#include <stdlib.h>

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <math.h>

#include <time.h>

#define Strobe (26) // IO8

#define GP\_4 (28) // IO4

#define GP\_5 (17) // IO5

#define GP\_6 (24) // IO6

#define GP\_7 (27) // IO7

#define GPIO\_DIRECTION\_IN (1)

#define GPIO\_DIRECTION\_OUT (0)

#define ERROR (-1)

#define HIGH 1

#define LOW 0

#define MSG\_RESET 0x0 /\* reset the sensor to initial state \*/

#define MSG\_PING 0x1 /\* check if the sensor is working \*/

#define MSG\_GET 0x2 /\* obtain the most recent ADC result \*/

#define MSG\_ACK 0xE /\* acknowledgement to the commands \*/

#define MSG\_NOTHING 0xF /\* reserved \*/

//open GPIO and set the direction

int openGPIO(int gpio, int direction )

{

char buffer[256];

int fileHandle;

int fileMode;

//Export GPIO

fileHandle = open("/sys/class/gpio/export", O\_WRONLY);

if(ERROR == fileHandle)

{

puts("Error: Unable to opening /sys/class/gpio/export");

return(-1);

}

sprintf(buffer, "%d", gpio);

write(fileHandle, buffer, strlen(buffer));

close(fileHandle);

//Direction GPIO

sprintf(buffer, "/sys/class/gpio/gpio%d/direction", gpio);

fileHandle = open(buffer, O\_WRONLY);

if(ERROR == fileHandle)

{

puts("Unable to open file:");

puts(buffer);

return(-1);

}

if (direction == GPIO\_DIRECTION\_OUT)

{

// Set out direction

write(fileHandle, "out", 3);

fileMode = O\_WRONLY;

}

else

{

// Set in direction

write(fileHandle, "in", 2);

fileMode = O\_RDONLY;

}

close(fileHandle);

//Open GPIO for Read / Write

sprintf(buffer, "/sys/class/gpio/gpio%d/value", gpio);

fileHandle = open(buffer, fileMode);

if(ERROR == fileHandle)

{

puts("Unable to open file:");

puts(buffer);

return(-1);

}

return(fileHandle); //This file handle will be used in read/write and close operations.

}

//write value

int writeGPIO(int fileHandle, int val)

{

if(val == 0)

{

// Set GPIO low status

write(fileHandle, "0", 1);

}

else

{

// Set GPIO high status

write(fileHandle, "1", 1);

}

return(0);

}

//read value

int readGPIO(int fileHandle)

{

int value;

read(fileHandle, &value, 1);

if('0' == value)

{

// Current GPIO status low

value = 0;

}

else

{

// Current GPIO status high

value = 1;

}

return value;

}

int closeGPIO(int gpio, int fileHandle)

{

char buffer[256];

close(fileHandle); //This is the file handle of opened GPIO for Read / Write earlier.

fileHandle = open("/sys/class/gpio/unexport", O\_WRONLY);

if(ERROR == fileHandle)

{

puts("Unable to open file:");

puts(buffer);

return(-1);

}

sprintf(buffer, "%d", gpio);

write(fileHandle, buffer, strlen(buffer));

close(fileHandle);

return(0);

}

//main

int main(void)

{

struct timespec delay;

delay.tv\_sec = 0;

delay.tv\_nsec = 10;

int adc=0;

int ack=0;

int fileHandleGPIO\_4;

int fileHandleGPIO\_5;

int fileHandleGPIO\_6;

int fileHandleGPIO\_7;

int fileHandleGPIO\_S;

fileHandleGPIO\_4 = openGPIO(GP\_4, GPIO\_DIRECTION\_OUT);

fileHandleGPIO\_5 = openGPIO(GP\_5, GPIO\_DIRECTION\_OUT);

fileHandleGPIO\_6 = openGPIO(GP\_6, GPIO\_DIRECTION\_OUT);

fileHandleGPIO\_7 = openGPIO(GP\_7, GPIO\_DIRECTION\_OUT);

fileHandleGPIO\_S = openGPIO(Strobe, GPIO\_DIRECTION\_OUT);

writeGPIO(fileHandleGPIO\_S,1);

nanosleep(&delay,NULL);

while(1)

