Sensors & Transducers
© 2014 by IFSA Publishing, S. L.
http://www.sensorsportal.com

The Design and Realization of Virtual Machine of Embedded Soft PLC Running System

[1] Qingzhao Zeng, [2] Suzhuo Wu, [3] Zheng Li
[1] Tianjin Keyvia Electric Co., Ltd, Tianjin, 300384, China
[2] Civil Aviation University of China, College of Aeronautical Automation, CAUC, Tianjin, 300300, China
[3] Tianjin Keyvia Electric Co., Ltd, Tianjin, 300384, China
[1] Tel.: 18322731573
[1] E-mail: wusuzhuo@163.com

Received: 4 August 2014 /Accepted: 30 October 2014 /Published: 30 November 2014

Abstract: Currently soft PLC has been the focus of study object for many countries. Soft PLC system consists of the developing system and running system. A Virtual Machine is an important part in running system even in the whole soft PLC system. It explains and performs intermediate code generated by the developing system and updates I/O status of PLC in order to complete its control function. This paper introduced the implementation scheme and execution process of the embedded soft PLC running system Virtual Machine, and mainly introduced its software implementation method, including the realization of the input sampling program, the realization of the instruction execution program and the realization of output refresh program. Besides, an operation code matching method was put forward in the instruction execution program design. Finally, the test takes PowerPC/P1010 (Freescale) as the hardware platform and Vxworks as the operating system, the system test result shows that accuracy, the real-time performance and reliability of Virtual Machine. *Copyright © 2014 IFSA Publishing, S. L.*

Keywords: Embedded soft PLC, Virtual machine, Running system, Instruction execution.

1. Introduction

Soft PLC, established on a certain operating system platform, realizes traditional PLC functions such as calculation, control, storage and programming using software, at the same time, completes field data acquisition and signal output through the I/O driver module and field bus devices [1]. Embedded system is a system which integrates the operating system and functional software into computer hardware system. Embedded system's software code size is small, its software modules can be cut and has better real-time responsiveness.

Embedded soft PLC [2] is a soft PLC running on embedded system. It has not only the characteristics of traditional PLC, but also has the advantages of soft PLC, such as low cost, easy to upgrade, flexible configuration, etc., and it can easily achieve universal PLC system which is independent with platform and hardware.

Virtual Machine, that is, an abstract and virtual machine is added between the machine and compiler. This Virtual Machine provides a common interface to compile programs on any platform available.. Compile programs only need to generate the code which Virtual Machine can understand, and then the VM codes are converted into machine codes for a specific system by the interpreter before execution. Interpreters on each platform are different, but the realization of the Virtual Machine is the same [3].

PLC Virtual Machine is a middle ware, which shields the direct contact between logical application programs and hardware. Logical application programs written by users are only associated with PLC Virtual Machine and hardware's mapping and management are finished by PLC Virtual Machine. A virtual machine is a very important part of the running system and the entire PLC system. The function of virtual machine is to explain and execute the PLC program downloaded from the development system, update I/O status of the PLC to complete control function of soft PLC.

The overall architecture of Soft PLC system is shown in Fig. 1.

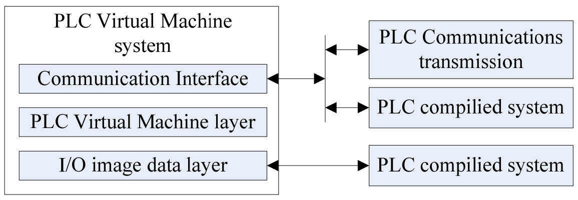

Fig. 1. The overall architecture of system.

2. Virtual Machine's Working Principle and Implementation Scheme

A Virtual Machine's working principle is similar with CPU, in essence, they both conduct a process of fetching, decoding and executing repeatedly. However, CPU is done by hardware, and Virtual Machine through pure software. CPU works in two ways: one is based on the register, this CPU executes instructions fast and easy to debug; the other is based on the stack, this CPU is more popular in embedded devices, because it better supports small program, takes up smaller memory and calls the function faster [4]. The two CPU have both advantages and disadvantages, but which are both reflected in the hardware. Virtual Machine designed by pure software in this article does not have this problem.

