) CurrentInputState = 1;

else CurrentInputState = 0;

if(CurrentInputState != LastInputState){

ReturnVal = true;

if(CurrentInputState != 0){

MorseEvent.EventType = RisingEdge;

MorseEvent.EventParam = ES\_Timer\_GetTime();

printf("Rising Time : %d", ES\_Timer\_GetTime());

PostMorseElementFSM(MorseEvent);

}

else{

MorseEvent.EventType = FallingEdge;

MorseEvent.EventParam = ES\_Timer\_GetTime();

printf("Falling Time : %d", ES\_Timer\_GetTime());

PostMorseElementFSM(MorseEvent);

}

}

LastInputState = CurrentInputState;

return ReturnVal;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

EventCheckers.h

Description

header file for the event checking functions

Notes

History

When Who What/Why

-------------- --- --------

08/06/13 14:37 jec started coding

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef EventCheckers\_H

#define EventCheckers\_H

// prototypes for event checkers

bool CheckMorseEvents(void);

#endif /\* EventCheckers\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

ButtonFSM.c

Revision

1.0.1

Description

This is a template file for implementing flat state machines under the

Gen2 Events and Services Framework.

Notes

History

When Who What/Why

-------------- --- --------

01/15/12 11:12 jec revisions for Gen2 framework

11/07/11 11:26 jec made the queue static

10/30/11 17:59 jec fixed references to CurrentEvent in RunTemplateSM()

10/23/11 18:20 jec began conversion from SMTemplate.c (02/20/07 rev)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*----------------------------- 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 <stdbool.h>

#include <stdio.h>

#include "ES\_Configure.h"

#include "ES\_Framework.h"

#include "ButtonFSM.h"

#include "ES\_Events.h"

#include "termio.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

#include "inc/hw\_sysctl.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "ES\_Timers.h"

/\*----------------------------- Module Defines ----------------------------\*/

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

/\* prototypes for private functions for this machine.They should be functions

relevant to the behavior of this state machine

\*/

/\*---------------------------- Module Variables ---------------------------\*/

// everybody needs a state variable, you may need others as well.

// type of state variable should match htat of enum in header file

static ButtonState\_t CurrentState;

static uint8\_t LastButtonState;

// with the introduction of Gen2, we need a module level Priority var as well

static uint8\_t MyPriority;

#define ALL\_BITS (0xff << 2)

#define Debouncing\_time 100

/\*------------------------------ Module Code ------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

InitTemplateFSM

Parameters

uint8\_t : the priorty of this service

Returns

bool, false if error in initialization, true otherwise

Description

Saves away the priority, sets up the initial transition and does any

other required initialization for this state machine

Notes

Author

J. Edward Carryer, 10/23/11, 18:55

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Register\_MONITOR\_BUTTONPins(void){

HWREG(SYSCTL\_RCGCGPIO) |= SYSCTL\_RCGCGPIO\_R5;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_DEN) |= BIT3HI;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_DIR) &= BIT3LO;

}

bool InitButtonFSM ( uint8\_t Priority )

{

ES\_Event ThisEvent;

MyPriority = Priority;

Register\_MONITOR\_BUTTONPins();

LastButtonState = 0;

if((HWREG(GPIO\_PORTF\_BASE + (GPIO\_O\_DATA + ALL\_BITS)) & BIT3HI))

LastButtonState = 1;

CurrentState = DEBOUNCING;

ES\_Timer\_SetTimer(0, Debouncing\_time);

ES\_Timer\_StartTimer(0);

ThisEvent.EventType = ES\_INIT;

if (ES\_PostToService( MyPriority, ThisEvent) == true) return true;

else return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

PostTemplateFSM

Parameters

EF\_Event ThisEvent , the event to post to the queue

Returns

boolean False if the Enqueue 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 PostButtonFSM( ES\_Event ThisEvent )

{

return ES\_PostToService( MyPriority, ThisEvent);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

RunTemplateFSM

Parameters

ES\_Event : the event to process

Returns

ES\_Event, ES\_NO\_EVENT if no error ES\_ERROR otherwise

Description

add your description here

Notes

uses nested switch/case to implement the machine.

