THE SOFTWARE STRUCTURE AND PRINCIPLES OF MAIN TASKS SCHEDULING IN AN EXECUTIVE SYSTEM

[1]**DRAGAN R. MILIVOJEVIC, [1]MARIJANA PAVLOV, [1]VISA TASIC, [2]VLADIMIR DESPOTOVIC**

[1]Mining and Metallurgy Institute Bor, Zeleni Bulevar 35, 19210 Bor, Serbia

[2]Technical Faculty of Bor, Vojske Jugoslavije 12, 19210 Bor, Serbia

E-mail: dragan.milivojevic@ymail.com

ABSTRACT

The paper describes a practical way of design and realization one simple real time software, better say: executive system. It is applied on an own programmable logic controller - microprocessor measuring station (MMS). In hardware point of view it is a modular unit based on M68HC11 microcontroller. All its functions are driven and controlled by a special real time working program, modular structure. Many program entities are running sequentially. The tasks are the main program elements. Because of many requirements and shortage of resources it was necessary to introduce a priority ordering in task execution. The article explains some principles of software organization and task management. The results achieved in real environment are also included.

Keywords: *Executive System, Real Time Operation, Task Scheduling, Process Control, Interrupts*

1. INTRODUCTION

The complex distributed control system was developed for control and monitoring of copper production process in smelting plants in Bor (Serbia). Microprocessor measuring Station (MMS) is a main unit of that system. It is a kind of programmable logic controller based on microcontroller MC68HC11. The core of the unit is the one board computer (OBC) extended with input-output modules. A (industrial) PC is added as local or remote workstation, Figure 1.

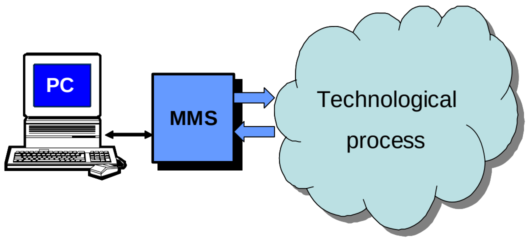

Figure 1. *MMS in simple control system block schema*

The most important functions of MMS are process parameter measuring and data collection. It is done via analog and digital input channels. The action to technological process is performed by execution of appropriate commands as a result of local control algorithm, or as a consequence of running of the remote command mechanism.

Understanding the underlying hardware of the real-time system makes the use of both hardware and software resources more efficient. Architectural features such as instruction set, addressing mode, memory organization and input/output functions are very important for optimal usage of available performances. Microcontroller MC68HC11 has a two-byte accumulator, a couple of index registers and executes a set of more than 90 instructions. A computer operating properly (COP) watchdog system protects against software failures. The illegal operation code detection provides a nonmascable interrupt indicating false state. Four independents timers are available. Presence of very stabile 8 channel 8-bit A/D converter gives a great facility for analog input connection. The serial communication interface allows efficient data exchange between MMS and subordinated PC, which is used as a system workstation. The input/output modules perform conditioning of input signals and transforming the output commands to actuating assembles. For easy local control and parameters changing, the functional keyboard and LC display are added. The described equipment is a hardware platform for realization of control system as real-time software.

Modern manufacturing faces two main challenges: more quality at lower prices and the need to improve productivity. Those are the requirements to keep manufacturing plants in developed countries, facing the competition from the low-salary regions of the world. Other very important characteristics of the manufacturing systems are flexibility and agility of the production process using present equipment and technology [1]. Current computer control facilities are networks of programmable logic controllers, microcomputer-based distributed control systems and real-time controllers [2].

2. SOFTWARE STRUCTURE

The way of MMS operation, response time and other performances classify it in a soft real-time unit class [3]. It means that the controlled processes are not extremely timely critical. For performing and supporting of functions in monitoring and process control, the appropriate software had to be realized. Hardware/software integration is usually more art than science; it relies on the intuition and experience of designers and integrators [4].

