The program design of video capture based on SYSBIOS embedded system

Chen Longhu[1], Xia Xiaorong[1], Zhao Bo[1,2,a]

[1]The Third Research Institute of Ministry of Pubic Security, Shanghai, P.R. China

[2]Shanghai University, Shanghai, P.R. China

[a] email: 95130880@163.com

Keywords: TMS320DM642; Video capture program; Driver layer model; FVID model.

Abstract. The traditional video collection program should complete all the design work from the hardware layer to the driver layer and the application layer, which is a large work. It develops a video capture program based on driver model defined in DDK kit with API of general input and output module (GIO), and all the design methods are given in this paper. The program proposed in this paper improves the reusability and portability of the program and save the development time.

Introduction

TMS320DM642 is a multimedia processing chip with a strong processing performance and a high degree of integration produced by TI company, which is based on the C64x and adds a lot of peripheral and interface[1]. The main peripherals on-chip are three configurable video interfaces, VCXO interpolation control port (VIC), I2C bus module, etc.. SYSBIOS system launched by TI company greatly simplifies the development of DSP program. The traditional software development requires all layer programs to complete, but the way given in this paper can call the relevant frame module directly with SYSBIOS embedded operating system. It develops a video capture program based on driver model defined in DDK and introduces in detail about the development of register configuration, driver and application program.

Driver layer model

In this paper, video capture driver is based on standard device driver model defined in DDK shown in Figure 1. The driver is divided into class driver layer (class driver) and micro driver layer (mini-driver). Class driver layer is located between the application and micro drive, and provides 3 driver API interfaces of the flow input and output module (SIO), pipeline management module (PIP), general input and output module (GIO). The class driver can call and access to peripherals by the interface standard, and connect to micro driver through the PIO and DIO adapter, and realize mapping between API interface function and mini driver layer. Micro driver can realize specific peripheral initialization and all control operation with chip support library (CSL)[2], and call the command received from class driver to control underlying hardware.

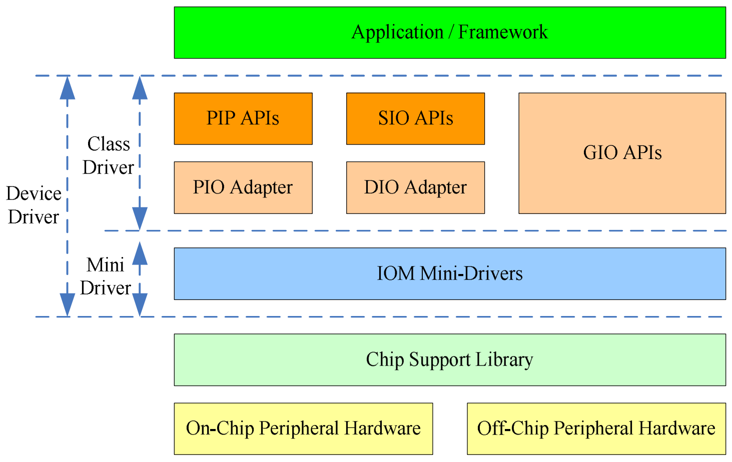


Figure 1. Driver layer model of SYSBIOS

The hardware design of video capture

DM642 controls the TVP5150 to capture video through the IIC bus. DM642 is the main device, and TVP5150 is a slave device, whose address will be selected by the voltage level of the VP0_D7, The software of capture will initialize IIC bus firstly, including unlock the I2C module, I2C clock Register settings, I2C mode register configuration. It sets the TVP5150 device address according to the circuit schematic diagram as shown in Figure 2, and initializes the registers in TVP5150 through the I2C reading and writing program, and then TVP5150 starts to work.