PLC Virtual Machine has three implementation scheme: interpreted program, intermediate program and compiled program [5].

Interpreted program, the PLC program compiled in Ladder Diagram (LD) or Instruction List (IL) languages is compiled into a series of intermediate form of instruction data. These instruction data is read into memory array one-time in the PLC Virtual Machine initialization, and then this memory array is scanned periodically, interpreted and executed. Interpreted program is shown in Fig. 2.

Intermediate program, the PLC program compiled in Ladder Diagram (LD) or Instruction List (IL) languages is also compiled into a series of intermediate form of instruction data, which is read into a C language header file when compiling source code of PLC Virtual Machine, and then it generates a new PLC Virtual Machine with the VM source code. Intermediate program is shown in Fig. 3.

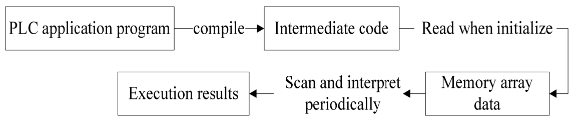

Fig. 2. The interpreted program implementation process.

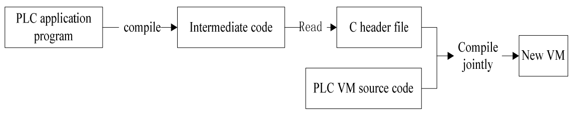

Fig. 3. The intermediate program implementation process.

Compiled program, the PLC program compiled in Ladder Diagram (LD) or Instruction List (IL) languages is compiled into C program, which is compiled into a new Virtual Machine with the Virtual Machine source code. Compiled program is shown in Fig. 4.

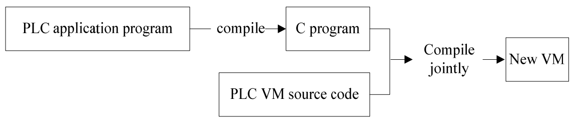

Fig. 4. The compiled program implementation process.

Interpreted program has intermediate form file and easy multilingual conversion. Compiling and debug do not need to be re-done when modifying the application program. So interpreted program was chosen as the implementation scheme of PLC Virtual Machine.

3. Virtual Machine Execution Process

The function of the Virtual Machine is to complete input processing, executing the PLC program compiled and generated by developing system, output processing, etc., and then achieves the final control. Virtual Machine will continuously scan cyclically, and each cycle finishes input sampling executing the program and output refresh until the shutdown. Virtual Machine execution process is shown in Fig. 5.

Virtual Machine program first initializes timers, interrupt system and so on. When the program starts scanning, first calls the function IoInput(), state of input terminals is collected and deposited in the memory. After that, calls VMExec() to explain and execute PLC program, then determine the state of output terminals according to the state of each input terminal and PLC program execution results. Afterwards, enters the output refresh stage, refresh of the output terminal state is completed by function IoOutput.

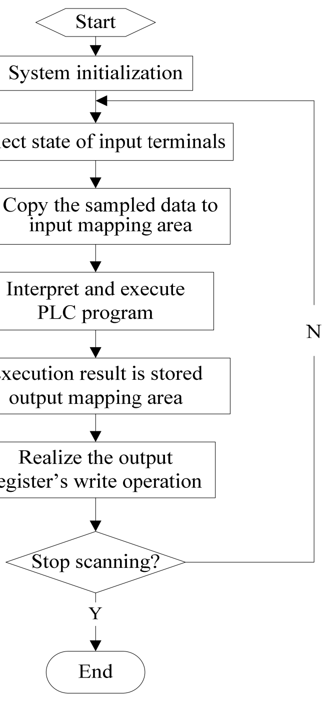
Start
System initialization
Collect state of input terminals
Copy the sampled data to input mapping area
Interpret and execute PLC program
Execution result is stored output mapping area
Realize the output register's write operation
Stop scanning?
N
Y
End

Fig. 5. Virtual Machine execution process.