Author

J. Edward Carryer, 01/15/12, 15:23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ES\_Event RunButtonFSM( ES\_Event ThisEvent )

{

ES\_Event ReturnEvent;

ReturnEvent.EventType = ES\_NO\_EVENT; // assume no errors

if(CurrentState == DEBOUNCING){

if(ThisEvent.EventType == ES\_TIMEOUT && ThisEvent.EventParam == 0) CurrentState = Ready2Sample;

}

else if(CurrentState == Ready2Sample){

printf("Ready2Sample");

if(ThisEvent.EventType == BUTTON\_UP){

ES\_Timer\_SetTimer(0, Debouncing\_time);

ES\_Timer\_StartTimer(0);

CurrentState = DEBOUNCING;

ES\_Event NEW\_EVENT;

NEW\_EVENT.EventType = DBButtonUp;

ES\_PostList01(NEW\_EVENT);

}

if(ThisEvent.EventType == BUTTON\_DOWN){

ES\_Timer\_SetTimer(0, Debouncing\_time);

ES\_Timer\_StartTimer(0);

CurrentState = DEBOUNCING;

ES\_Event NEW\_EVENT;

NEW\_EVENT.EventType = DBButtonDown;

ES\_PostList01(NEW\_EVENT);

}

}

return ReturnEvent;

}

bool CheckButtonEvents(void)

{

uint8\_t CurrentButtonState;

if(HWREG(GPIO\_PORTF\_BASE + (GPIO\_O\_DATA + ALL\_BITS)) & BIT3HI) CurrentButtonState = 1;

else CurrentButtonState = 0;

bool ReturnVal = false;

ES\_Event ButtonEvent;

if (CurrentButtonState != LastButtonState)

{

printf("BUTTON");

ReturnVal = true;

if(CurrentButtonState != 0){

ButtonEvent.EventType = BUTTON\_DOWN;

ButtonEvent.EventParam = 0;

PostButtonFSM(ButtonEvent);

printf("BUTTON DOWN");

}

else{

ButtonEvent.EventType = BUTTON\_UP;

ButtonEvent.EventParam = 0;

PostButtonFSM(ButtonEvent);

printf("BUTTON UP");

}

}

LastButtonState = CurrentButtonState;

return ReturnVal;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Header file for template Flat Sate Machine

based on the Gen2 Events and Services Framework

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef FSMButton\_H

#define FSMButton\_H

// Event Definitions

#include "ES\_Configure.h" /\* gets us event definitions \*/

#include "ES\_Types.h" /\* gets bool type for returns \*/

// typedefs for the states

// State definitions for use with the query function

typedef enum { DEBOUNCING, Ready2Sample} ButtonState\_t ;

// Public Function Prototypes

bool InitButtonFSM ( uint8\_t Priority );

bool PostButtonFSM( ES\_Event ThisEvent );

ES\_Event RunButtonFSM( ES\_Event ThisEvent );

bool CheckButtonEvents(void);

#endif /\* FSMTemplate\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

MorseElementFSM.c

Revision

1.0.1

Description

This is a template file for implementing flat state machines under the

Gen2 Events and Services Framework.

Notes

History

When Who What/Why

-------------- --- --------

01/15/12 11:12 jec revisions for Gen2 framework

11/07/11 11:26 jec made the queue static

10/30/11 17:59 jec fixed references to CurrentEvent in RunTemplateSM()

10/23/11 18:20 jec began conversion from SMTemplate.c (02/20/07 rev)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*----------------------------- 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 "MorseElementFSM.h"

#include <stdio.h>

#include "termio.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

#include "inc/hw\_sysctl.h"

#include "driverlib/gpio.h"

#include "ES\_Timers.h"

#include "ES\_Events.h"

#include "driverlib/interrupt.h"

/\*----------------------------- Module Defines ----------------------------\*/

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

/\* prototypes for private functions for this machine.They should be functions

relevant to the behavior of this state machine

\*/

static void TestCalibration(void);

static void CharacterizeSpace(void);

static void CharacterizePulse(void);

/\*---------------------------- Module Variables ---------------------------\*/

// everybody needs a state variable, you may need others as well.

// type of state variable should match htat of enum in header file

static MorseElementState\_t CurrentState;

// with the introduction of Gen2, we need a module level Priority var as well

static uint8\_t MyPriority;

static uint16\_t TimeOfLastRise;

static uint16\_t TimeOfLastFall;

static uint16\_t LengthOfDot;

static uint16\_t FirstDelta;

#define ALL\_BITS (0xff << 2)

/\*------------------------------ Module Code ------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

InitTemplateFSM

Parameters

uint8\_t : the priorty of this service

Returns

bool, false if error in initialization, true otherwise

Description

Saves away the priority, sets up the initial transition and does any

other required initialization for this state machine

Notes

Author

J. Edward Carryer, 10/23/11, 18:55

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Register\_MORSE\_BUTTONPins(void){

printf("Init\_Morse Pin");

HWREG(SYSCTL\_RCGCGPIO) |= SYSCTL\_RCGCGPIO\_R5;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_DEN) |= BIT2HI;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_DIR) &= BIT2LO;

}

bool InitMorseElementFSM ( uint8\_t Priority )

{

printf("Init");

ES\_Event ThisEvent;

MyPriority = Priority;

Register\_MORSE\_BUTTONPins();

