ICAT 2002
December 4-6, Tokyo, JAPAN

The Virtual Debugging System for Embedded Software Development

Yi PAN, Norihiro ABE, Kazuaki TANAKA

Kyushu Institute of Technology,
689-4, Kawazu, Iizuka-shi, Fukuoka-ken, 820-8502, JAPAN
panyi@sein.mse.kyutech.ac.jp

Abstract:

The development cycle of an embedded system should be shortened now. However, in order to perform rapid product development, there are various problems to be solved. Especially, in development of the embedded software which controls embedded machines, as debugging the embedded software cannot be developed until the system (hardware and mechanism) is completed, loss of time in development arises and the progress of the whole product development will be inhibited greatly. In this research, we build a virtual machine in virtual space, and create an interface between an embedded software and virtual machine. It is considered that embedding the software embedded in the real system into a virtual machine, and working the virtual machine.

Key Words: embedded system, virtual reality, embedded software,

1. Introduction

Virtual reality is the general term of the artificial world which is made by computer. It is the system which generates the temporary information world working on human feeling and presents the actual image artificially. Moreover, using this system, various information which one wants to know can be experienced completely as if it mere real. Virtual reality as a method of raising presence is considered in many applicable fields, such as a design, a simulation or a game, and education.

A computer system which is embedded in various kinds of apparatus and performs the control is called an embedded system. This embedded system will be applied to all the household electric appliances, such as not only industrial apparatus but television, a refrigerator, a microwave oven, a cellular phone, etc. The development cycle of an embedded system should be shortened now. However, in order to perform rapid product development, there are various problems to be solved. Especially, in development of the embedded software which controls embedded machines, as debugging the embedded software cannot be developed until the system (hardware and mechanism) is completed, loss of time in development arises and the progress of the whole product development will be inhibited greatly. Moreover, if development takes time, when software is completed, the problem may occur that hardware has already shifted to the architecture of the next generation. In this research, we build a virtual machine in virtual space, and create an interface between an embedded software and a virtual machine.

As an example of embedded apparatus, we take up the card printer used to print a credit card, a student identification card, etc. A virtual machine is constructed, the control function (virtual driver) which operates each part is defined, and an embedded software is performed on the embedded MPU emulator which is prepared beforehand. And the request from embedded software can be detected because a virtual driver accesses the memory and register on an emulator, and the behavior of the machine according to the request is rendered in virtual space. Using such a technique, the simulation of the printing process of a card printer is carried out, and it is tried to debug an embedded software.

2. System configuration

OpenGL

In this research, in order to draw (construct) virtual reality space more simply, 3 dimension graphics library "Open GL" was used.

LightWave3D

LightWave3D is a modeling tool for creating a 3-dimensional object, LigtWave3D modeler application adds sub division surface modeling technology to the polygon modeling which edits the polygon and the point (polygonal vertex). Consequently the form creation with high flexibility is possible. This research creates 3D model of a virtual machine using LightWave3D.

MFC

Microsoft Foundation Class Library is a class library for building the Windows application framework which can be used by Visual C++, and is used for development of Windows application and a component. Using MFC, GUI of application can be created easily.

3. Introduction of a simple model

As an example of a virtual machine, the card printer used to print a credit card, a student identification card, etc. is taken up, and the detailed condition is described. Unlike the real machinery, since a virtual machine does not need to model the frame which supports a mechanism, what is necessary is just to mount the mechanism parts which actually work, the burden of modeling work is comparatively light.

3.1 The flow of a card printer of operation

The simple model used for this research is a thermal transfer type card printer, it forces the heated printing head on the ink ribbon which applies solid ink thinly, melts ink, and the ink is transferred on paper.(Fig.1) Ink consists of 5 colors: Cyan, Magenta, Yellow, Black, and OP (printed surface is protected in transparent meltdown ink), and each color is printed on a card during 5 times. Moreover, in case each color is printed, ribbon rolling-up processing is performed first so that the boundary line of the color of an ink ribbon may come to the position of a thermal head (it is called ribbon search).

