THE RC 4000 REAL-TIME CONTROL SYSTEM AT PULAWY

PER BRINCH HANSEN

(1967)

This paper describes a real-time control system implemented on the RC 4000 computer with an internal store of 4096 words. The system permits a number of independent programs to be executed periodically on a time-sharing basis. The first version of the system performs supervisory control of the ammonium nitrate plant Pulawy II in Poland. After a description of the Pulawy system, the choice of a time-sharing scheme and the handling of shared facilities are discussed. This is followed by an evaluation of the size and performance of the system.

1 Introduction

The multiprogramming system described in this paper was developed by Regnecentralen on contract with the Danish engineering company Haldor Topsøe. In connection with this project, Regnecentralen also developed a medium-sized computer, the RC 4000, which is specially suited for real-time control applications (Brinch Hansen 1967).

The system is implemented on the RC 4000 computer with an internal store of 4096 words (backing storage is not used). It permits a number of independent programs to be executed periodically under the real-time control of a monitor. For each program, the operator can select the start time of its first execution and the time interval between its subsequent executions. The programs are executed in a simple time-sharing scheme, in which each program in turn is allotted a small quantum of computing time. A critical feature of any multiprogramming system is the handling of shared

P. Brinch Hansen, The RC 4000 real-time control system at Pulawy, *BIT 7*, 4 (1967), 279–288.

facilities. We have adopted the technique of binary semaphores suggested by E.W. Dijkstra (1965).

The first version of the system will be installed in 1967 in the ammonia nitrate plant Pulawy II, constructed by Haldor Topsøe in Poland. Here, the RC 4000 will perform regular alarm scanning, data logging, and evaluation of production and consumption figures.

In the following, we describe the supervision of the Pulawy plant in order to illustrate the requirements of a real-time control system and the difficulties of implementation. This is followed by a discussion of the time-sharing approach.

2 The RC 4000 Computer

The RC 4000 is a single-address, binary computer with typical instruction execution times from 2.5 to 5.5 microseconds. The following characteristics apply to the basic model used in the Pulawy plant.

Store: The internal store has a capacity of 4096 words. Each word contains 24 information bits, 1 parity bit, and 1 protection bit.

Registers: There are four working registers of 24 bits each. Three of these also function as index registers. The registers are addressable as the first four words of the internal store.

Addressing: Words of 24 bits and half-words of 12 bits are directly addressable. Address modification includes indexing, indirect addressing, and relative addressing.

Arithmetic: Integer arithmetic with operands of 12 and 24 bits is standard.

Input/Output: The standard data channel performs transfers of single words between low-speed devices and working registers under program control. Program execution continues while input/output operations are in progress.

Program Protection: In the RC 4000, the monitor program consists of all storage words in which the protection bits are set. A program stored in an unprotected area can neither alter nor jump to a protected area. All input/output operations as well as control of the interruption system and storage protection are handled by privileged instructions, which can only be executed within the monitor. Attempts to violate the protection system cause program interruption.

Program Interruption: The interruption system can register up to 24 signals simultaneously. These can be enabled and disabled individually. The

interrupts are examined after each instruction; an enabled interrupt will transfer control from the current program to the monitor. All interrupts are disabled when the monitor is entered; they can be enabled again by a privileged instruction.

3 The Pulawy Installation

The Pulawy II plant consists of three units for the production of ammonia, nitric acid, and ammonium nitrate, respectively. The plant is operated manually under the supervision of the computer. This section describes the configuration of peripheral equipment at Pulawy.

The operator controls the operation of the system by means of a control typewriter. A paper tape reader and punch are provided for the assembly and loading of programs.

Real-time operation is controlled by two interval timers, which generate interrupts every 2.5 milliseconds and every 1 second, respectively.

The computer receives measurements from the plant in the form of 543 analog inputs and 127 digital inputs. The analog inputs are primarily measurements of temperatures, pressures, and flows expressed as voltages. The voltages are converted to decimal numbers by an analog/digital converter. The selection of input points is performed by a relay multiplexer with a switching rate of 30 points per second.

Digital inputs are discrete events registered as single bits in external registers: one type of digital input defines the status of alarm contacts in the plant; another collects single counting pulses from kilowatt-hour meters and bag-filling devices.

A digital output register controls a display panel that shows the operator in which part of the plant alarm conditions exist.

Regular alarm reports and log reports are printed on two strip printers and two typewriters.

4 Process Control Tasks

The computer examines the analog and digital inputs at regular intervals and produces balance evaluation reports, log reports, and alarm reports.

