Real Time Multi-tasking Applying to AC800M Controller

Wang Guan[1], Feng Xing[2]
[1] University of International Relations, Beijing 100091
[2] China Nuclear Control System Engineering Co..Ltd. Beijing 100176

Abstract - In this paper I describe the demand and definition of multi-tasking. Introduce the specific methods and key technologies used in AC800M which applies digital multi-tasking controller. Practice has proved that these methods of priority, scheduling mechanis and so on can meet the requirements of real time multi-tasking for controller, and have achieved the anticipated target in practical operation.

Index Terms - Digital Controller, Real Time multi-tasking, Scheduling Mechanism, Priority.

1. The Demand and Definition of Real Time Multi-Tasking

Automatic control system is consisted of plants and controllers. Although automatic control system has various plants, demanding for real time multi-tasking is always the basic requirement to its controllers.

Real-time means “immediately” and “now”, it refers to the system’s capability to react to the specific input quickly enough that it can control the device, which releases the real-time signal. In other words, the controllers’ can respond the external events immediately, and not only finishing the processing of these events within the regulated deadline, also make sure all the real-time devices and real-time tasks operated coordinately.[1]

The function requirement for control system can always be divided into several control tasks. When we mention the concept of multi-tasking, we hope it can have several tasks process at one time. However, the more complicated of our control system, the higher demand for controller to complete multi-tasks simultaneously is needed, which we called multitasking character. System will crash or to behave in an unexpected fashion if it not handle properly while seizing time slice during preemptive multitasking[2].

The purpose of this paper is to discuss the characteristic of real time multi-tasking and to introduce the method and technique of compiling by using AC800M controller published by ABB Company.

Real time and multi-tasking are two aspects of an issue. AC800M controller is based on the control of PC(including embedded PC and soft logic PLC), and carrying out IEC61131-3 International standard programming language. Its kernel of mathematical processes is CPU. Nowadays, CPU is fast enough that it can satisfy most of requirements of real-time while processing single task. However, since CPU processes commands one after another, which means essentially it works sequentially. Therefore, there need appropriate methods and techniques to meet the demand of real-time multi-tasking when we want to control all the real-time equipments in system to work coordinately.

2. Methods to Implement Real Time Multi-Tasking

AC800M supports Real Time multi-tasking, which is to provide a task scheduling mechanism-a systematical organization of programs indeed-as mentioned before. On one hand, the organization of programs decide how the tasks are scheduled; on the other hand, the way in which programs are organized is largely depended on the quality of the software and closely related to the programming language. Because when there’s one resource in a configuration, it can also correspond to several tasks. When one resource is connected to several tasks, it actually requires flexible and configurable controls over the execution frequency of different parts of the program, in order to coordinate different tasks and make rational use of the resources. This not only meets the demand of the plants, but also maintains optimized resource efficiency. AC800M provides the following three methods:

a) Set priority of execution among several tasks

b) Divide a task into two identical execution circles, but execute the one with a shorter time period

c) When execute the task with shorter time period, if possible, extend the period of execution circle, so that several tasks could have enough time for running.

AC800M can provide two scheduling methods, preemptive scheduling mechanism and non-preemptive scheduling mechanism. Different scheduling mechanisms have great impact to the system behavior. It means when the hardware resources are the same while scheduling mechanisms are different, the systems’ quality of Real Time multi-tasking can differ a lot.

2.1 Non-Preemptive scheduling mechanism

By using this scheduling mechanism, once a certain task is executed, the task will continue until either all the programs or all the functions of the task are executed. After the implementation of the current task, the next task which will be executed in the waiting list enjoys the highest priority; if the waiting tasks share the equal priority, the task which holds the longest waiting time should be executed first. The task just accomplished will not be considered at the scheduled time until its next interval.

Compared with preemptive, it will be much easier to design non-preemptive operating system and to execute program, but in fact the performance of the function of multitasking is much worse. Because related to the accomplish time of other tasks, the actual time consuming of the task may vary widely. If time consuming of one task increases occasionally, the other tasks will be postponed and this detention will cause the uncertainty of the task

execution which means the exact execute time of one particular task cannot be predicted. Such systems are called non-deterministic, which cannot usually be used in time-critical applications.

2.2 Preemptive scheduling

Preemptive scheduling is needed in order to allow a system to have certainty, which means to have a certain sequential relationship among events that control the system. Under this kind of scheduling mechanism, when a task with higher priority is submitted, tasks currently running are suspended and the task with higher priority will run instead. After the execution of the preemptive task, the suspended tasks will continue to run. Since the priority is divided into many levels, when a task with higher priority is submitted, many running tasks have already been suspended probably. Preemptive scheduling delivers a good real-time performance. As long as the task with high priority meets operating conditions, or in ready state, it can run immediately, which ensures the real-time performance of the system. However, if the high priority task has an infinite loop, the tasks with low priority cannot execute correctly.

With two tasks with equal priority as an example, task 1 and task 2, the execution period of task 1 is 10ms and the lifecycle is 50ms, and the execution period of task 2 is 30ms and the lifecycle is 200ms. For the two tasks with the same priority, AC800M controller decides that the task with short execution time is run first. As illustrated by figure 1, task 2 is delayed for 10msleading to the uncertainty of the execution time.

