**SDK and system resources of B3VDU**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Chen Yong | Jul 8, 2022 | Initial draft | 1.0.1 |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Scope 4](#__RefHeading___Toc1662_3499696116)

[2 System 4](#__RefHeading___Toc1664_3499696116)

[3 Enviroment 4](#__RefHeading___Toc1666_3499696116)

[3.1 VDU 4](#__RefHeading___Toc1669_3499696116)

[3.2 Programming computer 5](#__RefHeading___Toc1671_3499696116)

[4 B3VDU system resources 5](#__RefHeading___Toc2357_3499696116)

[4.1 External interface 5](#__RefHeading___Toc2359_3499696116)

[4.1.1 LCD 5](#__RefHeading___Toc2361_3499696116)

[4.1.2 UART 5](#__RefHeading___Toc2363_3499696116)

[4.1.3 Ethernet 5](#__RefHeading___Toc2367_3499696116)

[4.1.4 KeyPad 5](#__RefHeading___Toc2369_3499696116)

[4.1.5 SDI Input 6](#__RefHeading___Toc2371_3499696116)

[4.1.6 CAN 6](#__RefHeading___Toc2373_3499696116)

[4.2 Internal Interface 6](#__RefHeading___Toc2379_3499696116)

[4.2.1 I2C 6](#__RefHeading___Toc2381_3499696116)

[4.2.2 video registers 6](#__RefHeading___Toc2383_3499696116)

[4.2.3 OSD memory 7](#__RefHeading___Toc2395_3499696116)

[4.2.4 iio sensor 7](#__RefHeading___Toc7529_1605989181)

[4.2.5 Watchdog 7](#__RefHeading___Toc6414_1605989181)

[5 Programming 7](#__RefHeading___Toc6416_1605989181)

[5.1 A simple firmware 7](#__RefHeading___Toc6418_1605989181)

[5.1.1 Edit 7](#__RefHeading___Toc6420_1605989181)

[5.1.2 Compile 8](#__RefHeading___Toc6422_1605989181)

[5.1.3 Run firmware 9](#__RefHeading___Toc6424_1605989181)

# Scope

# System

VDU

VCU

Programming

computer

Figure is the full system physical linking. The devices and linking for programming are in the dotted line box.

# Enviroment

## VDU

Defaut setting:

* static IP (192.168.9.115)
* username: root
* password: root

The user can download and upload files to VDU with sftp client software.

The user can control the VDU with the serial port debug application(e.g: teraterm).

The user also can log in and control the VDU with putty or ssh.

## Programming computer

* This computer is for programming the software fo the VDU. The Ubuntu(20.04) is required and verified to work.
* This computer has the same subnet IP address with the VDU.
* The user need install the requirement software eg Filezilla(sftp client), GTKTerm(serial debug) or ssh for downloading, uploading and debugging the VDU files.
* The user run sdk.sh(from teamone) to install the sdk compiler.
* <your-sdk-dir>/environment-setup-cortexa72-cortexa53-xilinx-linux includes the enviroment setup for the compiler. The user need setup the enviroment of the compiler based on their SDK requirement.

# B3VDU system resources

## External interface

### **LCD**

.LVDS: LCD video data signal.

.LCD DIM PWM output: adjust LCD bright.

*output control reference 4.2.2*

### UART

This port is link to the VCU for the VDU switching command.

***Format****:* ***TIA/EIA-422***

***Device name****: /dev/ttyS1*

***working mode****:Duplex*

***flow control****: no*

### Ethernet

1**0/100/1000 BASE-T**

**ifname: eth0**

### KeyPad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key No | Gpio port | Key name | Linux code |  |
| KEY\_DIN\_1 | <axi\_gpio\_0 0> | key1 | KEY\_1(2) | FRONT |
| KEY\_DIN\_2 | <axi\_gpio\_0 1> | key2 | KEY\_2(3) | REAR |
| KEY\_DIN\_3 | <axi\_gpio\_0 2> | key3 | KEY\_3(4) | IR |
| KEY\_DIN\_4 | <axi\_gpio\_0 3> | key4 | KEY\_4(5) | NUC |
| KEY\_DIN\_5 | <axi\_gpio\_0 4> | key5 | KEY\_5(6) | NIGHT |
| KEY\_DIN\_6 | <axi\_gpio\_0 5> | key6 | KEY\_6(7) | MINUS |
| KEY\_DIN\_7 | <axi\_gpio\_0 6> | key7 | KEY\_7(8) | PLUS |
| KEY\_DIN\_8 | <axi\_gpio\_0 7> | key8 | KEY\_8(9) | MENU |