Executive system is a complex operational program with a few global functions:

- power on and self testing,
- hardware resources managing and functions supporting,
- interaction with an environment which has time-varying properties,
- execution of appropriate process interactions with limited resources.

The MMS ability to meet demands above depends on its capacity to perform the necessary activities in the given time. The process requirements as the consequences of different events very often are unexpected and irregular [5]. Executive system has to solve all of demands on the defined way and in determine time intervals. Task is the basic element of executive system. It is defined as small amount of code (a small program module) which performs a specific function. In concrete case the terms task and process are interchangeable. It is possible to differentiate a few discrete task states, Figure 2 [6].

In concrete case the tasks are in three states only: Ready, Suspend and Activate. Extant state means available to run, and it is finished after power on sequence. Terminated state is not possible because the program is residential in EPROM and not removable. Executive system must be able to change the state of all tasks according to scheduling and resource management conditions. The executing tasks are in active state. The tasks are waiting for resources (CPU, or memory) in Ready and to terminate the other task in Suspended state. The commands which perform the state changing are shown as the graph branches on Figure 2.

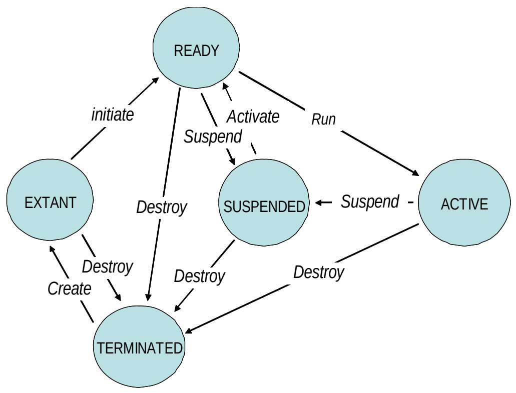

Figure 2. *Task states and relationships graph*

The MMS control program is, in fact, a multitask executive system. It must provide three specific functions:

- task scheduling, determines which task will run next,
- task dispatching, performs the necessary bookkeeping to start that task and
- intertask communication, changes data between the tasks [6].

There are many techniques of task scheduling: polled loop mechanism, state-driven code, interrupts driven system, foreground/background principles and so on [7].

It was chosen for MMS executive system to use combination of polling mechanism and interrupt principles to maintain the task handling and execution. Polling is the simplest way to allow the fastest response to a demand. It is executing a single and repetitive group of instructions testing the flags that indicate whether or not some event has occurred. If the event has not occurred, the execution of polling loop continues. It is useful for systems when overlapping of events is not allowed or kept to minimum. But the practical MMS application requires different task execution at the most of time, and it is not possible to control the flow of program sequence only by polling technique. That is a reason to introduce a complex task scheduling system combining polling and interrupts. Instead of execution of jump-to-self instruction and waiting on interrupt request, the program runs a wide instructions endless loop. As a result of occurring of a periodic or sporadic (or both) interrupt request, the task switching takes place, and changes the program flow.

Design techniques of real-time executive system include efficient solutions of intertask communication and synchronization.

2.1. The MMS software creation

The microcontroller MC68HC11 has two basic modes of operation: single chip and expanded mode [8]. Because of process control demands and operation mode performances, the expanded mode is chosen. MMS as an one board computer has the associative memory, RAM and (or) EPROM. The executive system as a machine code is situated in this memory: in RAM when software testing is performing, and most frequently in EPROM as run time version.

Because MMS was designed, developed and realized as an integral unit, the process of its integration and synthesis includes the hardware component as well. The way of system creation has a few phases illustrated by Figure 2.

