DMA2440 Camera Lab

課程：嵌入式系統與軟體工程
開發學校：台大電機系
王勝德 教授

Lab Objectives

- Objectives
  - To learn frame buffer programming and basic Qt GUI
  - To learn the interface to CMOS camera
  - Design an image display and taking application on DMA-2440 Development Board using frame buffer
  - Adding Qt GUI Interface to the image display and taking application

Lab contents and suggestions

- In this lab, we have two exercises
  - Ex1: Display colors on LCD
  - Ex2: Enhance the IPC between the form1 and testcamera programs
- And a suggestion for term project topics
  - Term project topic: QR image acquisition and decoding

Equipments required

- PC host
  - A Linux PC or
  - A Windows PC plus a guest Linux running on a Virtual Machine Software (such as Vmware)
- Embedded Development Boards
  - DMA-2440 + CMOS camera + Linux BSP CD
  - Gcc toolchain 3.3.2
  - jpegsrc.v6b

Background Knowledge

- Frame buffer
  - The frame buffer is the memory on the video card (video controller)
- System calls
  - open(), close()
  - mmap(), munmap()
    - void *mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset);
    - int munmap(void *start, size_t length);
    - map or unmap files or devices into memory
  - ioctl()
    - int ioctl(int d, int request, ...);
    - The ioctl() function manipulates the underlying device parameters of special files.

Frame buffer driver

- Frame buffer driver normally has three layers
  - The top layer is the basic console driver drivers/char/console.c,
    - text console.
  - in the middle layer, or drivers/video/fbcon.c
    - video mode.
  - the lowest layer is a very hardware-specific driver, support the different hardware aspects of the video card
- Frame buffer speeds both drawing and overall performance.
  - like enabling/disabling the video card controller, the depths and modes supported, the palettes, etc.
- The device associated with the frame buffer is /dev/fb0 (Major Number 29, Minor Number 0).

mmap()

- #include <sys/mman.h>
- void *mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset);
- int munmap(void *start, size_t length);
- The mmap() function asks to map length bytes starting at offset offset from the file (or other object) specified by the file descriptor fd into memory, preferably at address start. This latter address is a hint only, and is usually specified as 0.
- The actual place where the object is mapped is returned by mmap().

Frame Buffer used in testcamera.c

- Trace the testcamera.c file, we can find out fb_buf is for LCD output

```
699         screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
700         fb_buf = (char *)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED,
701                               fbfd, 0);
```

  and buf is for image raw data(yuv422).

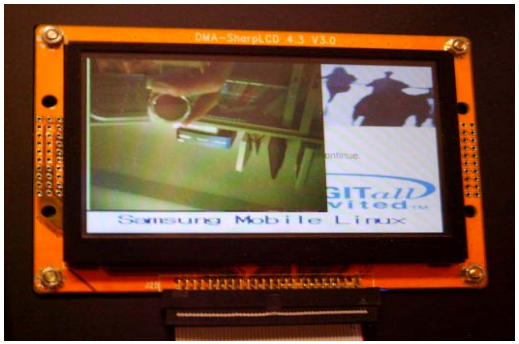
```
796                     buf = (__u8 *)mmap(0, vm.size, PROT_READ, MAP_SHARED, fd, 0);
797                     if((int)buf==-1) {
798                             printf("mmap camera fail!\n");
799                             mmap_camera = 0;
800                     } else
801                             puts("mmap camera ok.\n");
802             }
803     }
804
805     if(!mmap_camera) {
806             puts("allocate memory for camera buffer.\n");
807             buf = malloc(image_width*image_height*2);
```

- We can get camera image from buf and show image to LCD by fb_buf, just follow the image format.