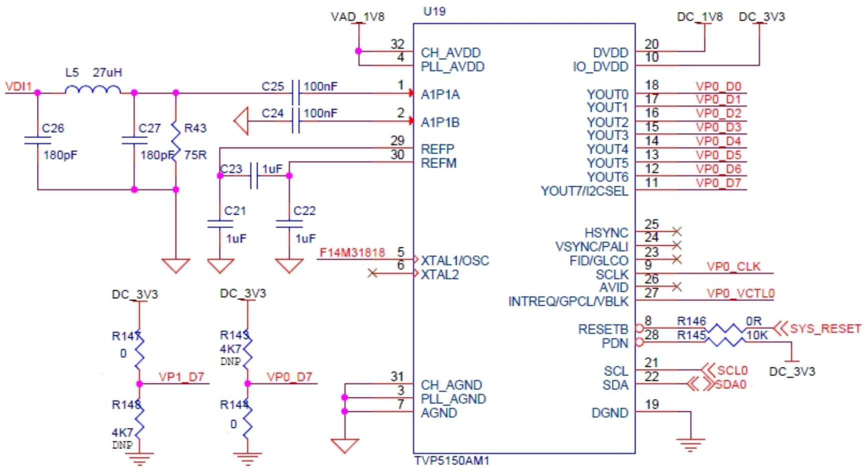


Figure 2. The hardware design of video capture

The software design of video capture

(1) Initialization settings

I2CSEL is the address selection bit of the TVP5150, whose voltage level will determine the device address, VP0 I2CSEL is low voltage seen from the hardware principle diagram as shown in Figure 3, so the device address of VP0 is 0xB8 according to Table 1. VP1 I2CSEL is high voltage, so the device address is 0xBA[3].

Table 1. Write address selection

I2CSEL	Write address
0	B8h
1	BAh

The I2C input clock is divided twice more inside the I2C module to produce to the module clock and the master clock[4]. The module clock determines the frequency at which the I2C module operates shown in Figure 3. A programmable prescaler in the I2C module divides down the I2C input clock to produce the module clock. To specify the divide-down value, initialize the IPSC field of the I2C prescaler register, I2CPSC. The resulting frequency is followed by formula (1).

$$\text{module clock frequency} = \text{I2C input clock frequency} / (\text{IPSC}+1) \qquad (1)$$

The input clock of IIC module is the CPU clock of DM642, that is 600MHz. I2C module clock is 7-12MHz, and it can select 8MHz. so IPSC=0x4Ah.

The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C-bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses ICCL value of I2CCLKL to divide down the low portion of the module clock signal and uses the ICCH value of I2CCLKH to divide down the high portion of the module clock signal[5]. The resulting frequency is followed by formula (2).

$$\text{master clock frequency} = \text{module clock frequency} / ((\text{ICCL}+d)+(\text{ICCH}+d)) \qquad (2)$$

