|  |
| --- |
| **SDK and System Resources of** **B3VDU****STELS PART NO: 2200298216**  |

**Change Index.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Date | Amendment contents | Version | Person |
| 1 | 08 July2022 | First Copy for release | 1.0.1 | Chen Yong |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[1 Scope 4](#_Toc108971434)

[2 System 4](#_Toc108971435)

[3 Environment 5](#_Toc108971436)

[3.1 VDU 5](#_Toc108971437)

[3.2 Programming computer 5](#_Toc108971438)

[4 B3VDU system resources 5](#_Toc108971439)

[4.1 External interface 5](#_Toc108971440)

[4.1.1 LCD 5](#_Toc108971441)

[4.1.2 UART 6](#_Toc108971442)

[4.1.3 Ethernet 6](#_Toc108971443)

[4.1.4Keypad 6](#_Toc108971444)

[4.1.5 SDI Input 6](#_Toc108971445)

[4.1.6 CAN 6](#_Toc108971446)

[4.2 Internal Interface 7](#_Toc108971447)

[4.2.1 I2C 7](#_Toc108971448)

[4.2.2 Video Registers 7](#_Toc108971449)

[4.2.3 OSD Memory 8](#_Toc108971450)

[4.2.4 iio sensor 8](#_Toc108971451)

[4.2.5 Watchdog 8](#_Toc108971452)

[5 Programming 8](#_Toc108971453)

[5.1 A simple firmware 8](#_Toc108971454)

[5.1.1 Edit 8](#_Toc108971455)

[5.1.2 Compile 10](#_Toc108971456)

[5.1.3 Run firmware 10](#_Toc108971457)

# 1 Scope

This document stipulates programming and system resources setup procedures for Video Display Unit (abbreviated as VDU B3 here after). This document will be used as a guideline to system resources and programming of VDU B3 in according to the technical specification requirements.

# 2 System



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

# 3 Environment

## 3.1 VDU

 Default setting:

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

The sftp client software will be used for upload and download the files to the VDU.

The “Tera Team” test software to control the VDU with the serial port debug application.

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

## 3.2 Programming computer

* This computer is for programming the software for 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 environment setup for the compiler. The user need setup the environment of the compiler based on their SDK requirement.

# 4 B3VDU system resources

## 4.1 External interface

### 4.1.1 LCD

LVDS: LCD video data signal.

LCD DIM PWM output: adjust LCD bright.

*output control reference 4.2.2*

### 4.1.2 UART

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

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

***Device name****: /dev/ttySC1*

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

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

### 4.1.3 Ethernet

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

### 4.1.4Keypad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key No | Gpio port | Key name | Linux code |  |
| KEY\_DIN\_1 | <axi\_gpio\_0 0> | key1 | KEY\_1(2) | Trailer ? |
| KEY\_DIN\_2 | <axi\_gpio\_0 1> | key2 | KEY\_2(3) | Reverse Gear |
| KEY\_DIN\_3 | <axi\_gpio\_0 2> | key3 | KEY\_3(4) | Forward Gear |
| KEY\_DIN\_4 | <axi\_gpio\_0 3> | key4 | KEY\_4(5) | TBD |
| KEY\_DIN\_5 | <axi\_gpio\_0 4> | key5 | KEY\_5(6) | TBD |
| KEY\_DIN\_6 | <axi\_gpio\_0 5> | key6 | KEY\_6(7) | TBD |
| KEY\_DIN\_7 | <axi\_gpio\_0 6> | key7 | KEY\_7(8) | TBD |
| KEY\_DIN\_8 | <axi\_gpio\_0 7> | key8 | KEY\_8(9) | TBD |

### 4.1.5 SDI Input

 control reference 4.2.2

### 4.1.6 CAN

*Ifname: can0*

*Spare device*

## 4.2 Internal Interface

### 4.2.1 I2C

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

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

**description**: log data storage

### 4.2.2 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 - 100default 10 |
| 0x0008 | W/R | NVG mode | 0 – normal mode 1 – NVG mode |
| 0x000C | W/R | auto\_manu | 0 – auto select video1 – manual select video |
| 0x0010 | W/R | osdCtrl | 0 – no OSD1 – overlay OSD |
| 0x0014 | W/R | video source | 0 – video source 01 -  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 signalbit 4 -- ddr read: 0 normal, 1 no read operationbit 5 -- ddr write: 0 normal, 1 no write operationbit 7-6 – display: 00 normal, 01 no signal, 10 error |

### 4.2.3 OSD Memory

OSD physical address: 0x70000000

OSD Resolution: 1920x1080

OSD format: RGBA32

### 4.2.4 iio sensor

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

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

### 4.2.5 Watchdog

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

*watchdog0 is controlled by OS.*

*Watchdog1 can be controlled by the user.*

# 5 Programming

## 5.1 A simple firmware

### 5.1.1 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;

}

### 5.1.2 Compile

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

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

### 5.1.3 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*