// put us into the Initial PseudoState

CurrentState = InitMorseElements;

FirstDelta = 0;

// post the initial transition event

ThisEvent.EventType = ES\_INIT;

if (ES\_PostToService( MyPriority, ThisEvent) == true) return true;

else return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

PostTemplateFSM

Parameters

EF\_Event ThisEvent , the event to post to the queue

Returns

boolean False if the Enqueue 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 PostMorseElementFSM( ES\_Event ThisEvent )

{

return ES\_PostToService( MyPriority, ThisEvent);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

RunTemplateFSM

Parameters

ES\_Event : the event to process

Returns

ES\_Event, ES\_NO\_EVENT if no error ES\_ERROR otherwise

Description

add your description here

Notes

uses nested switch/case to implement the machine.

Author

J. Edward Carryer, 01/15/12, 15:23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ES\_Event RunMorseElementFSM( ES\_Event ThisEvent )

{

MorseElementState\_t NextState;

ES\_Event ReturnEvent;

ReturnEvent.EventType = ES\_NO\_EVENT; // assume no errors

NextState = CurrentState;

switch(CurrentState){

case InitMorseElements :

//printf("Init\_MorseElements");

if(ThisEvent.EventType == ES\_INIT) NextState = CalWaitForRise;

break;

case CalWaitForRise :

//printf("CalWaitForRise");

switch(ThisEvent.EventType){

case RisingEdge :

//printf("RisingEdge");

TimeOfLastRise = ThisEvent.EventParam;

printf("TimeOfLastRise %d ",TimeOfLastRise);

NextState = CalWaitForFall;

break;

case CalibrationCompleted :

printf("CalibrationCompleted");

NextState = EOC\_WaitRise;

break;

}

break;

case CalWaitForFall :

if(ThisEvent.EventType == FallingEdge){

TimeOfLastFall = ThisEvent.EventParam;

printf("TimeOfLastFall %d ",TimeOfLastFall);

NextState = CalWaitForRise;

TestCalibration();

}

break;

case EOC\_WaitRise :

switch(ThisEvent.EventType){

case RisingEdge :

TimeOfLastRise = ThisEvent.EventParam;

NextState = EOC\_WaitFall;

CharacterizeSpace();

break;

case DBButtonDown :

NextState = CalWaitForRise;

FirstDelta = 0;

break;

}

break;

case EOC\_WaitFall :

switch(ThisEvent.EventType){

case FallingEdge :

TimeOfLastFall = ThisEvent.EventParam;

NextState = EOC\_WaitRise;

break;

case DBButtonDown :

NextState = CalWaitForRise;

FirstDelta = 0;

break;

case EOCDetected :

NextState = DecodeWaitFall;

break;

}

break;

case DecodeWaitRise :

switch(ThisEvent.EventType){

case RisingEdge :

TimeOfLastRise = ThisEvent.EventParam;

NextState = DecodeWaitFall;

CharacterizeSpace();

break;

case DBButtonDown :

NextState = CalWaitForRise;

FirstDelta = 0;

break;

}

break;

case DecodeWaitFall :

switch(ThisEvent.EventType){

case FallingEdge :

TimeOfLastFall = ThisEvent.EventParam;

NextState = DecodeWaitRise;

CharacterizePulse();

break;

case DBButtonDown :

NextState = CalWaitForRise;

FirstDelta = 0;

break;

}

break;

}

CurrentState = NextState;

return ReturnEvent;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void TestCalibration(void){

uint16\_t SecondDelta;

ES\_Event ThisEvent;

if(FirstDelta == 0) {FirstDelta = TimeOfLastFall - TimeOfLastRise; printf(" FirstDelta : %d", FirstDelta);}

else {

SecondDelta = TimeOfLastFall - TimeOfLastRise;

printf(" SecondDelta : %d", SecondDelta);

if((100\*FirstDelta / SecondDelta) <= 33){

LengthOfDot = FirstDelta;

printf(" LengthOfDot : %d", FirstDelta);

ThisEvent.EventType = CalibrationCompleted;

ThisEvent.EventParam = 1;

PostMorseElementFSM(ThisEvent);

}

else if((100\*FirstDelta / SecondDelta) >= 300){

LengthOfDot = SecondDelta;

ThisEvent.EventType = CalibrationCompleted;

ThisEvent.EventParam = 1;

PostMorseElementFSM(ThisEvent);

printf(" LengthOfDot : %d", SecondDelta);

}

else FirstDelta = SecondDelta;

}

}

static void CharacterizeSpace(void){

uint16\_t LastInterval;

LastInterval = TimeOfLastRise - TimeOfLastFall;

printf(" LastInterval : %d", LastInterval);