{

//send command

//1.Strobe high

writeGPIO(fileHandleGPIO\_S,HIGH);

//strobe low

writeGPIO(fileHandleGPIO\_S,LOW);

//write COMMAND MSG\_GET

writeGPIO(fileHandleGPIO\_4,LOW);

writeGPIO(fileHandleGPIO\_5,HIGH);

writeGPIO(fileHandleGPIO\_6,LOW);

writeGPIO(fileHandleGPIO\_7,LOW);

//3.Strobe HIGH

writeGPIO(fileHandleGPIO\_S,HIGH);

nanosleep(&delay,NULL);

//4.Strobe low

writeGPIO(fileHandleGPIO\_S,LOW);

writeGPIO(fileHandleGPIO\_S,HIGH);

closeGPIO(GP\_4, fileHandleGPIO\_4);

closeGPIO(GP\_5, fileHandleGPIO\_5);

closeGPIO(GP\_6, fileHandleGPIO\_6);

closeGPIO(GP\_7, fileHandleGPIO\_7);

fileHandleGPIO\_4 = openGPIO(GP\_4, GPIO\_DIRECTION\_IN);

fileHandleGPIO\_5 = openGPIO(GP\_5, GPIO\_DIRECTION\_IN);

fileHandleGPIO\_6 = openGPIO(GP\_6, GPIO\_DIRECTION\_IN);

fileHandleGPIO\_7 = openGPIO(GP\_7, GPIO\_DIRECTION\_IN);

// get data

// adc 0-3

writeGPIO(fileHandleGPIO\_S,HIGH);

writeGPIO(fileHandleGPIO\_S,LOW);

nanosleep(&delay,NULL);

writeGPIO(fileHandleGPIO\_S,HIGH);

adc += pow (2,0)\*readGPIO(fileHandleGPIO\_4);

adc += pow (2,1)\*readGPIO(fileHandleGPIO\_5);

adc += pow (2,2)\*readGPIO(fileHandleGPIO\_6);

adc += pow (2,3)\*readGPIO(fileHandleGPIO\_7);

// adc 4-7

writeGPIO(fileHandleGPIO\_S,HIGH);

writeGPIO(fileHandleGPIO\_S,LOW);

nanosleep(&delay,NULL);

writeGPIO(fileHandleGPIO\_S,HIGH);

adc += pow (2,4)\*readGPIO(fileHandleGPIO\_4);

adc += pow (2,5)\*readGPIO(fileHandleGPIO\_5);

adc += pow (2,6)\*readGPIO(fileHandleGPIO\_6);

adc += pow (2,7)\*readGPIO(fileHandleGPIO\_7);

// adc 4-7

writeGPIO(fileHandleGPIO\_S,HIGH);

writeGPIO(fileHandleGPIO\_S,LOW);

nanosleep(&delay,NULL);

writeGPIO(fileHandleGPIO\_S,HIGH);

adc += pow (2,8)\*readGPIO(fileHandleGPIO\_4);

adc += pow (2,9)\*readGPIO(fileHandleGPIO\_5);

// msg ack

writeGPIO(fileHandleGPIO\_S,HIGH);

writeGPIO(fileHandleGPIO\_S,LOW);

nanosleep(&delay,NULL);

writeGPIO(fileHandleGPIO\_S,HIGH);

ack += pow (2,0)\*readGPIO(fileHandleGPIO\_4);

ack += pow (2,1)\*readGPIO(fileHandleGPIO\_5);

ack += pow (2,2)\*readGPIO(fileHandleGPIO\_6);

ack += pow (2,3)\*readGPIO(fileHandleGPIO\_7);

writeGPIO(fileHandleGPIO\_S,LOW);

writeGPIO(fileHandleGPIO\_S,HIGH);

if (ack==15)

{

printf("adc receieved\n");

printf ("adc is %d",adc);

}

else

{

printf("adc error");

}

}

}