The Architecture and Applications of 12C Device Driver in Embedded Systems

S. Suresh Kumar[1]; Dr.C. V. Narasimhulu[2] & T Rama Krishna[3]

[1]Research Scholar, Geethanjali College Of Engineering And Technology, Cheeryal (Vill), Keesara (Mandal), RrDist,A.P, India
Email: sanam.sureshkumar93@gmail.com

[2]Professor &Hod:,Geethanjali College Of Engineering And Technology, Cheeryal (Vill), Keesara (Mandal), RrDist,A.P, India

[3]Professor, Geethanjali College Of Engineering And Technology, Cheeryal (Vill), Keesara (Mandal), RrDist,A.P, India

Abstract-

'*The device-core-bus three-layer architecture of i2c is the main reference framework to develop i2c device driver efficiently in Embedded system. With a structured viewpoint we have analyzed the driver layers, data structures, driving procedures, and especially analyzed the two developing approaches for driving procedures in device layer.*

Keywords-Embedded, I2c, Driver, Architecture

I. INTRODUCTION

I2C bus, a two-wire bus with a compact size and simple timing, is widely used in embedded field. But in the operating system, i2c device driver architecture becomes very complex in order to support multi-device, multi-tasking. The author, therefore, with a structured viewpoint, expounds the architecture and application of i2c under embedded Systems in detail.

This paper is based on the embedded Systems of 2.6.30 Kernel and the i2c kernel driver developed by Greg K, Hartman, Simon G. Vogl.

II. I2C-DRIVER'S LAYER STRUCTURE

A. Constitution of driver layer

kernel

According to the call layers, Device drivers can be divided the following three levels: i2c device layer driver, i2c core layer driver and i2c bus (adapter) layer drive. These three parts complete the framework for i2c driver with strong applicability. i2c bus driver and the device driver are linked by the kernel driver. And i2c source file layer mechanism corresponds to the driver layers structure.[l]

B. 12C file structure

Kernel source organization: the sources related to i2C are stored in i2c folder of kernel. There are i2c core.c, i2c _ dev.c and some folders such as busses, chips and algorithm.

The functions of the core layer are achieved by the i2c _ core.c. The device layer is rather special, and there are two equivalent ways. First, by i2c _dev.c, adapter file interface is realized (i2c_dev method), in other words, at the application layer calling the master node (the equivalent of the adapter) created by the kernel file i2c_dev.C, interface functions, such as read, ioctl, etc.. can have access to visit the device. This method is equivalent to programming device driver process in the application layer. 2. Through c file in the chips folder, device driver file interface is realized (driver method), that is, writing performance functions such as xxx command o (xxx is the name of a custom from the device) illc file in the chips folder can access device process and call this function at the application layer. The two methods are equivalent. [3]

ARM chips are generally integrated with i2c adapter hardware, and the C file under busses realizes the functions of the bus adapter by calling the driver algorithm file functions under algorithm, and busses and algorithm constitute the bus layer.

III. I2C DATA STRUCTURES AND PERFORMANCE FUNCTIONS

I2C can be divided into performance functions and data structures according to the function completed by driver code.

I2C performance functions, such as driver registration, attachment, attachment solution, cancellation, transfer, etc., reflect on the general methods of operating i2c hardware, data, timing, etc. while data structure represent example objects of the hardware, data, timing algorithm, and the device must be instantiated as corresponding data structure in order to be used by i2c performance function, and through its interaction with the data structure, it completes the driver process. Device layer, core layer and bus layer all include corresponding performance functions and data structures.

A. 12C driver data structure

The structure in i2c, as the abstract examples of the specific adapter, device, drivers, signal algorithm, a message, is the carriers of specific device, driving method, and data, which play an important role in the i2c framework. The main data structures in i2c driver include i2c_ dev, i2c _client, i2c_adapter, i2c_msg, etc., and there is also nestification between some structures. The correspondences between the main structures and objects are as follows:

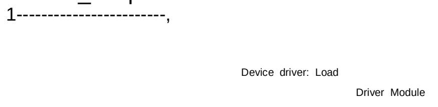
B. 12c Performance Functions

12c bus layer provides the core layer with the processing function of bottom signal timing. i2c core layer, as the intermediate layer, provides the specific device and the general driver function unrelated to adapter for calling by the device and the bus layer. The device layer, through calling the driver function of the core layer, calls the bus adapter function and provides the application layer with performance function of specific device.