I2C control

- We also can control HW feature by I2C bus. (Just refer to CMOS sensor spec. to call I2C API)

```
495         i2c_read(fd, CAMERA_PIDH, id+2);
496         i2c_read(fd, CAMERA_PIDL, id+3);
497         printf("product ID is 0x%04x\n", (id[2]<<8)|id[3]);
498
499         for(i=0; i<CAMERA_REGS; i++) {
500                 if(camera_regs[i].subaddr==CHIP_DELAY)
501                         delay_ms(camera_regs[i].value);
502                 else if(i2c_write(fd, camera_regs[i].subaddr, camera_regs[i].value))
503                         printf("write subaddr 0x%x fail!\n", camera_regs[i].subaddr);
504                 //delay_ms(2);
505                 //printf("%d\n", i);
506         }
507         if(cpu_type==1) {       //s3c2440a
508                 i2c_write(fd, 0x15, 0x00);      //always has PCLK
509                 i2c_write(fd, 0x3a, 0x01);      //YCbYCr order
```



Camera Device Control Registers

Table 7 Device Control Register List (Continued)

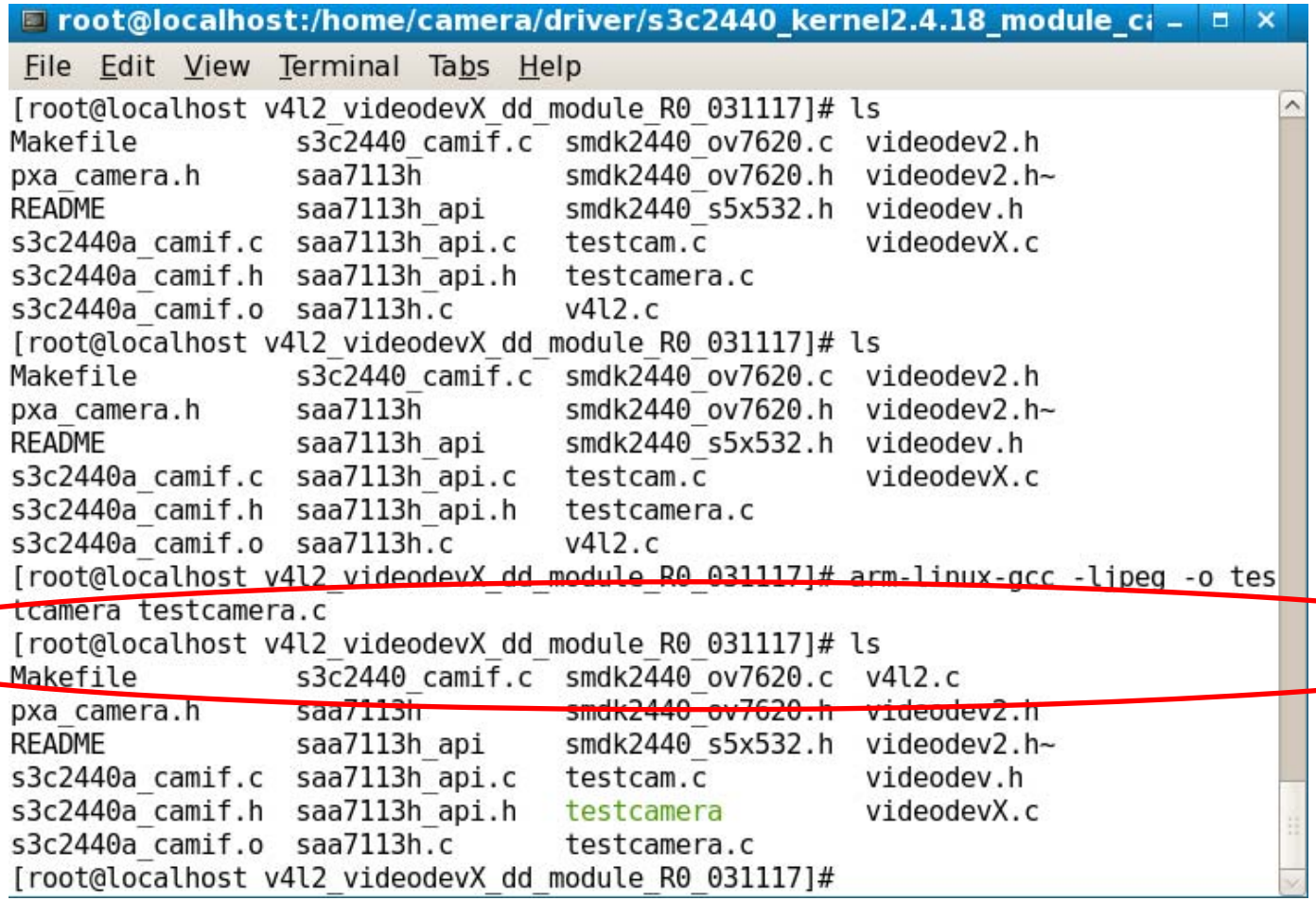
Address (Hex)	Register Name	Default (Hex)	R/W	Description
3E	COM14	0E	RW	Common Control 14 Bit[7:2]: Reserved Bit[1]: Enable edge enhancement for YUV output (effective only for YUV/RGB, no use for Raw data) Bit[0]: Edge enhancement option 1: Double edge enhancement factor
3F	EDGE	88	RW	Edge Enhancement Adjustment Bit[7:4]: Edge enhancement threshold Bit[3:0]: Edge enhancement factor
40	COM15	C0	RW	Common Control 15 Bit[7:6]: Data format - output full range enable 00: Output range: [00] to [FF] 01: Output range: [01] to [FE] 1x: Output range: [10] to [F0] Bit[5:4]: RGB 555/565 option (must set COM7[2] high) x0: Normal RGB output 01: RGB 565 11: RGB 555 Bit[3:0]: Reserved
41	COM16	00	RW	Common Control 16 Bit[7:2]: Reserved Bit[1]: Color matrix coefficient double option Bit[0]: RB average option for interpolation
42	COM17	08	RW	Common Control 17 Bit[7]: B channel pre-gain Bit[6]: R channel pre gain Bit[5:3]: Reserved Bit[2]: Select single frame out Bit[1]: Tri-state output Bit[0]: AGC maximum gain 16x
43-4F	RSVD	XX	–	Reserved