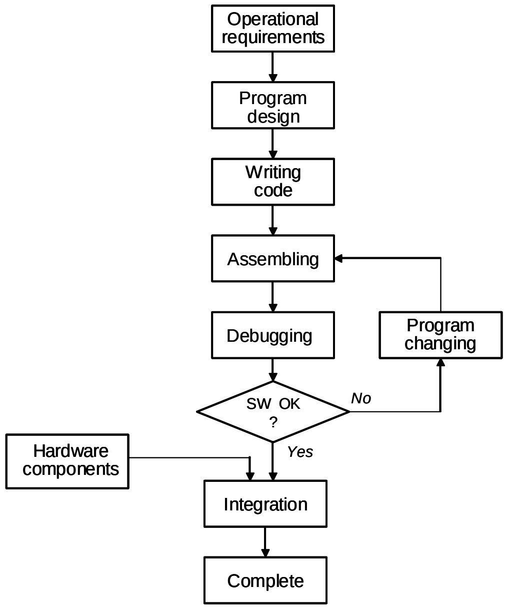

Figure 2. *Flowchart of MMS software design*

Although it was an intention MMS to be a general purpose unit, for its concrete usage in copper production metallurgy processes, the operational requirements are respect: the sampling rate from 1 second up to 5 minutes, a command response time less than 5 seconds and measuring and control of about 60 different input parameters. The efficient communication facilities are also required because of distributed process parameters. The first step in Figure 2 means honor the above demands. After an overall design of necessary program modules, the procedures drivers and routines were designed and developed, step 2. The source code of program is written in a symbolic Motorola language on Intel PC. For that purpose a special environment of ∏-assembler was used [9]. Assembling - composition is the combination and linking of many of the program parts to form a larger working program. Program composition (and hence decomposition) is complicated by the need to avoid interactions between and within procedures, routines and input/output drivers. The debugging step is very useful to examine the most of programmed functions [10]. But interrupt driven routines can not be tested by debugger. The simulation methods are applied for those purposes. After debugging and testing software, the corrections are undertaken and assembling is performed again. That phase is done by simulation on a PC in developing environment, and downloading the executive code in to MMS RAM. Some times certain hardware improvements are also done (if there is need and possibility).

Using described methods the complex program structure was realized [10], Figure 3.

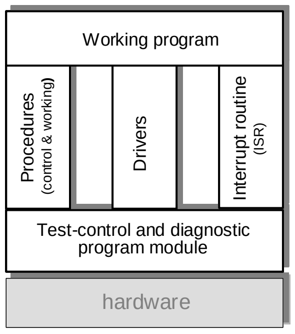
Working program
Procedures (control & working)
Drivers
Interrupt routine (ISR)
Test-control and diagnostic program module
hardware

Figure 3. *The MMS software block diagram*

Test-control and diagnostic programs are executing only at the power on sequence (self test at start of operation), or as a consequence of a MMS test command from PC.

Working program is running continuously as operational software by executing the program entities like procedures, drivers and interrupt handling routines.

2.2. Synchronization of tasks execution

The real-time systems interact with their external environment. The set of external objects products the requirements for a control system to respond on. Those requirements are external events [6]. The execution of build in self test (BIST) [11] and running the working program may cause the change of contents of MMS general status register (GSR). That can be named as internal events. The real-time software has to take appropriate action on any event, but at regular and timely rate. An event is a mechanism which allows interdependences between requirements, interrupts and changes in resource states. Synchronization of concurrent processes provoked by events requires the use of an event flag situated in status register. The events may occur at a fixed time intervals - periodically, quite stochastically (asynchronously) or combined. If any of regular events (external and internal) occurs the appropriate bit (flag) in general status register will be set, Figure 4.

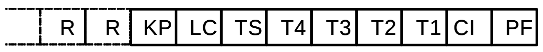

Figure 4. *General status register configuration*

Where: PF - Power Fault, CI - Comm Irq, T1 to T4 - Timer 1 to 4 Irq, TS – Test Switch, LC – Local Command, KP- Keyboard Pressed, R – Reserved.

The most important events in the MMS case are interrupt requests: from serial communication interface (Comm Irq) and from timers (Timer Irq). Timer interrupt requests are periodic with programming time rate and may be qualified as internal events. Communication interrupt request, as external events, is stochastic from MMS point of view. (In the simple network, Figure 1, PC starts a communication session as master node in its own time rate [12]).