//interval is ok for character space

if ((LastInterval <= (LengthOfDot - 10)) || (LastInterval >= (LengthOfDot + 10))){

if ((LastInterval >= (3\*LengthOfDot - 30)) && (LastInterval <= (3\*LengthOfDot + 30))){

ES\_Event ThisEvent;

ThisEvent.EventType = EOCDetected;

ThisEvent.EventParam = 1;

if(CurrentState != EOC\_WaitFall) ES\_PostList02(ThisEvent); //decode

PostMorseElementFSM(ThisEvent);

printf("EOCDetected");

}

else{

//interval is ok for word space

if ((LastInterval >= (7\*LengthOfDot-30)) && (LastInterval <= (7\*LengthOfDot+30))){

ES\_Event ThisEvent;

ThisEvent.EventType = EOWDetected;

ThisEvent.EventParam = 1;

ES\_PostList01(ThisEvent);

printf("EOWDetected");

}

else{

ES\_Event ThisEvent;

ThisEvent.EventType = BadSpace;

ThisEvent.EventParam = 1;

ES\_PostList02(ThisEvent);

printf("BadSpace");

}//end else

}//end else

}

}

static void CharacterizePulse(void){

uint16\_t LastPulseWidth;

LastPulseWidth = TimeOfLastFall - TimeOfLastRise;

printf(" LastPulseWidth : %d", LastPulseWidth);

//if interval ok for dot

if ((LastPulseWidth >= (LengthOfDot - 10)) && (LastPulseWidth <= (LengthOfDot + 10))){

ES\_Event ThisEvent;

ThisEvent.EventType = DotDetected;

ThisEvent.EventParam = 1;

ES\_PostList02(ThisEvent);

printf("DotDetected");

} //end if

else{

//if interval ok for a dash

if ((LastPulseWidth >= (3\*LengthOfDot - 30)) && (LastPulseWidth <= (3\*LengthOfDot + 30))){

ES\_Event ThisEvent;

ThisEvent.EventType = DashDetected;

ThisEvent.EventParam = 1;

ES\_PostList02(ThisEvent);

printf("DashDetected");

}//end if

else{

ES\_Event ThisEvent;

ThisEvent.EventType = BadPulse;

ThisEvent.EventParam = 1;

ES\_PostList02(ThisEvent);

printf("BadPulse");

}//end else

}//end else

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Header file for MorseElement Sate Machine

based on the Gen2 Events and Services Framework

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef FSMMorseElement\_H

#define FSMMorseElement\_H

// Event Definitions

#include "ES\_Configure.h" /\* gets us event definitions \*/

#include "ES\_Types.h" /\* gets bool type for returns \*/

// typedefs for the states

// State definitions for use with the query function

typedef enum { InitMorseElements, CalWaitForRise, CalWaitForFall, EOC\_WaitRise, EOC\_WaitFall, DecodeWaitRise, DecodeWaitFall } MorseElementState\_t ;

// Public Function Prototypes

bool InitMorseElementFSM ( uint8\_t Priority );

bool PostMorseElementFSM( ES\_Event ThisEvent );

ES\_Event RunMorseElementFSM( ES\_Event ThisEvent );

#endif /\* FSMTemplate\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Module

MorseDecodeFSM.c

Revision

1.0.1

Description

This is a template file for implementing flat state machines under the

Gen2 Events and Services Framework.

Notes

History

When Who What/Why

-------------- --- --------

01/15/12 11:12 jec revisions for Gen2 framework

11/07/11 11:26 jec made the queue static

10/30/11 17:59 jec fixed references to CurrentEvent in RunTemplateSM()

10/23/11 18:20 jec began conversion from SMTemplate.c (02/20/07 rev)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*----------------------------- 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 <string.h>

#include <stdbool.h>

#include "ES\_Configure.h"

#include "ES\_Framework.h"

#include "MorseDecodeFSM.h"

#include "ES\_Timers.h"

#include "ES\_Events.h"

#include "ES\_General.h"

#include "LCD.h"

#include "termio.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

#include "inc/hw\_sysctl.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "ES\_Timers.h"

/\*----------------------------- Module Defines ----------------------------\*/

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

/\* prototypes for private functions for this machine.They should be functions

relevant to the behavior of this state machine

\*/

static void ClearMorseChar(void);

static char DecodeMorseString(void) ;

static bool MorseStringHasSpace(void);

/\*---------------------------- Module Variables ---------------------------\*/

// everybody needs a state variable, you may need others as well.

