### Arduino code for ADC

int CLK = 13; //Define ports that will be used

int CS = 12;

int DIN = 11;

int counter = 0;

int i =0;

void setup()

{

pinMode(CLK, OUTPUT); //Define Output pins

pinMode(CS, OUTPUT);

pinMode(DIN, OUTPUT);

}

void loop()

{

digitalWrite(CS,HIGH); //Determine the first status

digitalWrite(CLK,HIGH);

digitalWrite(DIN,LOW);

for (counter =0; counter<15;counter ++)

{

digitalWrite (CS,LOW); //Counter to count the number of clock cycles

DIN <= !(CS);

digitalWrite(DIN,HIGH);

digitalWrite(CLK, LOW);

delay(10);

digitalWrite(CLK, HIGH);

delay(10);

}

if (counter == 15)

{

digitalWrite(CS,HIGH);

digitalWrite(DIN,LOW); //make CS= !DIN

delay(100);

counter =0;

}

}

………………………………………………………………………………………..

### Altera C coding for DAC

#include <stdio.h>  
 #include "system.h"  
#include "altera\_avalon\_pio\_regs.h"  
#include "alt\_types.h"  
  
#define SYNC\_PIN\_HI 0x01  
#define SYNC\_PIN\_LO 0x00  
#define SCLK\_PIN\_HI 0x02  
#define SCLK\_PIN\_LO 0x00  
#define DATA\_PIN\_HI 0x04  
#define DATA\_PIN\_LO 0x00 //Define Variables  
#define LDAC\_PIN\_HI 0x08  
#define LDAC\_PIN\_LO 0x00  
  
#define DATA\_BITS 16  
#define TOP\_DATA\_BIT (DATA\_BITS - 1)  
  
void Delay(void)  
{  
 int i;  
  
 for(i=0;I<1000;i++) { //Introduce a delay  
  
}  
}  
//void delayclock(void)  
//{  
// int i;  
//   
// for(i=0;i<3;i++) {  
//   
// }  
//}  
//void delaysync(void)  
//{  
// int i;  
//   
// for(i=0;i<300;i++) {  
//   
// }  
//}  
//void delayldac(void)  
//{  
// int i;  
//   
// for(i=0;i<30;i++) {  
//   
// }  
//}  
alt\_u32 CreateGpioOutputMask(void)  
{  
 alt\_u8 i;  
 alt\_u32 mask = 0x00000000;  
  
// Create mask, one bit at a time   
for(i=0;i<PORT\_A\_SPAN;i++) {  
 mask |= (ALTERA\_AVALON\_PIO\_DIRECTION\_OUTPUT << i);  
}  
  
// Report result  
return mask;  
}  
  
void ClockSyncDataLdac(int syncState,int dataState,int ldacState)  
{   
  
 alt\_u32 clkHiBits = SCLK\_PIN\_HI; //delayclock(); // Delay to achieve 33 microseconds  
 alt\_u32 clkLoBits = SCLK\_PIN\_LO;  
  
// Mask in Sync state  
if(syncState) {  
 clkHiBits |= SYNC\_PIN\_HI;  
 clkLoBits |= SYNC\_PIN\_HI;   
}  
else {  
 clkHiBits |= SYNC\_PIN\_LO;  
 clkLoBits |= SYNC\_PIN\_LO;   
//delaysync();// Delay to achieve right time  
  
}  
  
// Mask in Data state  
if(dataState) {  
 clkHiBits |= DATA\_PIN\_HI;  
 clkLoBits |= DATA\_PIN\_HI;  
}  
else {  
 clkHiBits |= DATA\_PIN\_LO;  
 clkLoBits |= DATA\_PIN\_LO;  
}  
  
// Mask in LDAC state  
if(ldacState) {  
 clkHiBits |= LDAC\_PIN\_HI;  
 clkLoBits |= LDAC\_PIN\_HI;   
}  
else {  
 clkHiBits |= LDAC\_PIN\_LO;//delayldac ();//  
 clkLoBits |= LDAC\_PIN\_LO;  
  
}  
  
// Bit-bang the GPIO port  
Delay();  
IOWR\_ALTERA\_AVALON\_PIO\_DATA(PORT\_A\_BASE,clkHiBits);  
Delay();  
IOWR\_ALTERA\_AVALON\_PIO\_DATA(PORT\_A\_BASE,clkLoBits);  
}  
  
void SendDataToDac(alt\_u32 data)  
{  
int i;  
alt\_u32 mask;  
  
// Clock CS high  
ClockSyncDataLdac(1,0,1);  
  
// Clock CS low for  
for(i=TOP\_DATA\_BIT;i>=0;i--) {   
  
// Create mask  
 mask = 0x0001 << i;  
  
 if(data & mask) {  
 ClockSyncDataLdac(0,1,1);  
}  
 else {  
 ClockSyncDataLdac(0,0,1);  
}  
}  
  
// Clock SYNC high  
ClockSyncDataLdac(1,0,1);   
  
// Cycle LDAC low then high  
ClockSyncDataLdac(1,0,0);   
ClockSyncDataLdac(1,0,1);   
//delayldac();  
}  
  
  
  
alt\_u32 GetDataFromAdc(void)  
{  
alt\_u32 data;  
  
  
  
return data;   
}   
  
int main(void)  
{  
int count = 0;  
alt\_u32 dirMask = CreateGpioOutputMask();  
alt\_u32 dacControl1 = 0x8003;  
alt\_u32 dacControl2 = 0xA000;  
alt\_u32 dacData = 0x0000;  
  
// Setup all pins to output  
IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(PORT\_A\_BASE,dirMask);  
  
while(1) {  
// printf("Count=%d Dac=%d\n",count,dacData);  
  
count++;  
if(count==3000) {  
// printf("Reloading configuration\n");  
 SendDataToDac(dacControl1);  
 SendDataToDac(dacControl2);  
 count=0;  
}  
  
  
dacData += 10;  
if(dacData==250) {  
 dacData = 0;   
}  
SendDataToDac(dacData << 4);  
// dacData--; // to change the data  
  
// adcData = GetDataFromAdc();  
  
}  
  
// Report success  
return 0;  
}