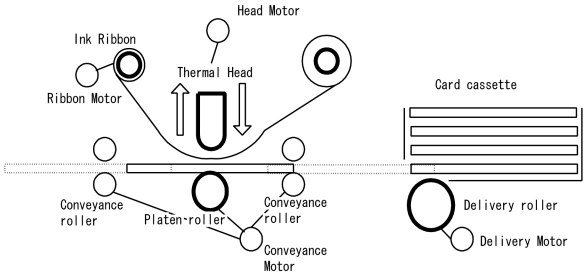

Fig.1 The sketch of a mechanism model

3.2 the model of mechanism

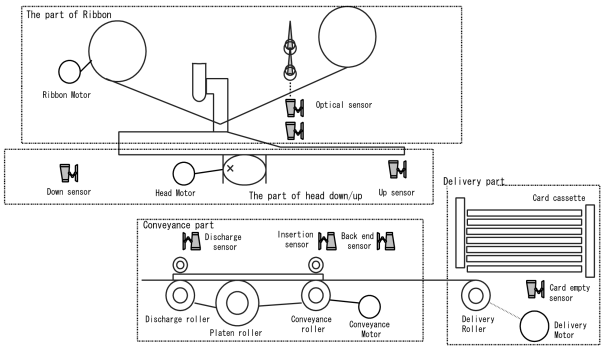

Fig.2 The model of mechanism

The sub assembly name of a mechanism model and the function are shown in Fig.2.

Delivery part: it sends out a card from a card cassette.

Conveyance part: it sends the card to a printing position; and returns a card to a printing position; send a card during printing; discharge a printed card.

Ribbon part: it rolls round the ribbon; detect the color of a ribbon.

Head up/down part: it makes a head go up and down.

3.3 The model of data flow

The sub assembly name and function of a data flow model are shown below:(Fig.3)

Image buffer: Large scale RAM which stores image data (SDRAM is used);

Reception DMA: The image data sent by the host is transmitted to the Image buffer. CPU (embedded software) specifies the head address of the image to be printed and the number of transmission bytes.

Printing DMA: The specific range of a image buffer (data for one line) is transmitted to a thermal head. CPU (embedded software) specifies the head address of the image to be printed, and the dot range of the thermal head conducted (a start position and an end position +1).

Thermal head: it prints according to the data of Printing DMA.

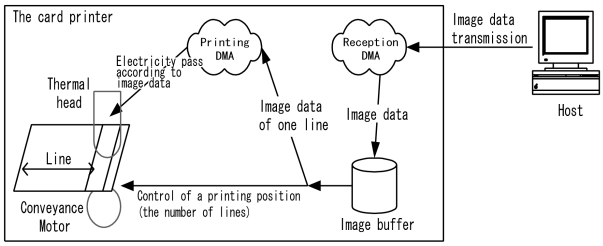

Fig.3 The model of data flow

4. Virtual machine creation

Although OpenGL has all primitives necessary, however, since creation of a complicated object requires time and effort using OpenGL, LightWave7.0 that is the exclusive tool of 3D model creation was used, and 3D model was created for every part of the system to be modeled. Moreover, since LightWave7.0 is equipped with the library for accessing an object from an external program, the created model can be used without adding a hand as it is. (Fig.4)

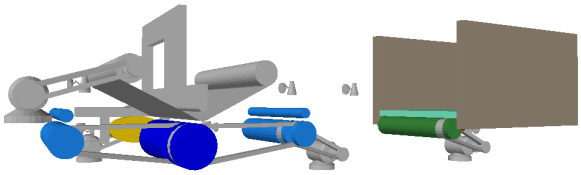

Fig.4 The Virtual Machine model of a card printer

5. Input and output specification of Composition apparatus with our system:

We assume that the system is memory mapped I/O that is input and output port and the memory are arranged in the same address space. In order to distinguish from an ordinary memory merely, the address currently assigned to input and output is called I/O register.

5.1 The example of input and output specification of Composition apparatus

The card printer is consisting of many composition apparatus, such as a motor and a sensor, the I/O specification of a stepping motor used for a conveyance motor must be described in detail.

Table 1. The control register of a stepping motor

Bit :	7	6	5	4	3	2	1	0
	—	—	—	—	—	Enb	Cwb	Clk
R/W :	—	—	—	—	—	R/W	R/W	R/W

A stepping motor is controlled by the register as shown above. (Table 1.)

Bit7~3: In reading mode, 0 is always read and rewriting is invalid.

Bit 2: specifies the excitation state of a stepping motor.

Bit 1: specifies the drive direction of a stepping motor.

Bit 0: the phase clock input of a stepping motor, when the clock changes from 0 to 1, a motor is go forward one step

5.2 Creation of the virtual driver class

Referring to the input and output specification of composition apparatus, to permit embedded software to communicate with a stepping motor in a virtual machine, the stepping motor class is defined to read/write the corresponding register. (Fig.5) Therefore, a virtual machine detects the operation request from the embedded software, and a virtual machine will be controlled by the embedded software.