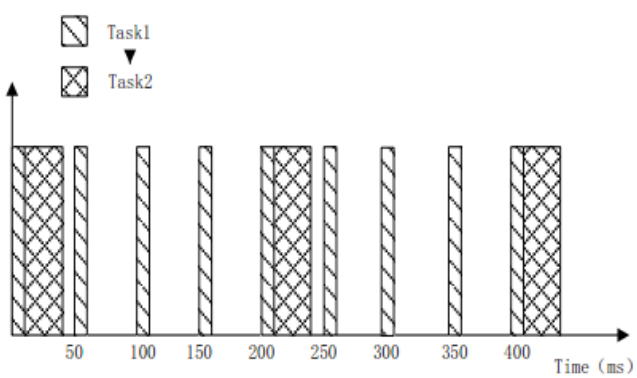

Fig. 1 Two tasks with equal priority

In order to not delay the execution of task 2, an offset is set for task 2, the execution of task 2 will no longer be delayed at this moment, and after every run circle, there will be an interval, which ensures the execution of tasks with the lowest priority such as data transmission. As shown in figure 2.

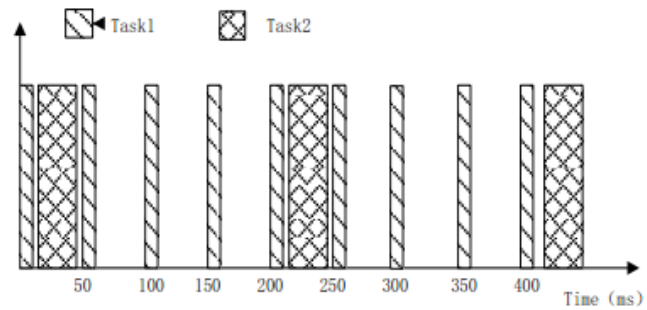

Fig.2 An offset is set for task 2

From the Figure 1 we can see that the execution time for next task becomes:

$$t = (n+1) * (T_{running}) + offset$$

$T_{running}$: Execution time.

The setting of offset has great correlation with task scheduling.

2.3 The setting of offset

Provided that there exists four tasks in a controller, their execution time and execution period shows in chart 1:

Chart 1 execution time and execution period for each task

Task	Execution period (ms)	Execution time (ms)
1	50	10
2	150	20
3	300	30
4	600	20

This chart illustrate that the execution time of the longest period task is twelve times longer than the shortest period task's. Therefore, we can use 50ms as a time slot and draw 12 time slots as follow:

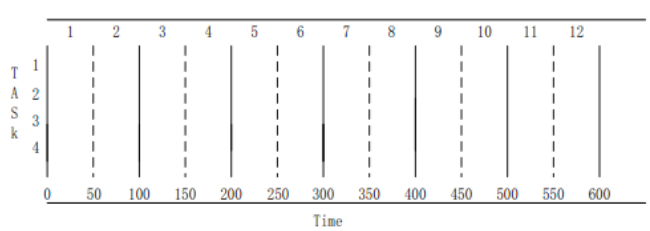

Fig.3 Time slot and task

In the same priority, the shortest period task will be the first to execute. Since the period of Task 1 is 50ms, Task 1 will be executed at the first place, and be executed in every time slots, like Figure 4 shows:

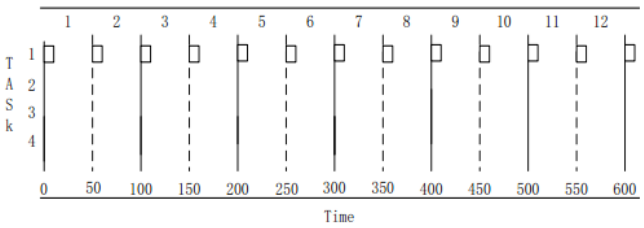

Fig.4 Task l is executed in every time slots

The next task is Task 2. The execution period of Task 2 is 150ms, which means that it will be executed every three time slots. However, there will set an offset for Task 2 in order to avoid the delay of execute Task 2.

The formula for offset:

Offset time=Execution time (Task 1) + [50-Execution time (Task 1)-Execution time(Task 2)]/2

Via this formula, the offset time is 20ms. Then when it comes to n+1, the execution time will becomes to (n+1)*execution period + offset time, as Figure 5 shows:

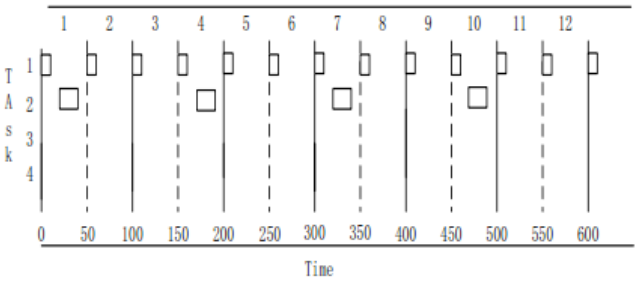

Fig.5 Task 2 is executioned in the time slot

Because the first time slot was occupied, the Task 3 can only be executed by second time slot, the offset of Task 3 is:

Offset=50(first time slot) + Execution time for Task 1 + (50-Execution time for Task 1-Execution time for Task 3)/2=65ms. The Execution period is 300ms, so it can only be in 2 and 8 time slot, like Figure 6 shows:

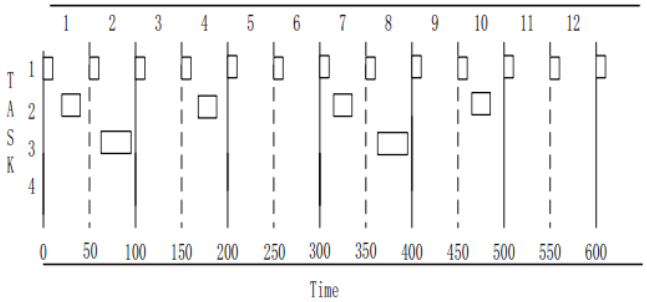

Fig.6 Task 3 is executed in the time slot

The offset time for Task 4 can be calculated exactly the same as Task 3

Offset=100 + Execution time for Task 1 + (50 – Execution time for Task1 – Execution time for Task 4)/2=120ms. Like Figure 7 shows:

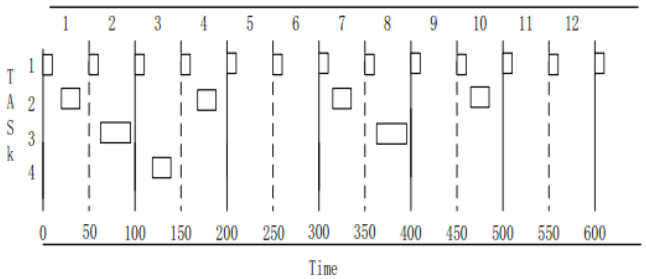

Fig.7 Task 4 is executed in the time slot

From the Figure 7, we can see that these four tasks assigned evenly in the time slot, which ensures the requirement for same priority of real time multi-tasking.

2.4 Method to set priority

In order to realize real time multi-tasking, it needs to acclaim several tasks in a resource and assign different time deviation for each task. This can make sure which specific task is working, or owning the control right to the CPU. Others have to be in the state of block or ready to avoid the appearance of delay or interrupted.

However, although it has already setting the time deviation, some tasks will delay occasionally. It can be solved by setting priority. There have two tasks, the execution time is 10ms and 50ms, and the execution period is 30ms and 200ms. The time deviation for Task 2 is 80ms:

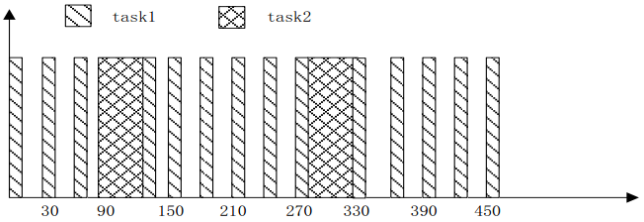

Fig.8 Two tasks with equal priority are executed

From the Figure 8, Task 1 is delayed, and in order to avoid this delay, it should set Task 1's priority higher than Task 2. Then we get this:

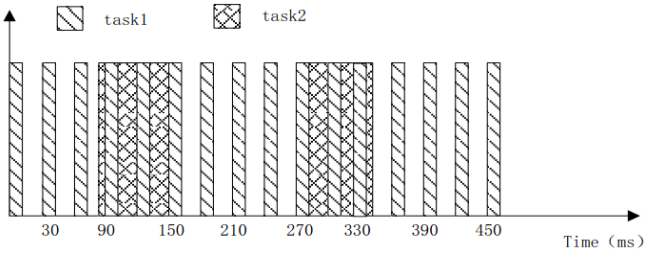

Fig.9 Two tasks with different priority are executed

Figure 9 show that after setting the priority for Task 1, and schedule properly, it can solve the problem. We can see that there are two disruptions during the execution of Task 2. This method can be used only under the condition that Task 2 is not strict with time.

In the field of industrial control, if adopt this method to

get period sampling without any delay then it can ensure the execution of task immediate. Therefore, the controller possesses the characteristic of real time multi-tasking. However, if strictly according to the priority to schedule tasks, it can greatly improve real-time of the system, but which will loss fairness. Since if there has an endless loop in a higher priority task which barely has wait time, then lower priority tasks will never have chance to run. Consequently, if we want controller can finish several tasks not only real-timely but also simultaneously, the methods we have introduced need comprehensively analyzed and used.

3. Conclusion

The characteristic of real time multi-tasking for AC800M can cut task into several independent task modules, which makes design process much easier; and can handle some demanding tasks quickly and reliably. The resources of AC800M can be fully used, which meet the requirement of real time multi-tasking for industrial control system through effect design method.

References

[1] Zheng Yu-quan Miniature grab design multi task real-time kernel. Electronic Engineering Magazine, 2004-03-22.

[2] Han Li-xin, Dou Dong-sheng. Real time multi task characteristics of programmable controller. Manufacturing automation, 2003, 2 (25): 59~62.