Compile Application Program

- Install tool chain 3.3.2
- Install jpegsrc.v6b
- Copy all files of camera driver from attached CD path:
  \DMA2440V45\Linux data\driver\s3c2440_kernel2.4.18_module_camera_v4l\
  v4l2_videodevX_dd_module_R0_031117
  to Linux OS work folder.
- Copy the 3 header files below to folder above.

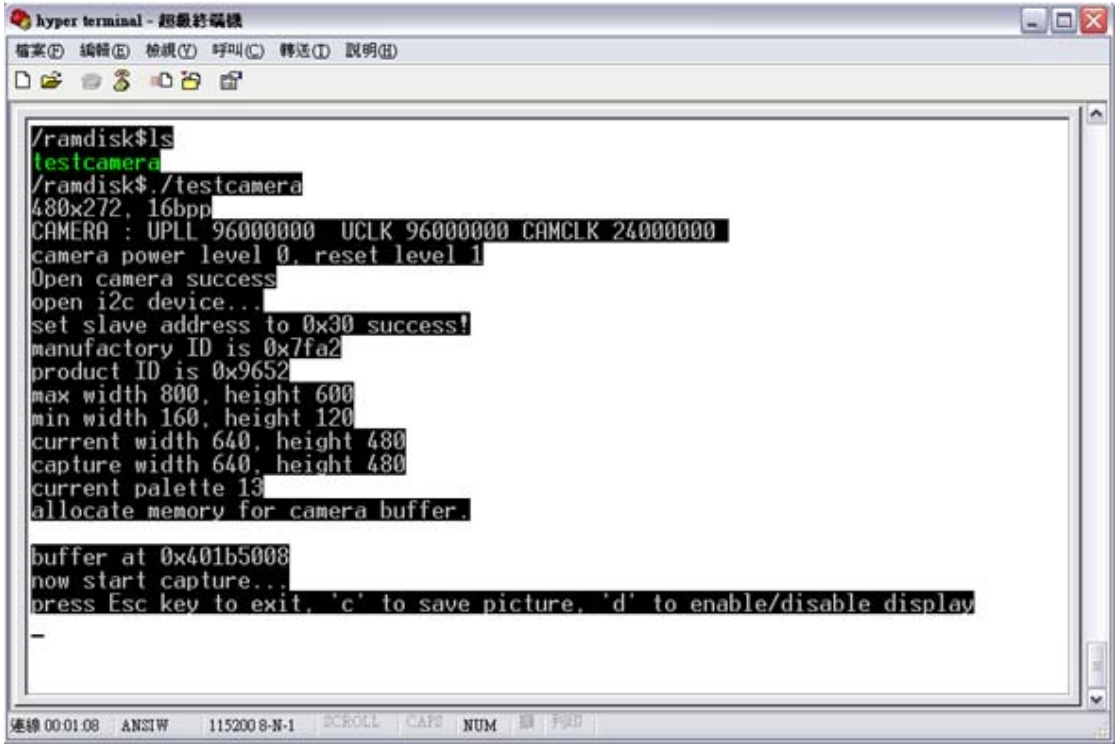
- Command to compile out App file.
  "arm-linux-gcc –ljpeg –o testcamera testcamera.c"
- Download testcamera file to the embedded board and run.