IV. THE LOADING OF 12C DRIVER

A. The loading of bus (adapter) driver

The bus driver, when separately loaded as a module driver, first needs to be registered a structure, platform_driver, including the specific adapter's probeO function, removeO function pointer, etc., which need to be assigned. When this module is loaded, the probeO function of the specific adapter is called to initialize the adapter hardware, including the initialization of hardware to enable and applying I / 0 address, interrupt numbers, and so on. When the initialization is completed correctly, the bus driver probeO function calls i2c_add_numbered_adapterO function in the i2c kernel to add the current bus driver to the linked list of kernel i2c _adapter.



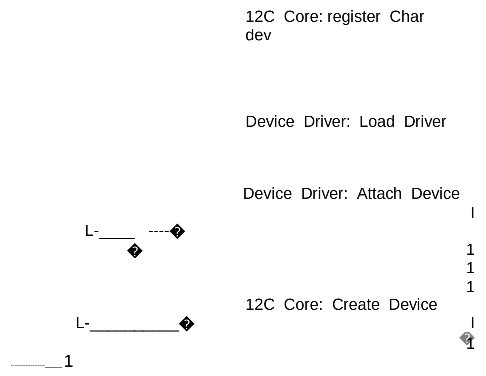

12C Core: register Char dev
Device Driver: Load Driver
Device Driver: Attach Device
12C Core: Create Device

Figure 2. driver loading process

B. The loading of the device layer driver

The device layer driver must be attached to corresponding adapter to call the bus drive function in the bottom when it is loaded. According to the driver way, for the attachment of the device drivers there are two methods: the adapter file interface (i2c_dev.c) and device driver file interface (Driver). [4]

The first method is to use the adapter file interface (i2 c_dev. c). It attaches master device data structure i2c_dev to the physical adapter, and operates slave unit through operating the master node (the equivalent of adapter) for the issue of timing signals on the bus. i2c_dev structure uses device object structure to organize the devices on the same bus into chain table, and add it to the chain table of global devices. i2c dev.c maintains a structure i2c driver, which is a supportive data structure instead of corresponding to a specific physical device, including two member functions attach_adapter and detach_adapter, whose pointers respectively points to two private member functions i2cdev_attach_adapter and i2cdev_detach_adapter. When i2c dev device driver is loaded, it first uses register_ chrdevO to register a character device in the init function, and through i2c add driverO calls i2cdev attach adapterO function, in whICh it first calls geUreej2c_dev() function to create a i2c dev data structure, and gets the adap members pointer to pomt the current adapter structure; second, i2c _dev structure needs to be added to the linked list in the kernel to maintain i2c dev; thirdly, it is supposed to call device_createO function to register device object in the kernel; finally, you should use device create fileO function to create the primary device file to complete the attachment of the master structure (i2c_dev) on/to the bus adapter (i2c _adapter) .

The second method is to use the device driver file interface. In this way each slave device corresponds to a client, which in turn, needs to attach to the physical adapter. When the device layer driver is loaded, i2c_add_driverO function will be called in the init function, and the parameter of the function is the structure pointer pointing to xxx_driver (xxx is slave unit name). xxx_driver structure contains the function pointers such as probeO, detectO, removeO, and so on. First, through the probeO function, it completes hardware detection, initiating the implement of detectO function, in which the i2c client structure of slave unit will be created and initialized� and then calls the i2c_attach_ c1ientO function in i2c core layer to add i2c_client structure to a static client chain table in the kernel, and finally it is supposed to initialize the device hardware, and complete the attachment of the slave unit structure (i2c_client) on the bus adapter (i2c _adapter).