// type of state variable should match htat of enum in header file

//static MorseDecodeState\_t CurrentState;

// with the introduction of Gen2, we need a module level Priority var as well

static uint8\_t MyPriority;

#define StringLength 8

static char MorseString[StringLength];

char LegalChars[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890?.,:'-/()\"= !$&+;@\_";

char MorseCode[][8] ={ ".-","-...","-.-.","-..",".","..-.","--.",

"....","..",".---","-.-",".-..","--","-.","---",

".--.","--.-",".-.","...","-","..-","...-",

".--","-..-","-.--","--..",".----","..---",

"...--","....-",".....","-....","--...","---..",

"----.","-----","..--..",".-.-.-","--..--",

"---...",".----.","-....-","-..-.","-.--.-",

"-.--.-",".-..-.","-...-","-.-.--","...-..-",

".-...",".-.-.","-.-.-.",".--.-.","..--.-"

};

/\*------------------------------ Module Code ------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

InitTemplateFSM

Parameters

uint8\_t : the priorty of this service

Returns

bool, false if error in initialization, true otherwise

Description

Saves away the priority, sets up the initial transition and does any

other required initialization for this state machine

Notes

Author

J. Edward Carryer, 10/23/11, 18:55

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool InitMorseDecodeFSM ( uint8\_t Priority )

{

ES\_Event ThisEvent;

MyPriority = Priority;

// put us into the Initial PseudoState

ClearMorseChar();

LCDInit();

//CurrentState = InitPState;

ThisEvent.EventType = ES\_INIT;

if (ES\_PostToService( MyPriority, ThisEvent) == true) return true;

else return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

PostTemplateFSM

Parameters

EF\_Event ThisEvent , the event to post to the queue

Returns

boolean False if the Enqueue 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 PostMorseDecodeFSM( ES\_Event ThisEvent )

{

return ES\_PostToService( MyPriority, ThisEvent);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function

RunTemplateFSM

Parameters

ES\_Event : the event to process

Returns

ES\_Event, ES\_NO\_EVENT if no error ES\_ERROR otherwise

Description

add your description here

Notes

uses nested switch/case to implement the machine.

Author

J. Edward Carryer, 01/15/12, 15:23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ES\_Event RunMorseDecodeFSM( ES\_Event ThisEvent )

{

ES\_Event ReturnEvent;

ReturnEvent.EventType = ES\_NO\_EVENT; // assume no errors

char OutputChar;

switch(ThisEvent.EventType){

case DotDetected :

if(MorseStringHasSpace()){

printf("add dot");

strcat(MorseString, ".");

}

else{

ReturnEvent.EventType = ES\_ERROR;

}

break;

case DashDetected:

if(MorseStringHasSpace()){

printf("add dash");

strcat(MorseString, "-");

}

else{

ReturnEvent.EventType = ES\_ERROR;

}

break;

case EOCDetected:

if (DecodeMorseString() != '\*'){

//print to LCD

OutputChar=DecodeMorseString();

LCDputchar(OutputChar);

ClearMorseChar();

}

else{

ReturnEvent.EventType = ES\_ERROR;

}

break;

case EOWDetected:

if (DecodeMorseString()!= '\*'){

//print to LCD

OutputChar = DecodeMorseString();

LCDputchar(OutputChar);

LCDputchar(' ');

//print space to lcd

ClearMorseChar();

}

else{

ReturnEvent.EventType = ES\_ERROR;

}

break;

case DBButtonDown:

ClearMorseChar();

break;

}

return ReturnEvent;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void ClearMorseChar(void){

for(int i=0; i < StringLength; i++) MorseString[i] = '\0';

}

static char DecodeMorseString(void){

for (int i = 0; i < 55; i++){

if (strcmp(MorseString,MorseCode[i]) == 0){

puts("decoded =") ;

printf("%c",LegalChars[i]);

puts("=");

return LegalChars[i];

}

}

puts("decode error");

return '\*';

}

static bool MorseStringHasSpace(void){

return StringLength-1-strlen(MorseString);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Header file for MorseDecode Sate Machine

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef FSMMorseDecode\_H

#define FSMMorseDecode\_H

// Event Definitions

#include "ES\_Configure.h" /\* gets us event definitions \*/

#include "ES\_Types.h" /\* gets bool type for returns \*/

// Public Function Prototypes

bool InitMorseDecodeFSM ( uint8\_t Priority );

bool PostMorseDecodeFSM( ES\_Event ThisEvent );

ES\_Event RunMorseDecodeFSM( ES\_Event ThisEvent );

#endif /\* FSMTemplate\_H \*/

Appendix

#include <stdint.h>

#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include "ES\_Timers.h"

#include "ES\_Port.h"

#include "termio.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

#include "inc/hw\_sysctl.h"

#include "driverlib/gpio.h"

#define clrScrn() printf("\x1b[2J")

#define ALL\_BITS (0xff<<2)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Delay for few 100uS.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void delay\_uS(unsigned int num\_100\_uS){

int i=0;

while(i <= 1000\*num\_100\_uS) i++;

