```
/*****************************************************************************
 Module
   ADCMulti.c

 Revision
   1.0.1

 Description
   This file implements a set of functions to initialize and read up to 4
   A/D channels on the Tiva

 Notes
  I started with ADCSWTrigger.c from Valvano's book for the basic operation
  sequence. That exmaple did only 2 fixed channels, so I re-wrote it to
  take a parameter to specify how many channels and generalize the init
  constants into a set of data structures rather than hard coded constants.
  All of the magic numbers are left over from the original example. Mea Culpa!

 History
 When           Who     What/Why
 -------------- ---     --------
 08/22/17 17:39 jec     started the cleanup and prep to make this the
                        standard A/D libary for ME218
*****************************************************************************/

/*----------------------------- Include Files -----------------------------*/
#include <stdint.h>
#include "inc/hw_gpio.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_sysctl.h"
#include "inc/tm4c123gh6pm.h"

#include "ADMulti.h"

/*----------------------------- Module Defines ----------------------------*/
// None

/*---------------------------- Module Functions ---------------------------*/
// None

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

static const uint32_t HowMany2Mask[4] = {0x01,0x03,0x07,0x0F};
// this mapping puts PE0 as resuult 0, PE1 as result 1...
static const uint32_t HowMany2Mux[4] = {0x03,0x023,0x123,0x0123};
static const uint32_t HowMany2CTL[4] = {ADC_SSCTL2_END0|ADC_SSCTL2_IE0,
                                        ADC_SSCTL2_END1|ADC_SSCTL2_IE1,
                                        ADC_SSCTL2_END2|ADC_SSCTL2_IE2,
                                        ADC_SSCTL2_END3|ADC_SSCTL2_IE3};
static uint8_t NumChannelsConverting;

/*------------------------------ Module Code ------------------------------*/
/*****************************************************************************
 Function
    ADC_MultiInit

 Parameters
    uint8_t : How many A/D channels to set up: 1-4

 Returns
    void

 Description
```

```
    enables the A/D #0 and Port E and configures the converter and the
    port to allow A/D conversions on the requested number of channels

 Notes
    Based on example code from Jonathan Valvano

 Author
    J. Edward Carryer, 08/22/17, 17:49
******************************************************************************/
void ADC_MultiInit(uint8_t HowMany){
  uint8_t index = HowMany-1; // index into the HowMany2Mask array

  // first sanity check on the HowMany parameter
  if ( (0 == HowMany) || (4 < HowMany))
    return;

  NumChannelsConverting = HowMany;

  SYSCTL_RCGCADC_R |= 0x00000001;                 // 1) activate clock for ADC0
  while((SYSCTL_PRADC_R & 0x0001) != 0x0001)
    ;                                             // 2) allow time for ADC clock
to stabilize
  SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R4;        // 3) activate clock for Port E
  while ((HWREG(SYSCTL_PRGPIO) & SYSCTL_PRGPIO_R4) != SYSCTL_PRGPIO_R4)
    ;                                             // 4) allow time for port E
clock to stabilize
  GPIO_PORTE_DIR_R &= ~HowMany2Mask[index];       // 5) make PE0, PE1, PE2, PE3
inputs
  GPIO_PORTE_AFSEL_R |= HowMany2Mask[index];      // 6) enable alternate function
on PE0 - PE3
  GPIO PORTE DEN R &= ~HowMany2Mask[index];       // 7) disable digital I/O on
PE0 - PE3
  GPIO_PORTE_AMSEL_R |= HowMany2Mask[index];      // 8) enable analog
functionality on PE0 - PE3

  ADC0_PC_R &= ~0xF;                              // 9) clear max sample rate
field
  ADC0_PC_R |= 0x3;                               // 10) configure for 250K
samples/sec
  ADC0_SSPRI_R = 0x3210;                          // 11) Set sequencer 3 as
lowest priority
  ADC0_ACTSS_R &= ~0x0004;                        // 12) disable sample sequencer
2
  ADC0_EMUX_R &= ~0x0F00;                         // 13) Set seq2 as software
trigger
  ADC0_SSMUX2_R = HowMany2Mux[index];             // 14) set channels for SS2
  ADC0_SSCTL2_R = HowMany2CTL[index];             // 15) set which sample is last
  ADC0_IM_R &= ~0x0004;                           // 16) disable SS2 interrupts
  ADC0_ACTSS_R |= 0x0004;                         // 17) enable sample sequencer
2
}

/******************************************************************************
 Function
    ADC_MultiRead

 Parameters
    uint32_t data[4] pointer to the first element of an array to hold results

 Returns
    nothing

 Description
    Triggers A/D conversion and waits for result
```

```
    uses software trigger, busy-wait sampling, data returned by reference
    lowest numbered converted channel is in data[0]
    takes about 19.2uS to execute for 4 channels
    takes about 14.8uS to execute for 3 channels
    takes about 10.5uS to execute for 2 channels
    takes about 6.1uS to execute for 1 channel

 Notes
    Based on example code from Jonathan Valvano

 Author
     J. Edward Carryer, 08/22/17, 17:55
******************************************************************************///-
-----------ADC_MultiRead------------
void ADC_MultiRead(uint32_t data[4]){
  uint8_t i;

  ADC0_PSSI_R = 0x0004;               // 1) initiate conversion with SS2
  while((ADC0_RIS_R & 0x04) == 0)
    ;                                 // 2) wait for conversion(s) to complete
  for (i=0; i< NumChannelsConverting; i++){
    data[i] = ADC0_SSFIFO2_R & 0xFFF; // 3) read result(s), one at a time
  }
  ADC0_ISC_R = 0x0004;                // 4) acknowledge completion, clear int
}
```