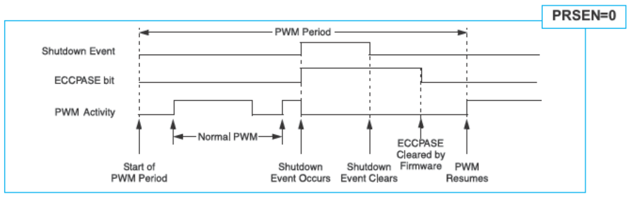
V. THE CALLING OF 12c DRIVER

A. The calling method of adapter file interface

The adapter interface file is i2c_dev.c, which can generate a primary device node of i2c-x(x is the serial

```
struei2.,,_msg
        {
          unsigned      short
          addr;     unsigned
          shon          flags
          unsigned  shon  en;
          unsigned char "buf;
        };
```

According to the i2c agreement, the number of i2c messages is related to the start signal, with a start signal corresponding to a message. Several messages are usually needed for a complicated i2c device to fmish a complete access, so the nmsgs should be more than 1. A pointer variable directing/pointing the address of i2c _msg is defmed in i2c_rdwr_ioctl_data. Thus, the structure of i2c_msg (see Fig. 3) should be initialized. In this structure, the device address needs to be fIrst initialized, then the read-write symbols of equipment operation, the length of the byte of the message - len, and the initial address of the buffer zone of message byte - buf pointer. Last, another member of the i2c _rdwr_ioctl_ data structure should be initialized: integer nmsgs variable. For example, The selected read time sequence of eeprom chip 24C02 (see Fig. 4) needs two start signals, so nmsgs should be equal to 2.



number in the kernel chain table of the present adapter) when the kernel has been loaded. The slave unit is operated indirectly through the manipulation of the primary device (adapter). To begin with, the application layer needs to use open function to open this node, transmit a reading and writing authority at the same time, and then invokes/calls ioctl function to deliver the timeout and retry times to the

core of i2c, next needs to defme a data structure of i2c rdwr-

ioctl data and makes it instantiated. An i2c-msg message structure pointer of msgs is nested in this data structure, which is pointed at an arrayof7 bytes. It also defines a variable for the number of the start signal-nmsgs. [5]

After the necessary i2c message data are ready, the equipment can be operated. The main operating functions are i2c_read, i2c _write, ioctl, etc. As for the two functions of i2c _read and i2c _write, the functions of i2c _master_recv

and i2c- master- send in the core of i2c need to be called to construct a i2c message and trigger to transmit in the way of i2c _masterJfer of the algorithm of corresponding adapter in the Bus Driver. But this method is only limited to the operation of one message, while ioctl method can be applied to the operation of many messages. Because it supports a method of Rep start and sends out start signals on the bus repeatedly (see Figure 5). Therefore, ioctl method has greater adaptability and should be the fIrst choice in the general case. Certainly, the bus transfer functions (i2c_masterJfer) that ioctl and the read and write functions call are the same.

B. The calling method of the device driver file interface

The loading procedure of the device driver fIle is similar to the adapter fIle interface. It loads specifIc device driver via i2c_add_driver () function in the process of init and attaches it to the corresponding bus driver. Driver can support the data structure of many slave units «i2c_c1ient), but they need to be attached to the present physical adapter (i2c_adapter). The timing sequence that has been organized by calling the device layer via the xxx_command function in the application program accomplish the i2c operation process of a specifIc device.

VI. CONCLUSION

Practice has shown that the device-core-bus three-layer architecture of i2c in Embedded Systems can fully satisi)r the requirements of more equipment, muItitask in embedded environment. But in some cases, it is necessary to simplii)rthe driver levels so as to simplii)r the development process. If the bus driver layer is only retained, all the driver task will be fmished on this floor. In short, it is supposed to choose suitable framework and the corresponding developing method according to the specifIc application in order to improve or enhance the efficiency of development.

REFERENCES

[I] Simon G.Vog i2c-core.c - a device driver for the i2c-bus interface. GNU,1999

[2] Greg Kroah-Hartman. i2c-dev.c - i2c-bus driver, char device interface. GNU,2003

[3] JonathanCorbet, Alessandro Rubini, Greg Kroah-Hartman. Device Drivers,3rd Edition., O'Reilly, 2005.12

[4] Jun Li, Development Solutions to Embedded Linux Device Driver, Beijing: Posts & Telecom Press, 2008.10

[5] Tianze Sun, Wenju Yuan, Haifeng Zhang, A Guide to Embedded Design and Linux Driver Development - Based on ARM9 Processor, Beijing: Electronic Industry Press,2005.2