#include <wiringPi.h>

#include <stdio.h>

#include <action.h>

int main(int agrc,char \*argv[]){

printf("XMPP the Low APP of XDM-I\n");

wiringPiSetup();

AppInit();

int i=0;

while(1){

i++;

printf("Main i=%d\n",i);

AppMain();

delay(200);

#ifdef DEBUG

if(i>10000)

break;

#endif

}

return 0;

}

#include <stdio.h>

#include <string.h>

#include <action.h>

#include <commu.h>

#include <types.h>

#include <cal.h>

#include <alg.h>

#include <record.h>

#define LEN\_BUF 32

U8 recBuf[LEN\_BUF];

U8 recBufOld[LEN\_BUF];

REC1 rec1;

int AppInit(){

memset(recBuf,0,LEN\_BUF\*sizeof(U8));

memset(recBufOld,0,LEN\_BUF\*sizeof(U8));

memset(&rec1,0,sizeof(REC1));

I2C\_Init();

CalInit();

return 0;

}

int AppMain(){

GetTPKG(recBuf);

// ShowTPKG(recBuf);

if(!TPKGIsNew(recBuf,recBufOld)){

ShowTPKG(recBuf);

CalRec1(&rec1,recBuf);

GetFlag(&rec1);

GetError(&rec1);

SaveRec11(rec1);

}

else {

}

return 0;

}

#include <stdio.h>

#include <string.h>

#include <alg.h>

#include <types.h>

#include <record.h>

#define TH\_M 100

#define TH\_X 10

int GetFlag(pREC1 rec){

// printf("cntM = %d\n",rec->cntM);

if(rec->cntX > TH\_X)

rec->isM = 2;

else if(rec->cntM > TH\_M)

rec->isM = 1;

else

rec->isM = 0;

return 0;

}

int GetError(pREC1 rec){

int err = 0;

int index[10];

for(int i=9;i>0;i--)

index[i] = index[i-1];

index[0] = rec->index;

int diff = 0;

for(int i=0;i<9;i++)

diff += (index[i] - index[i+1]);

if(diff == 0)

err = 1;

rec->error = err;

return 0;

}

int RotGap(int rot){

static int d2;

static int d1;

static int d0;

d2 = d1;

d1 = d0;

d0 = rot;

int d = d1;

if((d1 > d2) && (d1 > d0))

d = (d2 + d0) /2;

return d;

}

int RotFIR5(int rot\_now){

static int rot\_d[6];

for(int i=5;i>0;i--)

rot\_d[i] = rot\_d[i-1];

rot\_d[0] = rot\_now;

int sum =0;

sum += (rot\_d[0] \* 8);

sum += (rot\_d[1] \* 4);

sum += (rot\_d[2] \* 2);

sum += (rot\_d[3] \* 1);

sum += (rot\_d[4] \* 1);

int fir5 = sum / 16;

return fir5;

}

int RotAve(int rot\_now,int level){

static int rot\_d[20];

if (level > 19)

level = 19;

for(int i=level;i>0;i--)

rot\_d[i] = rot\_d[i-1];

rot\_d[0] = rot\_now;

int sum = 0;

for(int i=0;i<level;i++){

sum += rot\_d[i];

}

int ave = sum / level;

return ave;

}

int RotModK(int rot){

float k;

float b;

LoadKBTest(&k,&b);

float rot\_f = k \* rot + b;

return rot\_f;

}

int RotMod100(int rot){

int a;

if(rot > 2750)

a = 2800;

else if (rot < 100)

a = 0;

else a = rot;

int b = a /10;

return b\*10;

}

#include <stdio.h>

#include <types.h>

#include <cal.h>

#include <alg.h>

#define EN\_SHOW 0

//unsigned int index;

int CalInit(){

return 0;

}

int CalRec1(pREC1 rec,const U8 \* recBuf){

int head = CalHead(recBuf);

int index = CalIndex(recBuf);

int rot = CalRot(recBuf);

float tCore = CalTCore(recBuf);

float t1 = CalT1(recBuf);

float t2 = CalT2(recBuf);

float oilP = CalOilP(recBuf);

int cntM = CalCntM(recBuf);

int cntX = CalCntX(recBuf);

long sumX = CalSumX(recBuf);

rec->head = head;

rec->index = index;

rec->rot = rot;

rec->t1 = t1;

rec->t2 = t2;

rec->tCore = tCore;

rec->oilP = oilP;

rec->cntM = cntM;

rec->cntX = cntX;

return 0;

}

int CalHead(const U8 \* recBuf){

int head = \*(recBuf+0) | (\*(recBuf+1) << 8);

return head;

}

int CalIndex(const U8 \* recBuf){

int index = \*(recBuf+2) | (\*(recBuf+3) << 8);

return index;