}

static void Register\_Data\_Pins(void){

HWREG(SYSCTL\_RCGCGPIO) |= SYSCTL\_RCGCGPIO\_R2;

HWREG(GPIO\_PORTC\_BASE + GPIO\_O\_DEN) |= 0xf0;

HWREG(GPIO\_PORTC\_BASE + GPIO\_O\_DIR) |= 0xf0;

HWREG(SYSCTL\_RCGCGPIO) |= SYSCTL\_RCGCGPIO\_R1;

HWREG(GPIO\_PORTB\_BASE + GPIO\_O\_DEN) |= 0xc0;

HWREG(GPIO\_PORTB\_BASE + GPIO\_O\_DIR) |= 0xc0;

}

static void Set\_RS\_bit(int state){

switch(state){

case 1: HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) |= BIT6HI; break;

case 0: HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) &= BIT6LO; break;

}

}

static void Set\_E\_bit(int state){

switch(state){

case 1: HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) |= BIT7HI; break;

case 0: HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) &= BIT7LO; break;

}

}

static void Wait(int Time){

uint16\_t T1 = ES\_Timer\_GetTime();

while(true){

uint16\_t T2 = ES\_Timer\_GetTime();

if((T2-T1)>Time) break;

}

}

static void Enable\_pulse(void){

HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) |= BIT7HI;

delay\_uS(10);

HWREG(GPIO\_PORTB\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) &= BIT7LO;

}

static void Set\_Bits(uint32\_t hex\_int){

HWREG(GPIO\_PORTC\_BASE+(GPIO\_O\_DATA+ALL\_BITS)) = hex\_int;

}

static void Move\_Cursor(void){

Set\_Bits(0x10);;

Enable\_pulse();

Set\_Bits(0xc0);;

Enable\_pulse();

Wait(5);

}

void Cursor\_To\_LeftMost(void){

for(int i=0; i<=23; i++) Move\_Cursor();

}

void LCDInit(void){

\_HW\_Timer\_Init(ES\_Timer\_RATE\_1mS);

TERMIO\_Init();

clrScrn();

Register\_Data\_Pins();

Set\_RS\_bit(0);

Set\_E\_bit(0);

Wait(150);

Set\_Bits(0x30);

Enable\_pulse();

Wait(5);

Set\_Bits(0x30);

Enable\_pulse();

Wait(1);

Set\_Bits(0x30);

Enable\_pulse();

Wait(1);

Set\_Bits(0x20);;

Enable\_pulse();

Wait(1);

Set\_Bits(0x20);;

Enable\_pulse();

Set\_Bits(0x00);;

Enable\_pulse();

Wait(1);

Set\_Bits(0x00);;

Enable\_pulse();

Set\_Bits(0x80);;

Enable\_pulse();

Wait(1);

Set\_Bits(0x00);;

Enable\_pulse();

Set\_Bits(0x10);;

Enable\_pulse();

Wait(5);

Set\_Bits(0x00);;

Enable\_pulse();

Set\_Bits(0x70);;

Enable\_pulse();

Wait(1);

Set\_Bits(0x00);;

Enable\_pulse();

Set\_Bits(0xf0);;

Enable\_pulse();

Wait(3);

Cursor\_To\_LeftMost();

}

void LCDputchar(char source){

Set\_RS\_bit(1);

char upper\_bits = source & 0xf0;

char lower\_bits = source & 0x0f;

lower\_bits <<= 4;

Set\_Bits((uint32\_t)upper\_bits);

Enable\_pulse();

Set\_Bits((uint32\_t)lower\_bits);

Enable\_pulse();

Wait(5);

Set\_RS\_bit(0);

}

#ifndef LCD\_H

#define LCD\_H

void LCDInit(void);

void LCDputchar(char source);

#endif

Morse Code Sample Pseudo Code Using the Gen2.x Event Framework

Rev 13 10/25/14

Pseudo-code for the Button module (a service that implements a state machine)

Data private to the module: LastButtonState

InitializeButton

Takes a priority number, returns True.

Initialize the MyPriority variable with the passed in parameter.