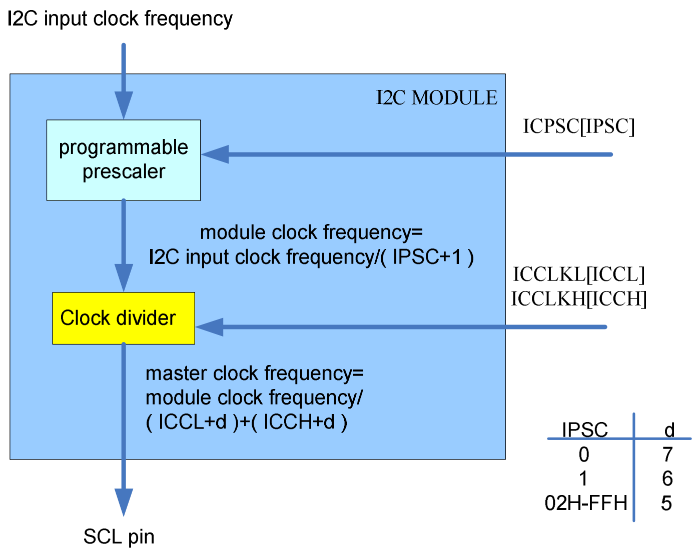


Figure 3. The clock setting of IIC module

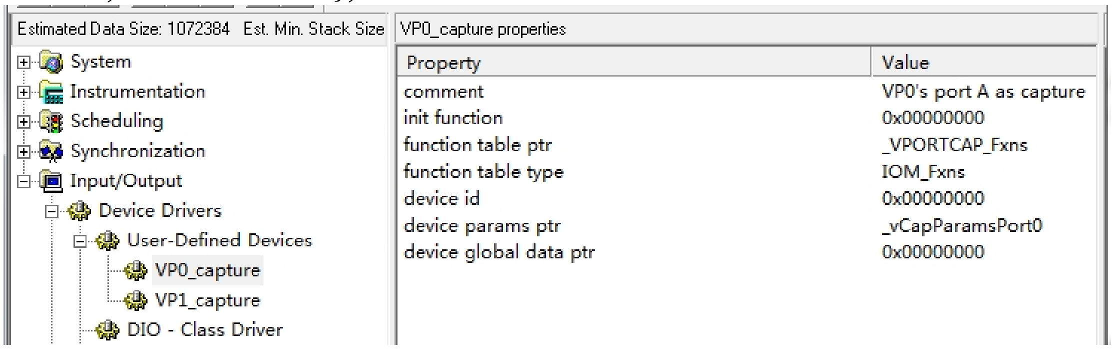
The IIC modules of DM642 and TVP5150 both support the fast mode of 200kbps, so d=5 according to the value of IPSC, and you can set the ICCL=15 and ICCH=15 to output 200kbps clock on SCL pin.

The I2C mode register (I2CMDR) contains the control bits of the I2C module. It will set 0x4620 for writing and 0x4420 for reading, which contains 8 bits IIC data format, 7-bit address mode, IIC enable setting, Nonrepeat mode of data exchanged, master/slave mode and so on.

(2) Register micro driver

It should add device driver in project configuration file. It is the first step to add VP0_capture and VP1_capture in the path Input/Output->Device Drivers->User-Defined Devices in sysbios configuration file and set its properties shown in Figure 4, of which the device ID specifies which channel acquisition, and it will be set to 0 and 1 for the VP0 and VP1 channels. function table PTR is

driver function table name for instance, named VPORTCAP_Fxns, and it can be defined as follow in VPORTCAP_Fxns function table.
IOM_Fxns VPORTCAP_Fxns =
{ mdBindDev, (IOM_TmdUnBindDev) IOM_mdNotImpl, mdControlChan, mdCreateChan, mdDeleteChan, mdSubmitChan };


Estimated Data Size: 1072384 Est. Min. Stack Size

- System
- Instrumentation
- Scheduling
- Synchronization
- Input/Output
  - Device Drivers
    - User-Defined Devices
      - VP0_capture
      - VP1_capture
    - DIO - Class Driver

VP0_capture properties

Property	Value
comment	VP0's port A as capture
init function	0x00000000
function table ptr	_VPORTCAP_Fxns
function table type	IOM_Fxns
device id	0x00000000
device params ptr	_vCapParamsPort0
device global data ptr	0x00000000

Figure 4. The configure of micro driver

After definition of the drive function table, it can call the class driver function to call the corresponding driver based on the device instance channel, and then realize the peripheral equipment operation. The correspondence functions of FVID, GIO and micro device driver are shown in the table 2[6-8] .

Table 2. The correspondence functions of FVID, GIO and micro device driver

FVID	GIO	IOM driver
------	------	mdBindDev
FVID_create	GIO _create	mdCreateChan
FVID_control	GIO _control	mdControlChan
FVID_alloc	GIO _submit	mdSubmitChan
FVID_exchange	GIO _submit	mdSubmitChan
FVID_free	GIO _submit	mdSubmitChan
FVID_delete	GIO _delete	mdDeleteChan

(3) Capture program design

TI, the company produces DM642, provides a complete set of DDK software development kit for sysbios developers. According to the standard driver model defined by DDK, video capture system calls the standard interface functions of FVID model, and maps these to the IOM micro drive functions, which controls specific peripherals based on the chip support library (CSL). So the development of the application program is to call the FVID model API function in class driver layer to complete the video capture. These functions are as follows: FVID_create, FVID_control, FVID_alloc, FVID_exchange, FVID_free. The program flow chart is shown in Figure 5.