There are a couple of interrupt service routines (ISR). Those routines solve the hardware interrupt requests from serial communication port and from any of four independent timers. The start addresses of all of routines are written in interrupt vector table on the highest memory addresses. During the executive code downloading in RAM or writing into EPROM the high memory locations ($ffd6 - $ffe2 respectively) are filled by starting ISR addresses creating a vector interrupt table (VIT).

The executive system consists of three main parts:

- test, control and diagnostic module, for examining hardware correctness and input-output functions,
- operating program with procedures and routines and,
- working program containing concrete applications solutions.

The test and control programs are executing on power on, or reset sequence (start of day) [11]. After successful passing, the operating program takes place. It is running as a complex program loop activating some of procedures according to contents of general status register. It obtains some common functions like measuring, collection data and information interchanging with subordinated PC. The interrupt routines start to operate as a consequence of interrupt requests and change appropriate bits in status register

The working program is designed to perform concrete particular functions:

- choose sampling rate and support measuring,
- present the result in appropriate format,
- perform data exchanging with PC in both directions,
- perform defined commands (local and remote) to control process etc.

Task as a section of code is used to execute a particular operation. The modules of executive system described earlier (procedures, subroutines, and drivers, Figure 3) are divided into specific tasks. Some of them are the single tasks like driver for LCD, but more complex functions are performed as a logical sequence of tasks and subtasks.

The executive system is running as a complex interruptible endless loop. It is performing a sequence of background procedures and drivers. Some of them set defined bits in GSR. The occurring of interrupt requests change contents of that register also. Many test instructions in the main program loop examine status register bits (the decision boxes on Figure 5) as often as possible and activate the appropriate program modules (procedure or driver).

Real-time systems are characterized by the asynchronization of activities (events). There are many active events at a time, all of which could demand simultaneous attention. This situation is called concurrent processing [13].

The executive system has to decide which of several competing tasks should be processed first. Because some of activities are more important then the other, the task execution priorities are introduced. In a step of operations requirements (Figure 2) the task signification was established, see Table 1.

Table 1. *Task priorities*

Priority	Task	Level
1	Communication (message receiving)	highest
2	Inputs sampling	
3	Real time clock (RTC) and calendar	
4	Test switch	
5	LED display	
6	Commands execution	
7	Functional keyboard	
8	Message to LCD	
9	Local MMS functions	lowest

There are many methods of priority ordering. In MMS executive system the positional priority without change in time is in use [6] (Figure 5). The data transfer (receiving) from associated PC is the highest priority task. Always when communication session ends, or any other procedure finishes, the flag CI in GSR [14] is checked first. If it is set, communication session starts again. If there is no communication interrupt request, the measuring can take place, after that real-time clock, the commands, and so on.

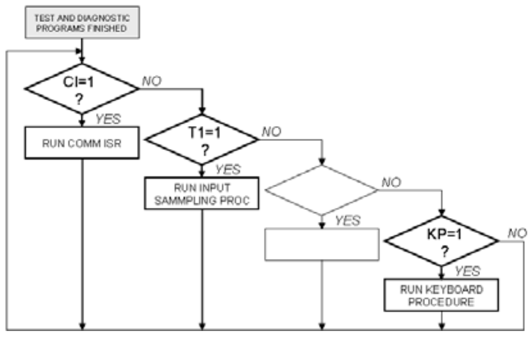

Figure 5. *Positional priority task processing*

The flag examination illustrated by decision box on the above diagram is realized as one timely optimal assembler procedure, Figure 6:

```
        Procedure FlagTest;
        begin
          psha;
          pshb              {save accumulators}
L0:       ldd GSR;          {read GST into a+b}
          bitb 02 hex       {test bit CI in GSR}
          je L1             {CI=0}
          jsr Receive       {Receiving data}
          bra L0
L1:       bitb 04 hex       {test  bit T1 in GSR}
          je L2             {T1=0}
          jsr RTC           {Real Time Clock}
          bra L0
L2:       …
          …
          pulb;             {restore accumulators}
          pula;
        rts;
        end;
```