Initialize the port line to monitor the button

Sample the button port pin and use it to initialize LastButtonState

Set CurrentState to be DEBOUNCING

Start debounce timer (timer posts to ButtonDebounceSM)

End of InitializeButton (return True)

CheckButtonEvents

Takes no parameters, returns True if an event posted (11/04/11 jec)

Local ReturnVal = False, CurrentButtonState

Set CurrentButtonState to state read from port pin

If the CurrentButtonState is different from the LastButtonState

Set ReturnVal = True

If the CurrentButtonState is down

PostEvent ButtonDown to ButtonDebounce queue

Else

PostEvent ButtonUp to ButtonDebounce queue

Endif

Endif

Set LastButtonState to the CurrentButtonState

Return ReturnVal

End of CheckButtonEvents

RunButtonDebounceSM (implements a 2-state state machine for debouncing timing)

The EventType field of ThisEvent will be one of: ButtonUp, ButtonDown, or ES\_TIMEOUT

If CurrentState is Debouncing

If EventType is ES\_TIMEOUT & parameter is debounce timer number

Set CurrentState to READY2SAMPLE

Else if CurrentState is Ready2Sample

If EventType is ButtonUp

Start debounce timer

Set CurrentState to DEBOUNCING

Post DBButtonUp to MorseElements & DecodeMorse queues

End if

If EventType is ButtonDown

Start debounce timer

Set CurrentState to DEBOUNCING

Post DBButtonDown to MorseElements & DecodeMorse queues

End if

End Else

Return ES\_NO\_EVENT

End of RunButtonDebounceSM

Pseudo-code for the Morse Elements module (a service that implements a state machine)

Data private to the module: MyPriority, CurrentState, TimeOfLastRise, TimeOfLastFall, LengthOfDot, FirstDelta

InitializeMorseElements

Takes a priority number, returns True.

Initialize the MyPriority variable with the passed in parameter.

Initialize the port line to receive Morse code

Set CurrentState to be InitMorseElements

Set FirstDelta to 0

Post Event ES\_Init to MorseElements queue (this service)

End of InitializeMorseElements

CheckMorseEvents (This function could also be placed in an "EventCheckers" module)

Takes no parameters, returns True if an event was posted (11/4/11 jec)

Static local: LastInputState; local RetrunVal = False, CurrentInputState

Get the CurrentInputState from the input line

If the state of the Morse input line has changed

If the current state of the input line is high

PostEvent RisingEdge with parameter of the Current Time

Else (current input state is low)

PostEvent FallingEdge with parameter of the Current Time

Endif

Set ReturnVal = True

Endif

Set LastInputState to CurrentInputState

Return ReturnVal

End of CheckMorseEvents

RunMorseElementsSM (implements the state machine for Morse Elements)

The EventType field of ThisEvent will be one of: ES\_Init, RisingEdge, FallingEdge, CalibrationCompleted, EOCDetected, DBButtonDown. The parameter field of the ThisEvent will be the time that the event occurred.

Returns ES\_NO\_EVENT

Local Variables: NextState

Set NextState to CurrentState

Based on the state of the CurrentState variable choose one of the following blocks of code:

CurrentState is InitMorseElements

If ThisEvent is ES\_Init

Set NextState to CalWaitForRise

Endif

End InitMorseElements block

CurrentState is CalWaitForRise

If ThisEvent is RisingEdge

Set TimeOfLastRise to Time from event parameter

Set NextState to CalWaitForFall

Endif

If ThisEvent is CalibrationComplete

Set NextState to EOC\_WaitRise

Endif

End CalWaitForRise block

CurrentState is CalWaitForFall

If ThisEvent is FallingEdge

Set TimeOfLastFall to Time from event parameter

Set NextState to CalWaitForRise

Call TestCalibration function

EndIf

End CalWaitForFall block

CurrentState is EOC\_WaitRise

If ThisEvent is RisingEdge

Set TimeOfLastRise to Time from event parameter

Set NextState to EOC\_WaitFall

Call CharacterizeSpace function

Endif

If ThisEvent is DBButtonDown

Set NextState to CalWaitForRise

Set FirstDelta to 0

Endif

End EOC\_WaitRise block

CurrentState is EOC\_WaitFall

If ThisEvent is FallingEdge

Set TimeOfLastFall to Time from event parameter

Set NextState to EOC\_WaitRise

EndIf

If ThisEvent is DBButtonDown

Set NextState to CalWaitForRise

Set FirstDelta to 0

Endif

If ThisEvent is EOCDetected

Set NextState to DecodeWaitFall

Endif

End EOC\_WaitFall block

CurrentState is DecodeWaitRise

If ThisEvent is RisingEdge

Set TimeOfLastRise to Time from event parameter

Set NextState to DecodeWaitFall

Call CharacterizeSpace function

Endif

If ThisEvent is DBButtonDown

Set NextState to CalWaitForRise

Set FirstDelta to 0

Endif

End DecodeWaitRise block

CurrentState is DecodeWaitFall

If ThisEvent is FallingEdge

Set TimeOfLastFall to Time from event parameter

Set NextState to DecodeWaitRise

Call CharacterizePulse function

Endif

If ThisEvent is DBButtonDown

Set NextState to CalWaitForRise

Set FirstDelta to 0

Endif

End DecodeWaitFall block

Set CurrentState to NextState

Return ES\_NO\_EVENT

End of RunMorseElementsSM

TestCalibration

Takes no parameters, returns nothing.