}

int CalRot(const U8 \* recBuf){

int rot0 = \*(recBuf+4) | (\*(recBuf+5) << 8);

int rot = rot0 ;

int rot1 = RotGap(rot);

int rot\_d = RotFIR5(rot1);

// int rot\_d = RotAve(rot,5);

int rot\_f = RotModK(rot\_d);

// int rot\_100 = rot\_f;

int rot\_100 = RotMod100(rot\_f);

// if(EN\_SHOW) printf("rot = %d\t rot\_d = %d\n",rot,rot\_d);

return rot\_100;

}

float CalT1(const U8 \* recBuf){

int tH = (\*(recBuf+14) >> 4) | (\*(recBuf+15) << 4);

int tL = \*(recBuf+14) & 0xf;

float t1 = tH + tL \* 0.0625;

if(EN\_SHOW) printf("t1 = %03.2f\ttH = %02x\ttL = %02x\n",t1,tH,tL);

return t1;

}

float CalT2(const U8 \* recBuf){

int tH = (\*(recBuf+16) >> 4) | (\*(recBuf+17) << 4);

int tL = \*(recBuf+16) & 0xf;

float t2 = tH + tL \* 0.0625;

if(EN\_SHOW) printf("t2 = %03.2f\ttH = %02x\ttL = %02x\n",t2,tH,tL);

return t2;

}

float CalTCore(const U8 \* recBuf){

int tH = (\*(recBuf+10) >> 4) | (\*(recBuf+11) << 4);

int tL = \*(recBuf+10) & 0xf;

float tCore = tH + tL \* 0.0625;

if(EN\_SHOW) printf("tCore = %03.2f\ttH = %02x\ttL = %02x\n",tCore,tH,tL);

return tCore;

}

float CalOilP(const U8 \* recBuf){

int oilP0 = \*(recBuf+12) | (\*(recBuf+13) << 8);

float oilP = oilP \* 1.2 + 100;

return oilP;

}

int CalCntM(const U8 \* recBuf){

int cntM0 = \*(recBuf+6) | (\*(recBuf+7) << 8);

int cntM = cntM0;

return cntM;

}

int CalCntX(const U8 \* recBuf){

int cntX0 = \*(recBuf+8) | (\*(recBuf+9) << 8);

int cntX = cntX0;

return cntX;

}

int CalSumX(const U8 \* recBuf){

int SumX0 = \*(recBuf+20) | (\*(recBuf+21) << 8) | (\*(recBuf+22)<<16) | (\*(recBuf+23)<<24);

int SumX = SumX0;

return SumX;

}

#include <stdio.h>

#include <string.h>

#include <wiringPiI2C.h>

#include <commu.h>

#include <types.h>

#define LEN\_TPKG 32

int fd;

int I2C\_Init(){

fd = wiringPiI2CSetup(0x30);

printf("I2C setup ret = %d\n",fd);

return 0;

}

int I2C\_Test(){

//int q = wiringPiI2CWriteReg8(fd,0x12,0x34);

int q = wiringPiI2CReadReg8(fd,0x56);

printf("read q = %d\n",q);

return 0;

}

int GetTPKG(U8 \* recBuf){

for(int i=0;i<LEN\_TPKG;i++){

U8 ret = wiringPiI2CReadReg8(fd,i);

\*(recBuf+i) = ret;

}

return 0;

}

int ShowTPKG(const U8 \*recBuf){

for(int i=0;i<LEN\_TPKG;i++){

printf("\treg %d = 0x%02x",i,\*(recBuf+i));

if(i%4 == 3)

printf("\n");

}

return 0;

}

int TPKGIsNew(U8 \* recBuf,U8 \* recBufOld){

int isSame = 0;

int index = \*(recBuf+2) | (\*(recBuf+3) << 8);

int indexOld = \*(recBufOld+2) | (\*(recBufOld+3) << 8);

if(index == indexOld){

// printf("Same : index = %04x\n",index);

isSame = 1;

}

else {

// printf("Diff : index = %04x\n",index);

// printf("Diff : indexOld = %04x\n",indexOld);

isSame = 0;

}

memcpy(recBufOld,recBuf,LEN\_TPKG \* sizeof(U8));

return isSame;

}

#include <stdio.h>

#include <string.h>

#include <record.h>

#include <types.h>

int SaveRec1(REC1 rec){

char fn[] = "rec1.dat";

FILE \* fid = fopen(fn,"w");

fwrite(&rec,1,sizeof(REC1),fid);

fclose(fid);

return 0;

}

int SaveRec11(REC1 rec){

char fn[] = "../rec11.dat";

