```
/************************************************************/
/* FILENAME: MSP430F169 on ADC/DAC 12bits.c ltc1257 and 8560    */
/* DESCRIPTION: This program uses the MSP430F169 to write       */
/*    Dac control SPI                                  */
/*    AUTHOR: Wiotte Fabrice, LPL                       */
/*    CREATED November 2019                             */
/*    VERSION: 02                                       */
/* IAR Embedded Workbench KickStart for MSP430 Version: 6.40.1  */
/************************************************************/

#include <msp430x16x.h>
#include "MSP430F169 on UART0 and ADC3_DAC0 on.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>     /* atoi */
#include <math.h>

#define PHASEA BIT6
#define PHASEB BIT7

#define PHASEC BIT6
#define PHASED BIT7

/*************** Function prototypes ***********************************/
/*                                                  */
void init_sys(void);              // MSP430 Initialisation routine
void tempo_loop(long loop_number);         // wait
void standby();               // init state of state machine
void stop_programm();
void write_serial_port1(int data, int num_byte);
void write_immediate(void);
```

```
void copy_RAM_to_step();
void write_RAM_step();
void write_RAM_level_select();
void copy_RAM_to_level_select();
int read_ADC12(int chanel);       // read routine on ADC12
//static unsigned int DAC0_out;
void write_DAC12_0(int out_data); // write DAC12_0 routine

/*************** variable declarations ********************************/
int i = 0;
char j = 0;
int  t = 0;
int pas = 0;
unsigned char temp[2];
unsigned int a=0;
char serial[2];
int level_select = 0;
//static int ADC_in;
static int ADC_in2;
int inc;
int attente;

void main(void)
{
   // initialisation du msp430
   init_sys();      // Initialise the MSP430
   P1IE |= PHASEA;          // P1.6 interrupt enabled
   P1IES |= PHASEA;         // P1.6 Hi/lo edge
   P1IFG &= ~PHASEA;        // P1.6  IFG cleared

   P2IE |= PHASEC;          // P2.6 interrupt enabled
```

```
   P2IES |= PHASEC;         // P2.6 Hi/lo edge
   P2IFG &= ~PHASEC;        // P2.6  IFG cleared

   standby();

   _BIS_SR(LPM3_bits + GIE);           // Enter LPM3 w/ interrupt

}
/**********************function standby***********************************/
/*                                                  */
void standby()
{
  WDTCTL = WDTPW + WDTHOLD;          // clear and start WatchDog
  P5OUT |= BIT5; //=1   CS
  P5OUT &= ~ BIT5;  //=0 CS
  write_serial_port1(0x00,0x02);  // write to DAC6551-Q1 12 bits
}
// routine for encodeur 1 Reglage SLOW//
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{

 if (!(P1IN & PHASEB))
    {
     if(inc>0)
     {
     P5OUT |= BIT5; //=1   CS
     //tempo_loop(1);
     P5OUT &= ~ BIT5;  //=0 CS
     //inc -=1;
     inc--;
```

```
      write_serial_port1(inc<<4,2);  // write to DAC6551-Q1 12 bits
      tempo_loop(1);

     }P1IFG &= ~PHASEA;
     }
   else
   {
     if(inc>1 | inc <=4095)
      {
      P5OUT |= BIT5; //=1   CS
      //tempo_loop(1);
      P5OUT &= ~ BIT5;  //=0 CS
      //inc +=1;
      inc++;
      write_serial_port1(inc<<4,2);  // write to DAC6551-Q1 12 bits
      tempo_loop(1);
      P1IFG =0;
     }
     }
}
// routine for encodeur 2 Reglage FAST//
#pragma vector=PORT2_VECTOR
__interrupt void Port_2(void)
{

   if (!(P2IN & PHASED))
     {
      if(inc>=0 | inc <=4095)
      {
      P5OUT |= BIT5; //=1   CS
      //tempo_loop(1);
```

```
      P5OUT &= ~ BIT5;  //=0 CS
      //inc +=49;
      inc +=10;
      inc++;
      write_serial_port1(inc<<4,2);  // write to DAC6551-Q1 12 bits
      tempo_loop(1);

     }P2IFG &= ~PHASEC;
     }
    else
    {
     if(inc>10)
      {
      P5OUT |= BIT5; //=1   CS
      //tempo_loop(1);
      P5OUT &= ~ BIT5;  //=0 CS
      //inc -=49;
      inc -=10;
      inc--;