Compile testcamera.c



Download App and testing

- Open Hyper Terminal(115200 8 n 1 n) as console to DMA2440.
- Change folder to ramdisk and download testcamera.
- Chmod this file and run it.



Ex1: Display colors on LCD

Step1: 建立一簡單的程式，能夠在LCD上面畫出顏色。

(Host)# vi display.c

將緩衝區映射到user space之後，加入下面的程式碼。

```
for(i=0;i<screensize/2;i++)
{     fbp+2*i=0x1f;
      fbp+2*i+1=0x00; }
```

Step2: 開始編譯

(Host)# arm-linux-gcc –o display display.c

Step3: 執行display，看會顯示什麼？

frame buffer 的資料顏色格式是否為如下？

16bit : high: RRRRRGGG low: GGGBBBBB

Display.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/mman.h>
#include <unistd.h>
#include <linux/delay.h>
#include <sys/ioctl.h>
#include <linux/fb.h>
int main()
{
    int fbfd = 0;
    int i=0;
    struct fb_var_screeninfo vinfo;
    struct fb_fix_screeninfo finfo;
    long int screensize = 0;
    unsigned char *fbp;
    fbfd = open("/dev/fb0", O_RDWR);
    ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo);
    ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo);
    screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
```

```
    fbp=(unsigned char*)mmap(0,screensize,PROT_READ|PROT_WRITE,MAP_SHARED, fbfd,0);
    printf("x:%d y:%d bpp:%d",vinfo.xres,vinfo.yres,vinfo.bits_per_pixel);

    for(i=0;i<screensize/4;i++)
    {
       // 16bit : high: RRRRRGGG  low: GGGBBBBB