Figure 6. *Part of GSR bit testing procedure*

MMS builds a simple two-node network with subordinated PC as a workstation, Figure 1. The classical serial asynchronous communications with 19.2 kbps are in use. It means for the transfer of one character about 0.5 ms is needed. The standard message from PC is 11 bytes long [12] and it takes

$$T_{receive} = 11 \text{ x } 0.5 = 5.5 \text{ ms}$$

The necessary time for execution of all instructions of a *Receive* procedure is less then 0.2 ms. The PC message receiving takes not more then 6 ms. In the worst case that is a delay time to honor the timer interrupt requests. For MMS used as a soft real-time unit it is quite acceptable for practical applications. The duration of the measuring procedure, real-time clock calculation, command sending and other functions are timely optimized and much shorter than 2 ms.

It was already mentioned that the event flag bits in GSR are set by interrupt service routines, or regarding to results of execution of some procedures and data processing. MMS should operate as an autonomous unit, but more often it is configured in the network with a PC as a workstation. For stabile and reliable operation special Supervisory Control And Data Acquisition (SCADA) software on PC is developed. It includes the communications module for supporting of data transfer between PC and MMS. Master-slave principles are in use, and PC (master node) initiates, controls and maintains every session. The command message originated by PC is sent as a block of 11 characters and MMS has to receive it as an entity. For MMS the start of session appears quite stochastically, and it must react immediately to receive the data. When the character is accepted the receive data register full (RDRF) bit is set [8]. This event raises the interrupt request to microcontroller and activates appropriate ISR. That routine sets the CI bit in GSR. The serial communication interface in MMS uses double buffering mechanism, which allows receiving of two characters without the corruption of the previous with the next one. That means that communication procedure of the executive system can start with a delay which is not longer then 0.5 ms (the one byte transmission time with the speed of 19.2 kbps) to receive a correct message. In the other case the communication session should be unsuccessful.

Although it is possible to change priority, the communication tasks have the highest, rather than the other. It is acceptable for the applications with relatively slow changes of controlled process parameters, like in copper production plants. For the other applications task execution priority order is not the same. In some of them the MMS is a self-contained unit without network environment [14].

Figure 6 shows a chart of task movement in real MMS function with assumption of 100 ms sampling rate, and about 5 communications interrupts per second.

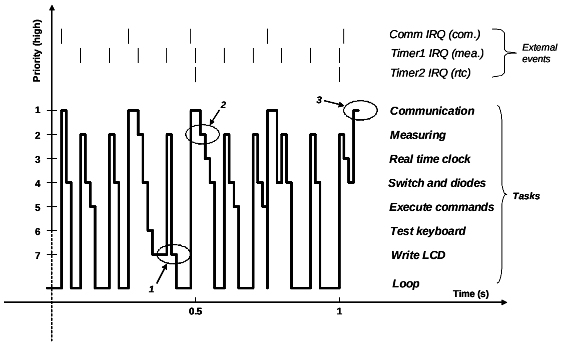

Figure 6. *Plot of task movement*

As it can be seen, lots of time is spent for running the loop instructions. But there are a few interesting points:

- 1, the lower priority task (show on LCD) is interrupted by measuring (with higher priority) and finished later,
- 2, the communication session runs when measuring should start. The start of measuring is delayed because of lower priority and
- 3, start of communication session is delayed until real time clock task is finished (< 0.6 ms). If the received message is wrong, the session will be repeated.

3. FAULT TOLERANCE

Fault tolerance is the ability of the system to continue to function in presence of hardware or software faults. It is designed to increase reliability of system test [6]. During MMS operation under the executive system running, there are some of hardware tests of input/output modules. Typical one is saturation test which compares the analog/digital conversion output with the boundary values (0 and FF hex). The analog input chains are hardware adjusted in a way that the digital representation fails in interval (1, FE hex). The difference indicates a fault and sets appropriate flag in the status register. This is a built-in-test-software [11].