4. Virtual Machine Program Design

4.1. The Realization of the Input Sampling Program

Virtual Machine input sampling program is performed at the beginning of each scan cycle. The program calls the function IoInPut (InBuffer) first, collects state of input terminals and deposits in the memory, then copies the input state data of the memory to the input image area for the use of PLC program execution. Key code of input sampling program is as follows.

```
IoInPut(InBuffer);// Sampling input
{
 for (1nt index=0; index (chanNum; :index++)
 {
 p=(IoVarDesc *)GetVarDesc(chanNum);// p
points to I / O variable descriptor
 p→ramAddr=InBuffer [index];// Copy the
sampled data to input mapping area
p++;
 }
}
```

4.2. The Realization of the Instruction Execution Program

The intermediate code of the system is executed by the function VMExec() of Virtual Machine. PLC Virtual Machine executes from the first instruction of the intermediate code file until finish End instruction. In this way, PLC Virtual Machine completes a program execution stage of scan cycle.

VMExec() is execution engine of a Virtual Machine, but the execution itself is not complicated. The Virtual Machine instructions can be roughly divided into basic instructions and function instructions two major categories. The Virtual Machine instruction pointer points to the current instruction operation code of a user's program. VMExec() will compare the current instruction operation codes with all the Virtual Machine instruction operation codes. If it finds a match, the Virtual Machine will "execute" the instructions this operation code represents; Otherwise, Virtual Machine will think it encounters a fatal error and will report the error. LogicFunc(Fcode) is the basic instruction execution function, SpecFuncFcode () is functional instruction execution function. Pseudo-code of VMExec () is as follows.

```
int VMExec()
{
 Unsigned char Fcode;// Instruction function code
 int Result;
 while (InstructionMem[Instructionlndex]!=0xff)
 {
 Fcode = InstructionMem[Instructionlndex];
 if((Fcode&0xf0) = = 0xf0)
 Result = LogicFunc(Fcode);// Basic instruction
execution
 else if (Fcode = = 0x01)
 Result = SpecFunc(Fcode);// Functional
instruction execution
 else
 return OK;
 }
 InstructionIndex++;
 return Result;
 }
```

Basic instructions is the logical instructions including LD (at the start of each logical line or logical block), OUT (output), AND (AND instructions), OR (OR instructions), NOT (NOT instruction), CNT (counter), TIME (timer) and so on. Function LogicFunc (Fcode) is used to achieve the above instructions execution in this system, the pseudo code is as follows.

```
void LogicFunc (unsigned char Fcode)
{
 switch (Fcode)
 {
 case I_LD:/// Push the previous result into stack,
take the operands out, and the Result and deposit the
new result into Result
......
 break;
 case I_AND:// Take the operands out, and the
Result and deposit the new result into Result
......
 break;
 case I_OR:// Take the operands out, or the Result
and deposit the new result into Result
......
 break;
 }
}
```

Functional instructions mainly finish complex calculation and control function including data shift instructions, data transfer instructions, data comparison instructions, the arithmetic operation instructions, numerical conversion instructions, logic operation instructions, program branch and jump instructions, subroutine called instructions and other system operation instructions. Used in this system function SpecFunc(Fcode) to achieve the above instruction execution, the pseudo code is as follows. Function SpecFunc(Fcode) is used to achieve the above instructions execution in this system, the pseudo code is as follows.

```
void SpecFunc (unsigned char Fcode)
{
 switch (Fcode)
 case FKEEP:/ /Latch instructions
 FKEEP ();
 break;
 case FMOV:// Data transfer instructions
 FMOV ();
 break;
 case FSET:// Data setting instructions
 FSET();
 break;
......
}
```