### **SDI Input**

*control reference 4.2.2*

### **CAN**

*Ifname: can0*

*Spare device*

## Internal Interface

### I2C

**device name**: /*dev*/i2c-0

**slave chip**: mb85rc256vfp **slave address**: 0x50

**description**: log data storage

### **video registers**

Registers Physical address: 0xa0000000

Registers Physical size: 0x2000

Registers data: 32bits

|  |  |  |  |
| --- | --- | --- | --- |
| registers | W/R | items | value |
| 0x0000 | W/R | brightness | 0 - 100 |
| 0x0004 | W/R | NVG brightness | 0 - 100  default 10 |
| 0x0008 | W/R | NVG mode | 0 – normal mode  1 – NVG mode |
| 0x000C | W/R | auto\_manu | 0 – auto select video  1 – manual select video |
| 0x0010 | W/R | osdCtrl | 0 – no OSD  1 – overlay OSD |
| 0x0014 | W/R | video source | 0 – video source 0  1 -  video source 1 |
| 0x0018 | W/R | contrast value | 0 – 100 convert to -255 ~ 255  (x- 50) x 255 /50 |
| 0x001C | W/R | color value | 0 – 100 convert to -255 ~ 255  (x- 50) x 255 /50 |
| 0x0100 | read only | status | bit 3-0 -- sdi 4 inputs: 0 normal, 1 no signal  bit 4 -- ddr read: 0 normal, 1 no read operation  bit 5 -- ddr write: 0 normal, 1 no write operation  bit 7-6 – display: 00 normal, 01 no signal, 10 error |

### **OSD memory**

Osd physcial address: 0x70000000

Osd Resolution: 1920x1080

Osd format: RGBA32

### **iio sensor**

**CPU**:/sys/bus/iio/devices/iio\:device0

**Power Current Sensor:/**sys/bus/iio/devices/iio\:device1

### Watchdog

**Device**: /*dev/watchdog0, /dev/watchdog1*

*watchdog0 is controlled by OS.*

*Watchdog1 can be controoled by the user.*

# Programming

## A simple firmware

### Edit

Install Atom as the SDK editor. Install Atom requirement plugin.

Create the Project Folder and Add this project folder into the Atom.

create a new file ‘Makefile’ and a folder ‘src’ in this project folder.

Create a source file ‘main.c’ in the folder ‘src’.

**edit Makefile:**

TARGET := $(notdir $(CURDIR))

CSOURCES := $(wildcard src/\*.c)

COBJECTS := $(CSOURCES:%.c=%.o)

CPPSOURCES := $(wildcard src/\*.cpp)

CPPOBJECTS := $(CPPSOURCES:%.cpp=%.o)

CFLAGS := -O

LDFLAGS :=

LIBS :=

.PHONY: $(TARGET)

$(TARGET): $(COBJECTS) $(CPPOBJECTS)

$(CXX) -o $@ $^ $(LDFLAGS) $(LIBS)

$(COBJECTS): %.o: %.c

$(CC) $(CFLAGS) -o $@ -c $<

$(CPPOBJECTS): %.o: %.cpp

$(CXX) $(CFLAGS) -o $@ -c $<

clean:

rm -rf \*~ .dep $(COBJECTS) $(CPPOBJECTS) $(TARGET)

**edit the main.c:**

#include <stdio.h>

int main()

{

printf(“hello world!\n”);

return 0;

}

### Compile

Open Terminal and Go to your project folder. Run “make”.

Then you get a executable file ‘<your-project-folder>’ in <your-project-folder>.

### Run firmware

* Download the executable file to /*usr/local/bin in the VDU with the sftp client software*
* *Open GTKTerm and connect to the VDU with the USB debug port.(e.g /dev/ttyUSB0)*
* *Disable the original VDU firmware in /etc/init.d/loaduserapp.sh and reboot the VDU*
* *go to the folder /usr/local/bin and run your application*