      *(fbp+2*i) = 0x1f;
      *(fbp+2*i+1)= 0x00;
     }

   for(i=screensize/4;i<screensize/2;i++)
  {
     *(fbp+2*i)=0x00;
     *(fbp+2*i+1)=0x00;
   }
     return 0;
}
```

QT GUI Interface for testcamera.c

Objectives and Howto do

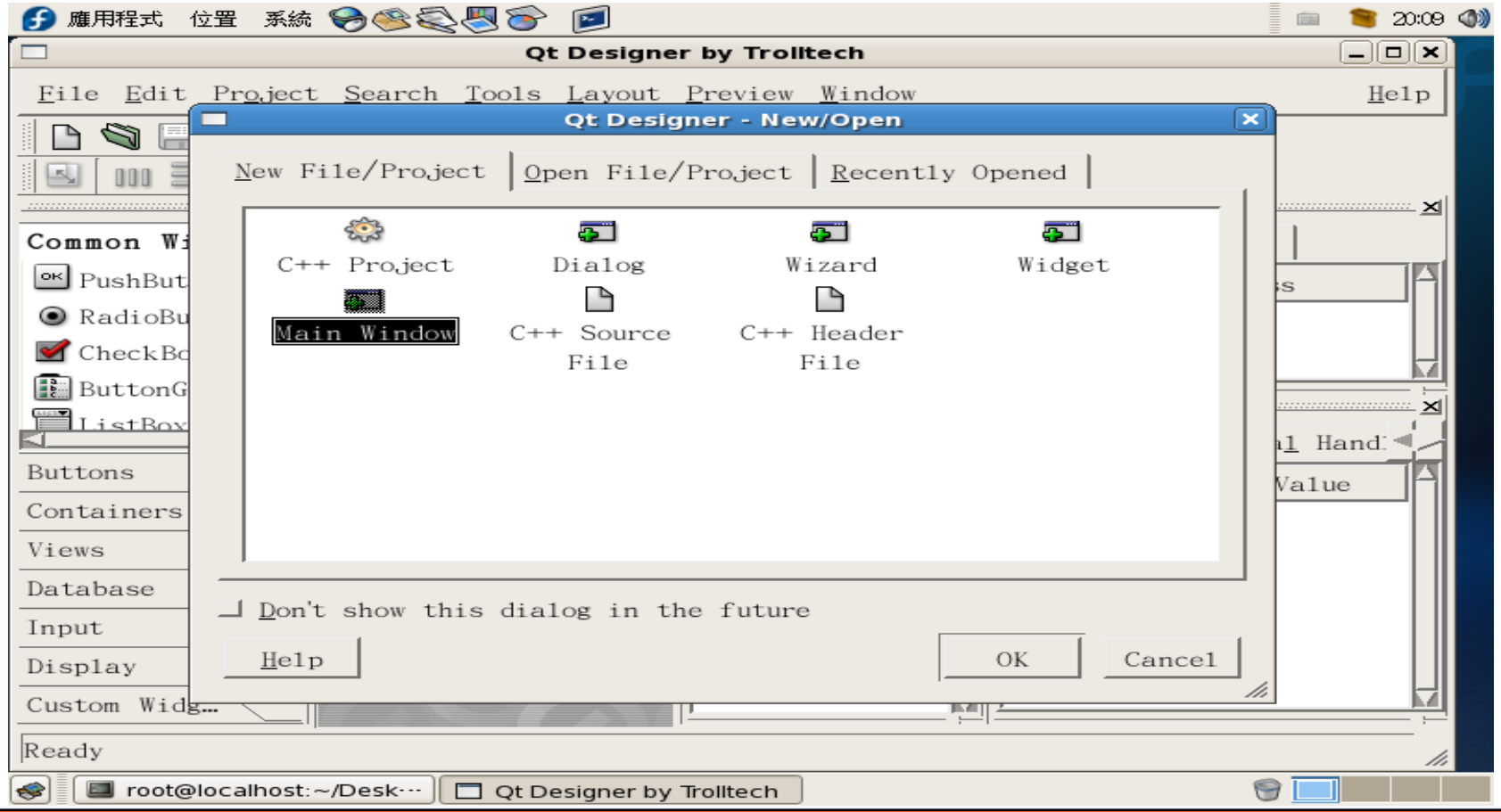
- In the first part, we have designed a testcamera program to display video and capture images
- We want to add a Qt GUI program to control the operations of the testcamera program, called form1
- testcamera and form1 can communicate through any interprocess communication mechanism (IPC) such as
  - POSIX thread library, fifo, popen function
    - fifo為GNU的標準C function，可供開發人員將訊息寫入fifo檔中，提供process間的資料交換功能。
    - popen為GNU的標準C function，可將某一process的stdout導入一檔案中，使用者便可從此檔案得到process的stdout內容。
    - POSIX thread library 為linux的thread library,以提供multi-thread的功能，QT2.3亦提供QThread達到multi-thread的功能。
  - Qt2.3 does not has QProcess API.

Steps

- Please refer the lab guide Lab-Qt-guess-num-game.pdf for the Qt2 program design example
- Use Qt designer to design the graphic components for form1, called form1.ui