      write_serial_port1(inc<<4,2);  // write to DAC6551-Q1 12 bits
      tempo_loop(1);
      P2IFG =0;
     }
     }

}
/*************** function write_DAC12_0 ********************************/
/*                                                 */
void write_DAC12_0(int out_data)
{
   DAC12_0DAT = out_data;
```

```
}
/*************** function read_ADC12 *********************************/
/*                                                                    */
int read_ADC12(int chanel)
{
 int data;
 ADC12MCTL0 |= chanel;
 ADC12CTL0 |= ENC + ADC12SC;       // Start conversion
 tempo_loop(10);                  // wait loop
 data=ADC12MEM0;                   // read data
 ADC12CTL0 ^= ENC ;               // reset conversion
 ADC12MCTL0 ^= chanel;             // reset chanel
 return data;
}
/*************************************************************/
/* Prototype - init_sys                                      */
/*                                                           */
/*  Description                                              */
/*  This prototype initialises the MSP430F169                */
/*************************************************************/
void init_sys(void)
{
  WDTCTL = WDTPW + WDTHOLD;   // stop Watch Dog Timer
  init_port();   // init usart1, spi mode, 2wire
}
/*************************************************************/
/* Prototype - tempo_loop                                    */
/*                                                           */
/*  Description                                              */
/*  This prototype wait i loop                               */
/*************************************************************/
```

```
void tempo_loop(long loop_number)
{
  long i;
  long j;
  for(i = 0; i < loop_number; ++i)         // wait loop
  {
    j=j+i;
  }
}
void tempo(long loop_number)
{
  long i;
  long j;

  for(i = 0; i < 1000000; ++i)         // wait loop
  {
    j=j+i;
    attente = 1;
  }
}
/*************************************************************/
/* Prototype - write_serial_port1                            */
/*                                                           */
/*  Description                                              */
/*  This prototype write the AD9959 Frequency Tuning Word    */
/*  on USART1                                                */
/*************************************************************/
void write_serial_port1(int data, int num_byte)
{
  TXBUF1 = 0;
  int i=0;
```

```
 do
 {
   TXBUF1 = data >>(num_byte-1-i)*8;
   i++;
 }
 while (i < num_byte);
}
// UART0 TX ISR
#pragma vector=USART0TX_VECTOR
__interrupt void usart0_tx (void)
{

   if (i < sizeof t)
   TXBUF0 = temp[t++];

}
/*************************************************************/
/* Prototype - Incoming message interrupt                    */
/*                                                           */
/*  Description                                              */
/*  Treat incoming message from serial port                  */
/*************************************************************/
// UART0 RX ISR
#pragma vector=USART0RX_VECTOR
__interrupt void usart0_rx (void)
{
  temp[t++] = RXBUF0; //Read from buffer
  if(t ==2)
  {
    pas = (unsigned int) temp[1] << 8 | (unsigned int) temp[0]; //Convert to unsigned int
    //level_select = (unsigned int) temp[3] << 8 | (unsigned int) temp[2]; //Convert to unsigned int
```

```
    t = 0; //Reset read length
    ADC_in2 = 0;
    ADC_in2 = read_ADC12(INCH_5);

    if(ADC_in2<4095)
   {
      P5OUT |= BIT5; //=1   CS
      P5OUT &= ~ BIT5;  //=0 CS
      write_serial_port1(pas<<4,2);  // write to DAC DAC6551-Q1 12 bits
      //tempo_loop(1);
   }

   else
   {
      P5OUT |= BIT5; //=1   CS
      P5OUT &= ~ BIT5;  //=0 CS
      write_serial_port1(0,2);  // write to DAC DAC6551-Q1 12 bits
      tempo_loop(1);
   }

   }
}
```