4.3. The Realization of Output Refresh Program

Execution result of the Virtual Machine is not directly output to the external ports, but stored in an output image area. The value of the output image area is copied to the corresponding output ports at the end of the each scan cycle, and then realizes the output register's write operation by executing function IoOutPut (outBuffer) and drive peripherals to complete the corresponding control function. The key code of output refresh program is as follows:

```
for (int index=0; index<chanNum; index++)
{
 p = (IoVarDesc*)GetVarDesc(chanNum);// p
points to I / O variable descriptor
 OutBuffer [index] = p→ramAddr;// Copy the
sampled data to output mapping area
 p++;
}
IoOutput(OutBuffer);// Output port status
```

5. Performance Testing of Virtual Machine of Embedded Soft PLC Running System

The main characteristic of Soft PLC better than the traditional hard PLC is fast response [7], so the focus of the test is virtual machine real-time performance. Accuracy is the basic performance indicators of soft PLC system, the Virtual Machine should be ensures the accuracy of logic operation. Reliability is an important performance indicator of soft PLC system, requiring long-term and stable work.

In order to test the real-time performance, accuracy and reliability of the Virtual Machine, taking PowerPC/P1010 (Freescale) as the hardware platform and Vxworks as the operating system, which was used to run Virtual Machine of the soft PLC running system. The result was that Virtual Machine can successfully complete data collection, program execution as well as the output refresh, and performed accurately.

In order to verify the real-time performance of Virtual Machine, an oscilloscope combined with a time function provided by the system was used to test more than 1000 lines of logic instruction. Test the time required when performing program according to the time indicator flashes and high and low changes of an oscilloscope. The result: the system took about 200 microseconds to perform more than 1000 lines of logic instructions, but the traditional PLC took a few milliseconds even tens of milliseconds.

In order to verify the reliability of the Virtual Machine, we kept the system running for a long time, it performed reliably in the whole process. So it can meet the requirements of industrial site for system's reliability.

6. Conclusions

With the emerging of controllers based PC platform and embedded operating system, as a cropped, high portability, more secure open platform complied with internationally accepted industry standard, PLC virtual machine is in line with the development trend of controllers[8]. Virtual Machine is the core of the running system. It uses software technology to realize the function of the microprocessor of traditional PLC. It has good performance in real-time, accuracy and reliability. Especially in the execution speed, it is much better than traditional hard PLC system. Virtual Machine technology ensures the independence and portability of the target code. No matter what operation system, the target code does not need to be changed. Therefore, Virtual Machine strengthens the openness and flexibility of soft PLC system and makes the system to have a more broad application prospects and development space.

Acknowledgements

This work was financially supported by National High Technology Research and Development Program 863 (2011AA11A102), The Natural Science Foundation of Tianjin (Grant 13JCYBJC39000), The Natural Science Foundation of Civil Aviation University Of China (2012QD21x).

References

[1]. Namie, Masaki, Application of PLC for PA and future evolution of PLC based process controller. *Japan Technical Assoc of the Pulp and Paper Industry,* Japan, 2006, pp. 37-43.

[2]. Zhibing Ren, The research and implementation of the embedded soft PLC system, *Harbin Engineering University*, 2005.

[3]. Bin Blunden, The design and implementation of a virtual machine, *China Machine Press,* Beijing, 2002.

[4]. Lei Zhang, Research and implementation of soft PLC running system based on PC, *Taiyuan University of Technology*, 2008.

[5]. Wenkai Zhu, Weihua Wang, Han Ding, Open soft PLC based on embedded PC, *Machinery & Electronics*, 3, 2002, pp. 12-18.

[6]. Zihua Cheng, PLC principle and programming example analysis, *National Defense Industry Press,* Beijing, 2007, pp. 16-30.

[7]. Jae-Ho Lee, Heung-Nam Kim, Implementing priority in-heritance semaphore on C/OS real-time kernel, *Software Technologies for Future Embedded Systems,* 5, 2003, pp. 83-86.

[8]. SoftPLC Corporation, http://www.softplc.com