```
/*****************************************************************************
 Module
     main.c
 Description
     starter main() function for Events and Services Framework applications
 Notes

 History
 When            Who     What/Why
 --------------- ---     --------
 08/21/17 12:53 jec     added this header as part of coding standard and added
                        code to enable as GPIO the port poins that come out of
                        reset locked or in an alternate function.
*****************************************************************************/
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/gpio.h"

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "ES_Port.h"
#include "termio.h"
#include "EnablePA25 PB23 PD7 PF0.h"
#include "PWM16Tiva.h"
#include "ADMulti.h"

#define clrScrn() printf("\x1b[2J")
#define goHome() printf("\x1b[1,1H")
#define clrLine() printf("\x1b[K")

int main(void)
{
  ES_Return_t ErrorType;

  // Set the clock to run at 40MhZ using the PLL and 16MHz external crystal
  SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN
      | SYSCTL_XTAL_16MHZ);
  TERMIO_Init();
  clrScrn();

  // When doing testing, it is useful to announce just which program
  // is running.
  puts("\rTest Harness for \r");
  printf( "the 2nd Generation Events & Services Framework V2.4\r\n");
  printf( "%s %s\n", __TIME__, __DATE__);
  printf( "\n\r\n");
  printf( "Press any key to post key-stroke events to Service 0\n\r");
  printf( "Press 'd' to test event deferral \n\r");
  printf( "Press 'r' to test event recall \n\r");

  // reprogram the ports that are set as alternate functions or
  // locked coming out of reset. (PA2-5, PB2-3, PD7, PF0)
  // After this call these ports are set
  // as GPIO inputs and can be freely re-programmed to change config.
  // or assign to alternate any functions available on those pins
  PortFunctionInit();
```

```
  // Your hardware initialization function calls go here
  // Initialize Port A
      HWREG(SYSCTL_RCGCGPIO) |= BIT0HI;

      // Wait for clock to get ready
      while ((HWREG(SYSCTL_PRGPIO) & BIT0HI) != BIT0HI)
      {
      }

      // Set data direction and assign digital port on Port A
      // PA2 = Harvest Button (INPUT)
      // PA3 = Harvest Button LED (OUTPUT)
      // PA4 = Coin Sensor (INPUT)
      HWREG(GPIO_PORTA_BASE+GPIO_O_DEN) |= (BIT2HI | BIT3HI | BIT4HI);
      HWREG(GPIO_PORTA_BASE+GPIO_O_DIR) = BIT3HI;

  // Initialize Port B
      HWREG(SYSCTL_RCGCGPIO) |= BIT1HI;

      // Wait for clock to get ready
      while ((HWREG(SYSCTL_PRGPIO) & BIT1HI) != BIT1HI)
      {
      }

      // Set data direction and assign digital port on Port A
      // PB0 = SR Data (OUT)
      // PB1 = SR Shift clock (OUT)
      // PB2 = SR Register clock (OUT)
  // PB3 = Vibration Motor (OUT)
  // PB5 = PWM Servo 1 (OUT)
  // PB6 = PWM Servo 2 (OUT)
  // PB7 = PWM Servo 3 (OUT)
      HWREG(GPIO_PORTB_BASE+GPIO_O_DEN) |= (BIT0HI | BIT1HI | BIT2HI | BIT3HI |
BIT5HI | BIT6HI | BIT7HI);
      HWREG(GPIO_PORTB_BASE+GPIO_O_DIR) = (BIT0HI | BIT1HI | BIT2HI | BIT3HI);

  // PWM and ADC Initialize
  static uint8_t PWM_NumChannels = 6;
  static uint8_t ADC_NumPins = 4;

  PWM_TIVA_Init(PWM_NumChannels);
  ADC_MultiInit(ADC_NumPins);

  // now initialize the Events and Services Framework and start it running
  ErrorType = ES_Initialize(ES_Timer_RATE_1mS);
  if (ErrorType == Success)
  {
    ErrorType = ES_Run();
    printf("did es_run \r\n");
  }
  //if we got to here, there was an error
  switch (ErrorType)
  {
    case FailedPost:
    {
      printf("Failed on attempt to Post\n");
    }
    break;
    case FailedPointer:
    {
      printf("Failed on NULL pointer\n");
    }
    break;
```

```
    case FailedInit:
    {
      printf("Failed Initialization\n");
    }
    break;
    default:
    {
      printf("Other Failure\n");
    }
    break;
  }
  for ( ; ;)
  {
    ;
  }
}
```