The created STPMotor class

```
class STPMotor{
prevate:
        unsigned Enb ;    // the excitation state
        unsigned Cwb ;    // the drive direction
        unsigned Clk ;    // the phase clock
public :
void  SetEnb(unsigned en);// specifies the excitation
                          //state
void  SetCwb(unsigned cw);// specifies the drive
                          //direction
void  SetClk(unsigned cl);// specifies the phase clock
unsigned GetEnb();        //acquires the present
                          //excitation state
unsigned GetCwb();   // acquires the present drive
                     //direction
void MotorTurn();    //Motor rotation
}
```

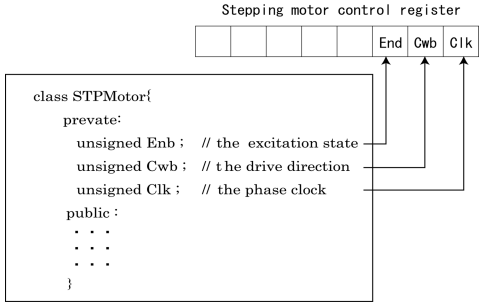

Fig.5 The image of a Virtual Driver

The control class corresponding to the control register was created for each operation module, such as DC motor and a sensor, using the same method as the stepping motor control class.

5.3 The interface between a software and a virtual machine

The card printer consisting of many apparatus, such as a motor and a sensor, and to make a virtual machine carry out the same operation as the real system, the virtual model of these composition apparatus should operate in the same way as the real one. Then, to make the virtual model of composition apparatus operate on virtual space, the control function (virtual driver) for the virtual model must be made. Moreover, it is necessary to create the behavior function for realizing the motion of the mechanism interlocked with other parts on the virtual space, such as motors and sensors. In the real system, according to software, CPU communicates with the register, generates various input and output signals, and communicates with composition apparatus through an interface. In this research, the emulator of an embedded microcomputer is created and the embedded software is performed on this emulator. The embedded software communicates with the register of an emulator, further, the request of operation from the embedded software is detected, the virtual driver of composition apparatus accessing a register operates a virtual machine, and the behavior a virtual machine is displayed on virtual space. Moreover, the present state of a virtual machine written into the register corresponding to a particular emulator through a sensor

etc., and information required is feedbacked to embedded software. Using this technique, the embedded software can control a virtual machine as well as the real system by creating the interface between a virtual machine and embedded software. (Fig6.)

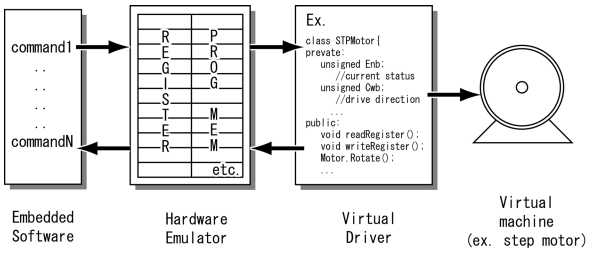

Fig.6 The interface between a software and the virtual machine

6. The emulation for embedded MPU

This time, we created experimentally the emulator of the small one tip microcomputer (PIC16F84) for embedded system made of Microchip Company. (Fig.7) About the memory and the I/O of architecture of a tip, the array variable of the same capacity is used as the real one. About a processor basic command, the command function that carries out the same function is created as the real one. Instead of rewriting the contents of a memory and I/O by executing a processor command, a memory array variable is rewritten by performing the command function equivalent to a processor command. Using the method described above, MPU is emulated. Fig.8 shows a class of embedded MPU. In the class, the array of memory and the processor command function are defined according to the architecture of the real MPU. Fig.9 shows a sample of the processor command function. INCF is a processor command which increments a value of specified address.