Local variable SecondDelta

If calibration is just starting (FirstDelta is 0)

Set FirstDelta to most recent pulse width

Else

Set SecondDelta to most recent pulse width

If (100 \* FirstDelta / SecondDelta) less than or equal to 33

Save FirstDelta as LengthOfDot

PostEvent CalCompleted to MorseElementsSM

ElseIf (100 \* FirstDelta / Second Delta) greater than or equal to 300

Save SecondDelta as LengthOfDot

PostEvent CalCompleted to MorseElementsSM

Else (prepare for next pulse)

SetFirstDelta to SecondDelta

EndIf

EndIf

Return

End of TestCalibration

CharacterizeSpace

Takes no parameters, returns nothing.

Posts one of EOCDetected Event, EOWDetected Event, BadSpace Event as appropriate

on good dot-space, does nothing

Local variable LastInterval

Calculate LastInterval as TimeOfLastRise – TimeOfLastFall

If LastInterval not OK for a Dot Space

If LastInterval OK for a Character Space

PostEvent EOCDetected Event to Decode Morse Service & Morse Elements Service

Else

If LastInterval OK for Word Space

PostEvent EOWDetected Event to Decode Morse Service

Else

PostEvent BadSpace Event to Decode Morse Service

EndIf

Endif

EndIf

Return

End of CharacterizeSpace

CharacterizePulse

Takes no parameters, returns nothing.

Posts one of DotDetectedEvent, DashDetectedEvent, BadPulseEvent,

Local variable LastPulseWidth

Calculate LastPulseWidth as TimeOfLastFall - TimeOfLastRise

If LastPulseWidth OK for a dot

PostEvent DotDetected Event to Decode Morse Service

Else

If LastPulseWidth OK for dash

PostEvent DashDetected Event to Decode Morse Service

Else

PostEvent BadPulse Event to Decode Morse Service

EndIf

EndIf

Return

End of CharacterizePulse

Pseudo-code for the Decode Morse module (a simple service)

Data private to the module: MorseString, the arrays LegalChars and MorseCode

InitMorseDecode

Takes a priority number, returns True.

Initialize the MyPriority variable with the passed in parameter.

Call ClearMorseChar

Call LCDInit()

End of InitMorseDecode

RunMorseDecode (implements the service for Morse Decode)

The EventType field of ThisEvent will be one of: DotDetectedEvent, DashDetectedEvent, EOCDetected, EOWDetected, ButtonDown.

Returns ES\_NO\_EVENT if No Error detected, ES\_ERROR otherwise

local var ReturnValue initialized to ES\_NO\_EVENT

Based on the state of the ThisEvent variable choose one of the following blocks of code:

If ThisEvent is DotDetected Event

If there is room for another Morse element in the internal representation

Add a Dot to the internal representation

Else

Set ReturnValue to ES\_ERROR with param set to indicate this location

End if ThisEvent is DotDetected Event

If ThisEvent is DashDetected Event

If there is room for another Morse element in the internal representation

Add a Dash to the internal representation

Else

Set ReturnValue to ES\_ERROR ERROR with param set to indicate this location

End if ThisEvent is DashDetected Event

If ThisEvent is EOCDetected Event

If call to DecodeMorse does not return an error

Print to LCD the decoded character

Call ClearMorseChar

Else

Set ReturnValue to ES\_ERROR ERROR with param set to indicate this location

End if is EOCDetected Event

If ThisEvent is EOWDetected Event

If call to DecodeMorse does not return an error

Print to LCD the decoded character

Print to the LCD a space

Call ClearMorseChar

Else

Set ReturnValue to ES\_ERROR ERROR with param set to indicate this location

End if ThisEvent is EOWDetected Event

If ThisEvent is DBButtonDown Event (Added 11/3/11 JEC)

Call ClearMorseChar

End if ThisEvent is ButtonDown Event

return ReturnValue

End MorseDecode

ClearMorseChar

Takes no parameters, returns nothing

Clear (empty) the MorseString variable

End of ClearMorseChar

DecodeMorseString

Takes no parameters, returns either a character or a symbolic value indicating failure

For every entry in the array MorseCode

If MorseString is the same as current position in MorseCode

return contents of current position in LegalChars

EndIf

EndFor

return ERROR, since we didn't find a matching string in MorseCode

End of DecodeMorseString

Pseudo-code for the Main module

Main Function

Takes no parameters, returns nothing

Initialize the event framework (ES\_Initialize)

This call to ES\_Initialize will call the initialization functions for the Button, MorseElements and DecodeMorse modules

Call ES\_Run to check for and process events

This call to ES\_Run will only return in case of an error that can not be handled.

Process whatever error occurred if we get to this point.

End of Main Function