Analyze of Figure 6 shows a real possibility of missing a command from PC, or some timely incorrect measuring. In the hard real-time systems it should be very serious problem, but in case of MMS it is possible to overcome the mentioned disadvantages. The unsuccessful data transfer initiates the new communication session from PC as soon as possible [10]. MMS is used for relatively slow industrial processes (the process parameters changes are not so dynamic) and a delay of a couple of micro seconds for some of measuring points is not so important. Used in such conditions, the MMS works very stable and reliable. As a consequence of task scheduling conflicts, it is possible to lose the message, the command from PC, or measured data from MMS.

The software communication modules in PC SCADA program and MMS executive system contain error recovery routines, and start the session again. In some applications with a compact entire network (hard PC-MMS interconnection) the starting of measuring is dictated by commands from PC, and the event conflict is avoided. It means that communication sessions are more frequent than the input measuring.

4. SUMMARY

The MMS most demanded practical usage (electrical power monitoring) with 1-second data, operates with 100 ms communication interval. There are less than two missing data daily, which is less than 2 / (60 x 60 x 24) = 0.0025 %. In usual applications in copper production processes a sampling rate is one minute. If 1-second communication frequency on PC is chosen [15], there is less than one data message missing per day, which is less than 0.07%. It is not easy always to determine exactly the consequence of task executing conflicts. Sometimes the data message shortage may be a consequence of bad transmission. Anyway, the appropriate program solution on dedicated PC software generates the nonexistent messages using the interpolation method. The described solutions of MMS executive system and event priority scheduling give very good results in industrial environment applications.

ACKNOWLEDGEMENT

This paper is supported by the Grant of the Ministry of Science of Republic of Serbia, as a part of a project EE18016 within the framework of Technological development programme.

REFRENCES

[1] J.N. Pires, PIRES, "Semi-autonomous manufacturing systems: The role of the human–machine interface software and of the manufacturing tracking software", *Mechatronics*, 2005, Vol. 15, Issue 10, 1191-1205

[2] G. Hassapis, "Implementation of model predictive control using real-time multiprocessing computing", *Microprocessors and microsystems*, 2003, Vol. 27, Issue 7, 327-340

[3] P.A. Laplante, "Real-time Systems Design and Analysis", *IEEE Computer society press*, New York, 1997

[4] K.H. Kim (Kane), "Object Structures for Real-Time Systems and Simulators", *Computer, IEEE*, August 1997

[5] M. Jozeph, "Real-time Systems", *Prentice Hall International*, UK, 1996

[6] G.C. Barney, "Intelligent Instrumentation, Microprocessor Applications in Measurement and Control", *Prentice Hall, Hertfordshire*, 1988

[7] J. Roland S. Burns, "Advanced Control Engineering", Butterworth – Heinemamm, Lineacre House, Oxford OX28DP, 2001.

[8] "M68HC11 Reference Manual", 1990, *Motorola* Inc.

[9] Z. Radonjic, "∏-assembler user guide" (in Serbian). Nis, Serbia, 1996

[10] B.P. Douglass, "Real-time UML" *Addison-Wesley*, Inc, 1998

[11] B. Djordjevic, M. Jevtic, M. Damjanovic, "Digital Real-Time Systems Testing", *Charisma, Evidenz und Caritas, Ruhr-Universiteat Bochum*, A31-A40, 1997

[12] D. Milivojevic, V. Tasic, "MMS in Real Industrial Network" *Information technology and control*, Vol. 36, No. 3, 318 – 322., 2007

[13] R.S. Barns, "Advanced control engineering", *Linaere House, Jordan Hill, Oxford*, 2001

[14] D. Milivojevic, V. Tasic, "Some Software Elements of the Microprocessor Measuring Station**".** *Acta electrotechnica et informatica*, No.2, Vol.7, 2007

[15] D.R. Milivojevic "MMS as an element of industrial control systems", *Ph.D. thesis, University of Belgrade*, TF Bor, 2008.