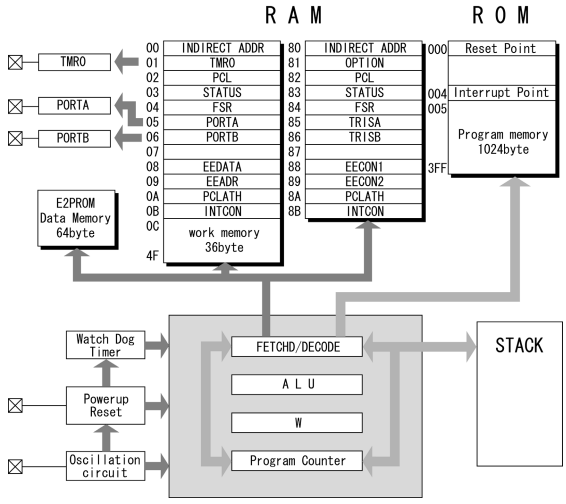

Fig.7 The architecture of PIC16F84

In this research, we create the general-purpose drive apparatus (i.e. stepping motors), and the program that controls the virtual model. (Fig.10&Fig.11) When debugging a program that controls the stepping motor using this debugging system, the same behavior as the real one was obtained when the same program as that applied to the real system are applied to the virtual system.

```
class Cmpu
{
public:
        int stack[8];
        //Prosesser Command Function
        void XORLW(int k);
        void SUBLW(int k);
        void SLEEP();
        void RETURN();
        ・  ・・
        ・  ・・
        void CLRW();
        void CLRF(int f);
        void ANDWF(int f,int d);
        void ADDWF(int f,int d);
        int EPROM[64];    //EPROM Memory
        int Wreg;
        int ram[2][80];    //Data Memory
        Cmpu();
        virtual ~Cmpu();
};
```

Fig.8 Mpu Class

```
void Cmpu::INCF(int f,int d)
{
        if(d==0)
                Wreg=ram[0][f]+1;
        else
                ram[0][f]=ram[0][f]+1;
        ram[0][2]++;

}

void Cmpu::INCFSZ(int f,int d)
{
        if(d==0)
                Wreg=ram[0][f]+1;
        else
                ram[0][f]=ram[0][f]+1;
        ram[0][2]++;
        if(ram[0][f]==0)
                ram[0][2]++;
}
```

Fig.9 The sample of processor command function

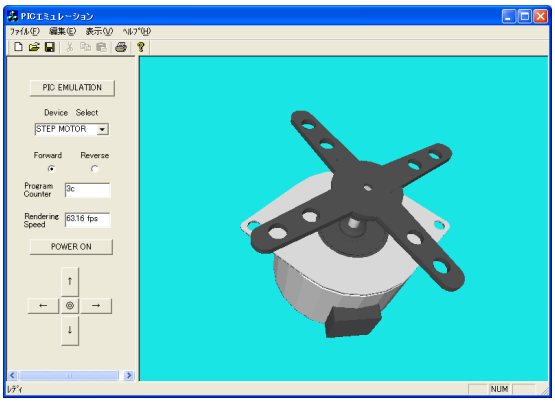

Fig.10 An example of debugging a program which controls the stepping motor

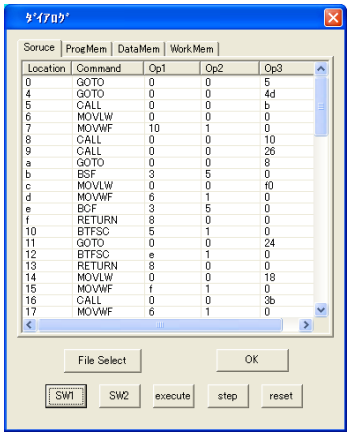

Fig.11 PIC Emulator

7.The application creation by Visual C++ and the execution result

In order to offer a user friendly debugging environment, a virtual debugging system application was created. Using this application, we can not only see the virtual machine, but also carry out the emulation of the hardware. We also offer the interface for grasping the status of debugging in real time, changing the viewpoint arbitrarily, and controlling the advance of debugging using this application. (Fig.12)

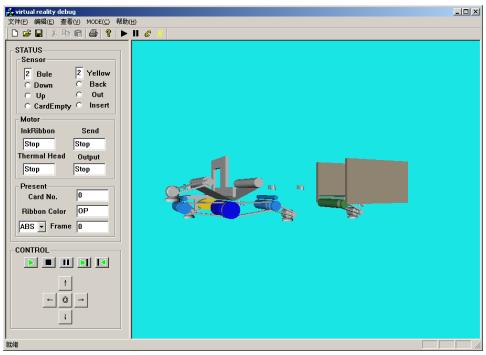

Fig.12 The virtual debugging system application

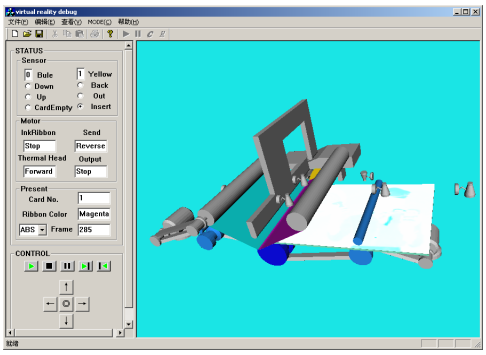

Fig.13 The situation under debugging (Cyan)