- To generate form1.h
  - uic -o form1.h form1.ui
- To generate form1.cpp
  - uic -i form.h -o form1.cpp form1.ui
- Design the main program
- Test the program

Start QT designer

- **cd qt-X11/bin and ./designer**



Main program

- Each Qt program should include <qapplication.h>

- #include "form1.h"
- #include <qapplication.h>

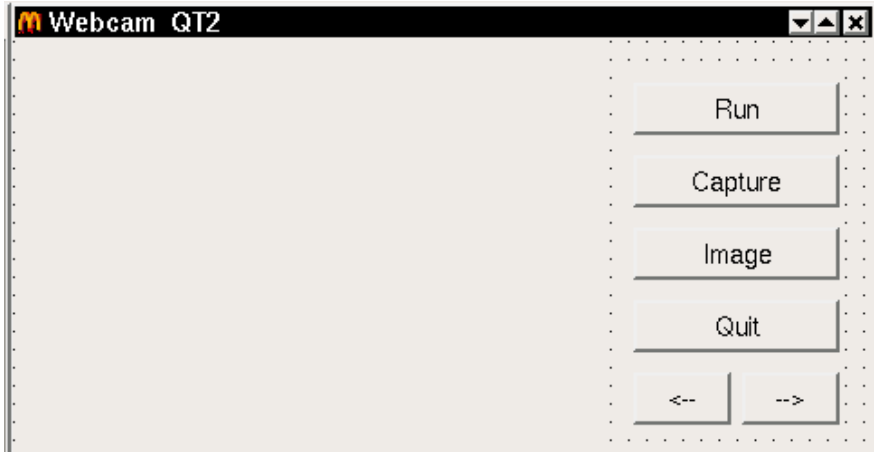
- int main(int argc,char **argv)
- {
- QApplication a(argc,argv);
- form1 dlg;
- QObject::connect(&dlg,SIGNAL(clicked()),&a,SLOT(quit()));
- a.setMainWidget(&dlg);
- dlg.show();
- return a.exec();

Program user interface layout

- PixmapLabel1 = new QLabel(this, "PixmapLabel1");
- PixmapLabel1->setEnabled(TRUE);

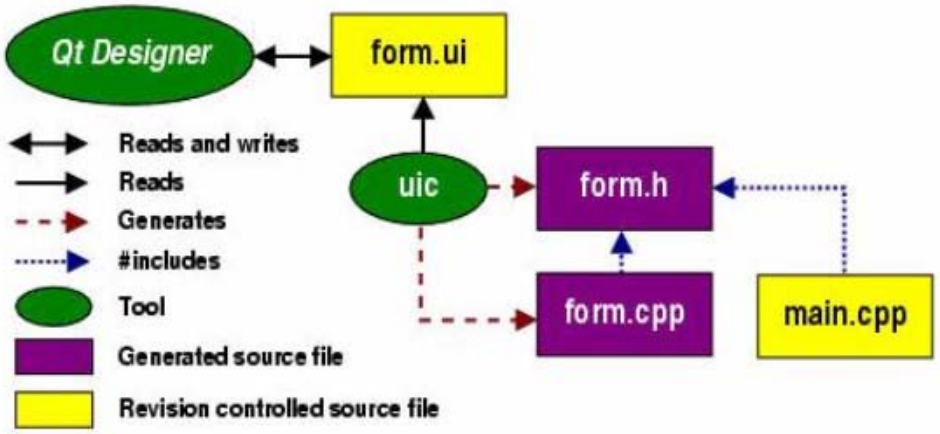
- PushButton1 = new QPushButton(this, "PushButton1");
- PushButton1->setGeometry(QRect(340, 20, 120, 35));
- PushButton1->setText(tr("Webcam"));

- Three modes
- mode = 1 : run mode(Default)
- mode = 2 : capture mode
- mode = 3 : showimagemode

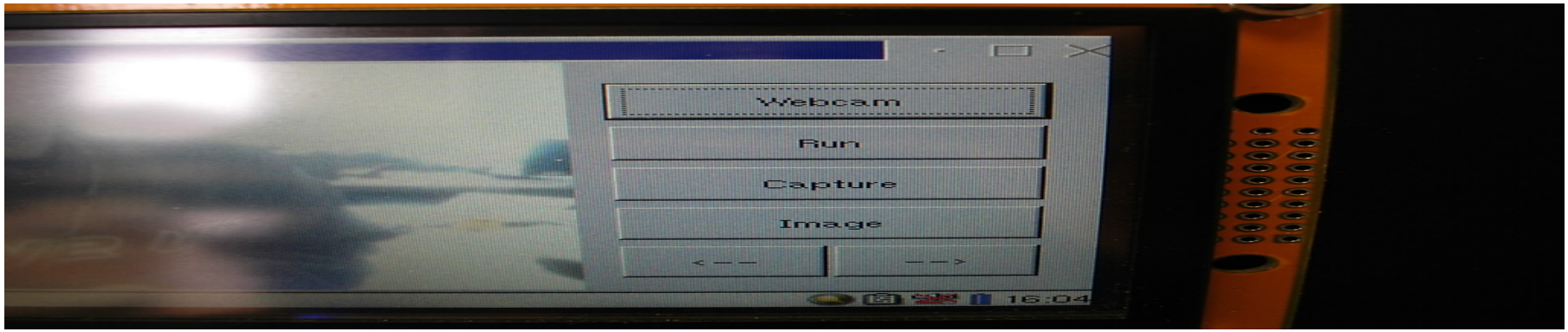


Use uic to generate form1.h form1.cpp

- uic: form.ui → form.h form.cpp