FILE \* fid = fopen(fn,"w");

fprintf(fid,"%08x\n%d\n%d\n",rec.head,rec.index,rec.rot);

//fprintf(fid,"%.2f\n%.2f\n%.2f\n%.2f\n",rec.t1,rec.t2,rec.tCore,rec.oilP);

fprintf(fid,"%.2f\n%.2f\n%.2f\n%.2f\n",rec.t1,rec.t2,rec.tCore,rec.oilP);

fprintf(fid,"%d\n%d\n",rec.cntM,rec.cntX);

fprintf(fid,"%d\n%d\n",rec.isM,rec.error);

fclose(fid);

return 0;

}

int LoadKBTest(float \*k,float \*b){

//\*k = 1.4;

//\*b = -840;

\*k = 1.3333333;

\*b = -666.666667;

return 0;

}

int LoadKB(float \*k,float \*b){

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include "opencv2/opencv.hpp"

#include "gui\_types.h"

#include "gui\_action.h"

#include "gui\_mat.h"

#include "gui\_dm.h"

char winName[20];

void \*GUI\_Action(void \* args){

printf("...GUI\_Action: \n");

strcpy(winName,"XDM-I");

struct timeval tv;

BSHOW blockShow;

CvCapture \*cam;

IplImage \*mat;

IplImage \*matRef;

mat = cvCreateImage(cvSize(MAT\_W,MAT\_H),IPL\_DEPTH\_8U,3);

cvNamedWindow(winName,CV\_WINDOW\_NORMAL);

cvSetWindowProperty(winName,CV\_WND\_PROP\_FULLSCREEN,CV\_WINDOW\_FULLSCREEN);

BuildBase(mat);

matRef = cvCloneImage(mat); //matRef is base

cam = cvCaptureFromCAM(0);

while(1)

{

cvCopy(matRef,mat);

GetBlockTest(&blockShow);

GetBlock(&blockShow);

ShowStat(mat,blockShow);

ShowRPM(mat,blockShow);

ShowTem(mat,blockShow);

ShowFlag(mat,blockShow);

BuildCam(mat,cam);

cvShowImage(winName,mat);

char key = cvWaitKey(100);

if(key == 'q')

break;

}

}

int CamInit(int cid){

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "gui\_types.h"

#include "gui\_dm.h"

#define EN\_SHOW 0

int GetBlockTest(pBSHOW pBlock){

pBlock->rpm = 2800;

pBlock->tem = 85.1;

pBlock->flag = 0;

pBlock->err = 0;

return 0;

}

int GetBlock(pBSHOW pBlock){

FILE \* fp = fopen("../../rec11.dat","r");

char st[20]; memset(st,0,20\*sizeof(char));

fgets(st,20,fp); //head

fgets(st,20,fp); //index

fgets(st,20,fp); //rot

pBlock->rpm = atoi(st);

fgets(st,20,fp); //t1

pBlock->tem = atof(st);

if(EN\_SHOW) printf("tem = %3.2f\n",pBlock->tem);

fgets(st,20,fp); //t2

fgets(st,20,fp); //tCore

fgets(st,20,fp); //oilP

fgets(st,20,fp); //cntM

fgets(st,20,fp); //cntX

fgets(st,20,fp); //isM

int isM = atoi(st);

if(isM != 0)

pBlock->flag = isM;

fgets(st,20,fp); //error

int err = atoi(st);

pBlock->err = err;

fclose(fp);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include "opencv2/opencv.hpp"

#include "gui\_mat.h"

int BuildBase(IplImage \* mat){

IplImage \*mBase;

char fnBase[200]; memset(fnBase,0,200\*sizeof(char));

getcwd(fnBase,200);

strcat(fnBase,"/pic/base4.png");

mBase = cvLoadImage(fnBase);

cvResize(mBase,mat);

// cvCopy(mBase,mat);

// cvShowImage("base2",mat);

return 0;

}

int BuildTitle(IplImage \* mat){

IplImage \* mTitle;

char fn[200]; memset(fn,0,200\*sizeof(char));

getcwd(fn,200);

strcat(fn,"/pic/Title.png");

printf("fnTitle = %s\n",fn);

mTitle = cvLoadImage(fn);

// cvShowImage("title",mTitle);

IplImage \*mTitle2 = cvCreateImage(cvSize(700,100),mTitle->depth,mTitle->nChannels);

cvResize(mTitle,mTitle2);

// cvShowImage("title",mTitle2);

CvRect roiTitle = cvRect(250,0,700,100);

cvSetImageROI(mat,roiTitle);

cvCopy(mTitle2,mat);

cvResetImageROI(mat);

return 0;

}

int BuildLogo(IplImage \* mat){

return 0;

}

int BuildCam(IplImage \* mat, CvCapture \* cam){

IplImage \* mCam;