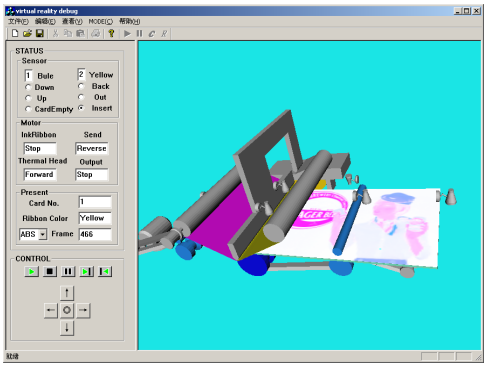

Fig.14 The situation under debugging (Cyan, Magenta)

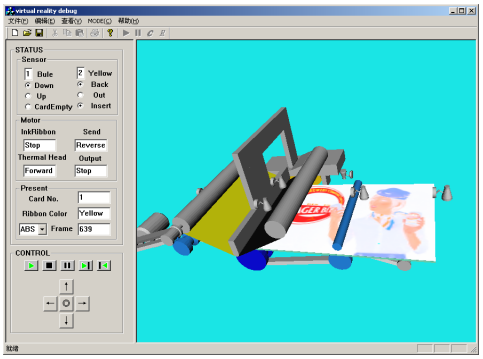

Fig.15 The situation under debugging (Cyan, Magenta, Yellow)

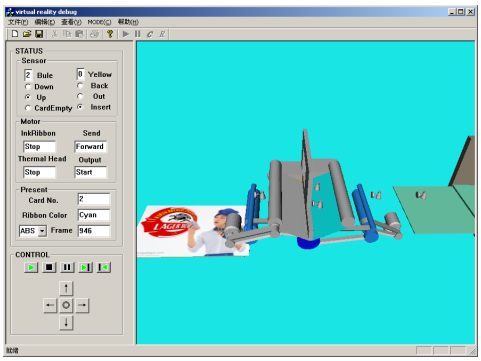

Fig.16 One card printing end

8.Consideration

1. When debugging was performed with the application created this time, execution speed (for animation) changed from a few frames to ten frames per second depending on the spec of computer. The number of polygons of a 3D model and the low rendering performance of a computer make it difficult for a virtual machine to work in the same speed as the real one. It is really difficult to work a virtual machine as fast as the real one. Considering the debugging process of a virtual machine, it is desirable to permit a programmer to see the state of the machine working at arbitrary speed.

2.To make the system construction easy, the friction coefficient of a machine part, the resistance under operation, etc. are not taken into consideration. In order to realize the simulation of more realistic printing system, it is necessary to consider these influence elements. This should be tackled as a future subject.

3.Whenever a model changes in virtual machine creation, we have to perform re-creation. Considering the efficiency, since the Standards Part is actually used in many cases in a machine design, constructing such a 3D model database of Standards Part will improve such problems to some extent.

4. When a part model of a virtual machine is replaced with a new one, the current system forces a developer to define a new virtual driver. In order to solve this, it is necessary to take in the concept of the design pattern and to consider that the created virtual driver (program) is reusable.

9.Conclusion

In this research, the virtual machine was built and the virtual driver that controls a virtual machine was created. As the result, it is known that embedded software debugging using the virtual machine is possible. Not only debugging of embedded software but debugging the mechanism composition of a virtual machine can also be preformed by visualizing debugging process. Moreover, the embedded software that is completely debugged can be shifted to the real system smoothly without any change. However, many problems are still remaining including the augmentation of flexibility and the construction of virtual space of high reality. Aside from expanding the support range of the MPU emulation, we want to make a system which helps as debug the complicated embedded system consisting of FPGA and a system LSI, etc.

Reference

1. “OpenGL Programming Guide”: Jackie Neider, Tom Davis, Mason Woo, Addison-Wesley Publishers Japan, Addison Wesley Professional, 1999.
2. Kouichi NAKAMOTO, Hiroaki TAKADA, Kiitirou TAMARU: “The present condition and the trend of embedded system technology”, Vol.38, No.10-004, 1997.
3. Michael Barr: “Programming Embedded Systems in C and C++”, Vol.1, 1999.
4. Data Sheet of PIC16F84 etc., Microchip Technology Inc.
5. “PIC practical guidebook”, Tetsuya Gokan, Gijyutsu Hyouron Sya, 2000.
6. Transistor technology, “A guide to one tip microcomputer practice”, Vol.36, No.5, pp.163-240, 1999.