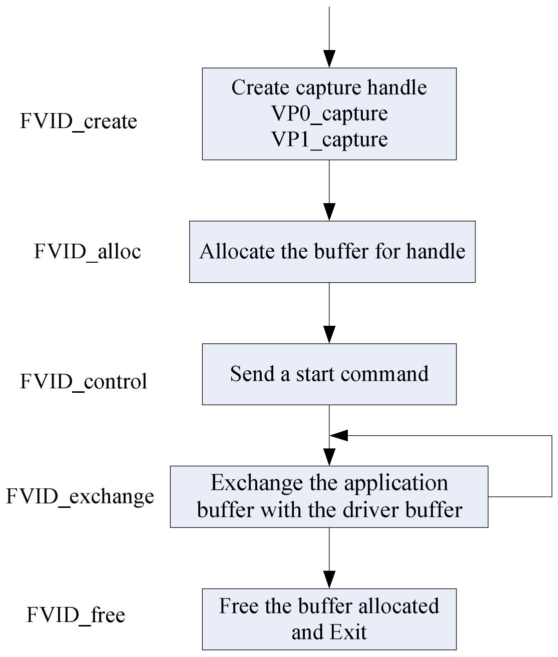

FVID_create
Create capture handle
VP0_capture
VP1_capture
FVID_alloc
Allocate the buffer for handle
FVID_control
Send a start command
FVID_exchange
Exchange the application buffer with the driver buffer
FVID_free
Free the buffer allocated and Exit

Figure 5. Program flow chart

Conclusions

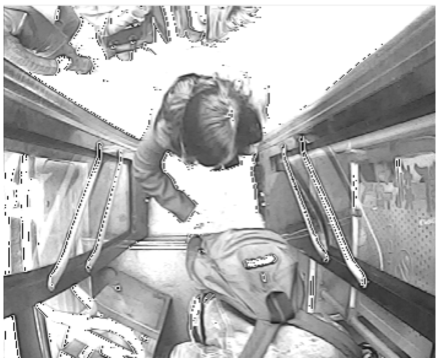


Figure 6. The captured image

The image captured by the way proposed in this paper is shown in Figure 6, which realizes a kind of the video capture function based on SYSBIOS system. It develops a program of collecting original video based on the DM642 hardware platform, and introduces the detailed design steps. In this method, the program has a strong adaptability, and it can modify the parameters to achieve the different needs of video acquisition. When the device is changed, the application needs only minor modifications, so it improves the reusability and portability of the program, and greatly saves development time, and it is a stable bottom support for all kinds of video processing algorithm.

Acknowledgments

This work was supported in part by the Program of Science and Technology Commission of Shanghai Municipality (No. 15530701300, 15XD15202000); National Natural Science Foundation of China (No. 61403084).

References

[1] TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor Data Manual. Texas Instruments. 2004:17-18.
[2] TMS320C6000 Chip Support Library API Reference Guide Literature Number SPRU401I. Texas Instruments. 2004:1-2.
[3] TVP5150A ultralow power NTSC/PAL/SECAM video decoder with robust sync detector data manual. Texas Instruments. 2004:2-9.
[4] Anonymous. Interface transports bi-directional I2C in one cable[J]. Electronics Weekly, 2008(2344), pp.20.
[5] TMS320C6000 DSP Inter-Integrated Circuit (I2C) Module Reference Guide. Texas Instruments. 2002: 13-14.
[6] TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide. Texas Instruments. 2003:2-95.
[7] DSP/BIOS Device Driver Developer's Guide. Texas Instruments. 2002:3-1.
[8] Parikh N, Itti L, Weiland J, Saliency-based image processing for retinal prostheses[J]. Journal of Neural Engineering, 2010, Vol.7 (1), pp.60.