  uic -o form1.h form1.ui
  uic -i form.h -o form1.cpp form1.ui



Signal and Slots Connections



- We change the user interface layout by adding some new buttons.
- // signals and slots connections
- connect(PushButton1, SIGNAL(clicked()),this, SLOT(run()));
- connect(PushButton2, SIGNAL(clicked()),this, SLOT(run2()));
- connect(PushButton2, SIGNAL(clicked()),PushButton2, SLOT(animateClick()));
- connect(PushButton3, SIGNAL(clicked()),this, SLOT(capture()));
- connect(PushButton4, SIGNAL(clicked()),this, SLOT(Showimage()));
- connect(PushButton5, SIGNAL(clicked()),this, SLOT(Left()));
- connect(PushButton6, SIGNAL(clicked()),this, SLOT(Right()));
- init();

Program Execution Snapshot

Left and Right buttons

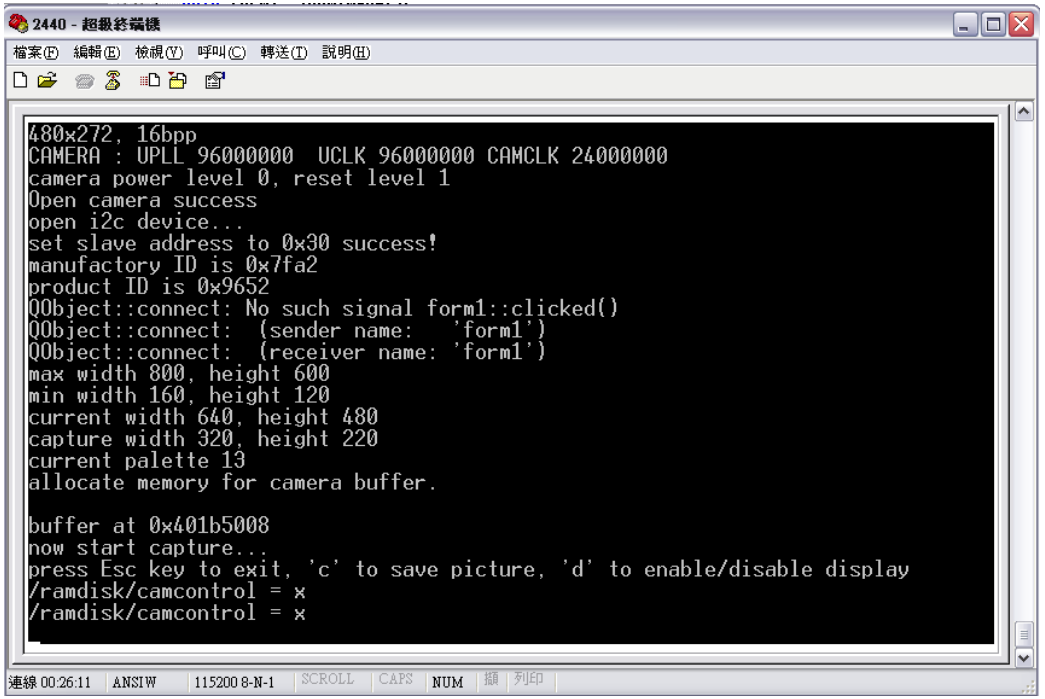
```
void form1::Left(){
    //setCaption( tr( "Webcam_QT2-mode4" ) );
if(showimageact == 1){
        if(imageindex > 0){
        imageindex -= 1;
        sprintf(filename,"%s%04d.jpg", directory, imageindex);
        }
QPixmap pixmap;
if (pixmap.load(filename))
PixmapLabel1->setPixmap(pixmap);
setCaption( tr( filename ) );
}
}
```