IplImage \* mCam2;

int camW = 360;

int camH = 360;

int camTop = 180;

int camLeft = 410;

if(cam == NULL){

char fnCam[200]; memset(fnCam,0,200\*sizeof(char));

getcwd(fnCam,200);

strcat(fnCam,"/pic/noCam.jpg");

mCam = cvLoadImage(fnCam);

// printf("no cam = %d\n",cam);

}

else{

mCam = cvQueryFrame(cam);

}

// cvShowImage("Cam",mCam);

mCam2 = cvCreateImage(cvSize(camW,camH),mCam->depth,mCam->nChannels);

cvResize(mCam,mCam2);

CvRect roiTitle = cvRect(camLeft,camTop,camW,camH);

cvSetImageROI(mat,roiTitle);

cvCopy(mCam2,mat);

cvResetImageROI(mat);

// cvReleaseImage(&mCam);

cvReleaseImage(&mCam2);

return 0;

}

int ShowRPM(IplImage \* mat,BSHOW block){

CvFont fontRPM;

double hS =1.8;

double vS =1.8;

int lineW = 2;

char cRPM[40]; memset(cRPM,0,40\*sizeof(char));

int rpm = block.rpm;

sprintf(cRPM,"%d",rpm);

// fontRPM = cvFontQt("Times");

// cvAddText(mat,cRPM,cvPoint(300,250),&fontRPM);

cvInitFont(&fontRPM,CV\_FONT\_HERSHEY\_SIMPLEX,hS,vS,0,lineW);

if(rpm < 2300)

cvPutText(mat,cRPM,cvPoint(240,260),&fontRPM,CV\_RGB(255,0,0));

else

cvPutText(mat,cRPM,cvPoint(240,260),&fontRPM,CV\_RGB(20,200,20));

return 0;

}

int ShowTem(IplImage \* mat,BSHOW block){

CvFont font;

double hS =1.8;

double vS =1.8;

int lineW = 2;

float tem = block.tem;

char cTem[40]; memset(cTem,0,40\*sizeof(char));

sprintf(cTem,"%02.1f",block.tem);

// fontRPM = cvFontQt("Times");

// cvAddText(mat,cRPM,cvPoint(300,250),&fontRPM);

cvInitFont(&font,CV\_FONT\_HERSHEY\_SIMPLEX,hS,vS,0,lineW);

if(tem <= 69)

cvPutText(mat,cTem,cvPoint(240,370),&font,CV\_RGB(20,200,20));

else

cvPutText(mat,cTem,cvPoint(240,370),&font,CV\_RGB(255,0,0));

return 0;

}

int ShowFlag(IplImage \* mat,BSHOW block){

int flag = block.flag;

IplImage \*mFlag;

IplImage \*mFlag2;

char fnFlag[200]; memset(fnFlag,0,200\*sizeof(char));

getcwd(fnFlag,200);

if(flag == 1)

strcat(fnFlag,"/pic/standby/1.png");

else if(flag == 2)

strcat(fnFlag,"/pic/working/1.png");

else

return 0;

mFlag = cvLoadImage(fnFlag);

// cvShowImage("Flag",mFlag);

int mW = 100; //width

int mH = 100; //height

mFlag2 = cvCreateImage(cvSize(mW,mH),mFlag->depth,mFlag->nChannels);

cvResize(mFlag,mFlag2);

// cvShowImage("title",mFlag2);

CvRect roiFlag = cvRect(250,440,mW,mH);

cvSetImageROI(mat,roiFlag);

cvCopy(mFlag2,mat);

cvResetImageROI(mat);

cvReleaseImage(&mFlag);

cvReleaseImage(&mFlag2);

return 0;

}

int ShowStat(IplImage \* mat,BSHOW block){

int stat = block.err;

IplImage \*mStat;

IplImage \*mStat2;

char fnStat[200]; memset(fnStat,0,200\*sizeof(char));

getcwd(fnStat,200);

if(stat == 0)

strcat(fnStat,"/pic/sysOK.png");

else if(stat == 1)

strcat(fnStat,"/pic/sysError.png");

else

return 0;

mStat = cvLoadImage(fnStat);

// cvShowImage("Stat",mStat);

int mW = 480; //width

int mH = 50; //height

mStat2 = cvCreateImage(cvSize(mW,mH),mStat->depth,mStat->nChannels);

cvResize(mStat,mStat2);

// cvShowImage("title",mStat);

CvRect roiStat = cvRect(50,130,mW,mH);

cvSetImageROI(mat,roiStat);

cvCopy(mStat2,mat);

cvResetImageROI(mat);

cvReleaseImage(&mStat);

cvReleaseImage(&mStat2);

return 0;

}