```
void form1::Right(){
    //setCaption( tr( "Webcam_QT2-mode5" ) );
if(showimageact == 1){
        if(imageindex < 10000){
        imageindex += 1;
        sprintf(filename,"%s%04d.jpg", directory, imageindex);
        }
QPixmap pixmap;
if (pixmap.load(filename))
PixmapLabel1->setPixmap(pixmap);
setCaption( tr( filename ) );
}
}
```

```
/ramdisk$ls
form1  tcam
/ramdisk$chmod 777 *
/ramdisk$ls
form1  tcam
/ramdisk$./form1  & ./tcam_
```

Program functionality

- Quit -> Quit all the program
- Run -> Camera run mode
- Capture -> Camera capture mode
- Image -> Show image mode
- Left -> Show image in decreasing number
- Right -> Show image in increasing number



Related Links

- Qt
- http://www.qiliang.net/qt/index.html
- Programming with Qt (O'Reilly)
- http://www.oreilly.com.tw/product_c.php?id=a057
- Qt/Qtopia 中文討論區
- http://www.qtopia.org.cn/phpBB2/

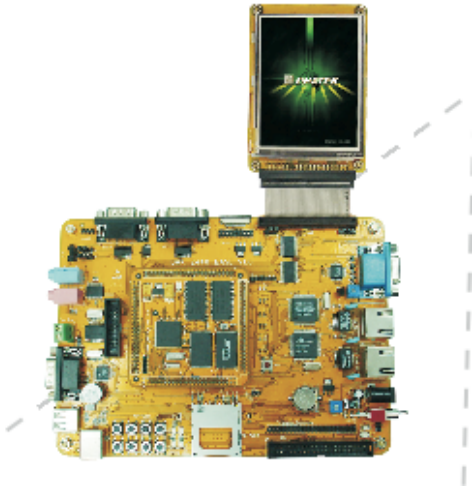
Ex2: Enhance the IPC between the form1 and testcamera programs

- Currently, we use regular file /ramdisk/camcontrol to exchange the message between form1 and testcamera
  - As can be seen in form1.cpp
- Modify the form1 and testcamera programs to use other IPC mechanisms, such as
  - POSIX thread library,
  - fifo,
  - popen function

Term project topic: QR image acquisition and decoding

- Extend the testcamera.c with a QR code decomposition algorithm to get the QR code from the captured image.
- Write a client-server program using network socket programming
  - Embedded system as a client to send the QR code
  - Using a PC as a server to receive the OR code

Client Architecture



I2C

Camera Driver

Raw Data

Request

APP

Image Processing

Reconstructed Data

Follow ISO/IEC 18004

QR Decoder

Information

DMA-2440

CPU:Samsung S3C2440A , 400MHz

SDRAM : 64M ,NOR Flash:2M,NAND Flash 64M

Camera: 1.3 mega pixel CMOS sensor camera

OS:Linux 2.6.14

System Architecture