Balance Evaluation: Every 8 hours, a report on 135 material balances is printed on one of the log typewriters. This report shows the consumption of electricity and production of ammonium nitrate during the period. It also includes an evaluation of the total inflow and outflow of materials such

as natural gas, steam, ammonia, and nitric acid. The information for this report is measured as follows: the digital pulses are input every second and accumulated in a table in the internal store; the analog flow values are measured every 5 minutes and accumulated in another table.

Data Logging: Every hour, two reports, each on approximately 275 analog values and 35 pulse counts, are printed simultaneously on the log typewriters. The log reports can be regarded as a snapshot of the operating state of the plant: the first report contains all data from the ammonia unit; the second covers the nitric acid and ammonium nitrate units.

Alarm Scanning: Every 5 minutes, the computer examines the state of 61 alarm contacts; at the same time, 188 analog variables are scanned and checked against alarm limits stored in a table. The operator is warned of alarm conditions by visible lamps and the printing of alarm messages on the strip printers.

Trend Logging: The operator can at any time request regular trend logging of a single analog variable on the strip printers.

Self-Checking: In the event of a computer malfunction, the plant can still be controlled manually while the system is being repaired. The computer must however be able to detect and report such malfunction; accordingly, in idle interals the computer performs checking of the instruction logic, the registers, the adder, and the analog/digital converter.

Operator Control: The operator can at any time type a command to the system on the control typewriter. The main options available to the operator are: selection of the start time and period of each process control task; exclusion of analog and digital inputs from one or more production lines; changing of scale factors and alarm limits of analog inputs; and selection of alternative output devices for the printing of balance and log reports.

5 Multiprogramming Approach

The table below summarizes the control tasks at Pulawy and their real-time requirements:

Task	Normal period	Completion time
Operator control	–	infinite
Pulse integration	1 second	2 milliseconds
Flow integration	5 minutes	10 seconds
Balance evaluation	8 hours	2.5 minutes
Data logging 1	1 hour	2.0 minutes
Data logging 2	1 hour	1.5 minutes
Alarm scanning	5 minutes	15 seconds
Trend logging	–	1 second
Self-checking	–	infinite

In the following discussion, it is important to note that several of the tasks use the same peripheral equipment: the analog/digital converter is used in all tasks except operator control and pulse integration; the log typewriters are shared in balance evaluation and data logging; the strip printers are used in both alarm scanning and trend logging.

From this description of the supervision of the Pulawy plant, we can draw a number of conclusions about the implementation of the real-time control system. We have a single computer that must perform a number of independent tasks, each with its own real-time requirements. The tasks are executed cyclically in periods determined by the operator. We have chosen to implement the tasks as separate programs, because they have individual and variable periods of execution. It is obvious, however, that we cannot fulfill the real-time requirements by executing one task program at a time: two task programs may well demand to be started at the same time; the time required for a single execution of a task program may also be longer than the time interval between successive executions of other task programs. Thus we are forced to introduce a multiprogramming scheme in which the computer performs rapid time-multiplexing among the task programs.

Ease of implementation requires that a task program can be programmed in as straightforward manner as in purely sequential programming; accordingly, time-sharing among task programs must be handled automatically by a monitor program activated regularly by interrupts from a clock.

For the sake of generality and simplicity, the individual task programs must be regarded as being independent of one another. In particular, we do not wish to impose any restrictions on the relative timing of programs. The operator must have complete freedom to change the frequency of task executions individually. He must even be able to stop one or more tasks completely for a period of time. The main problem introduced by this freedom is to find a general way to avoid conflicts about facilities shared among

the task programs.

The solution to these problems is considered in the following sections.

6 Real-time Scheduling

The choice of a multiprogramming scheme must be based on the knowledge of the computing capacity required in worst-case situations. In a heavily loaded system, it may be necessary to establish a system of priorities among the task programs to ensure that the most urgent tasks are completed first. A simple estimate of the system load at Pulawy convinced us that a priority scheme would place unnecessary restrictions on the system. First, we have no backing store to slow down the execution of programs. Second, the majority of the tasks are limited by low-speed devices with input/output times of from 35 milliseconds (analog input) to 70 milliseconds (typewriter output). The programs use less than 1 millisecond each to process an input word or produce an output word; that is to say, a task program uses only 1/70 to 1/35 of the computing time. With only nine task programs, the load is so light that we can afford to serve all programs on equal terms.

The real-time operation of the monitor is controlled by an interval timer, which causes a program interruption every second. The monitor increments a clock counter by one, and examines a table defining the start time and period of each task program. If real-time exceeds the scheduled start time of a program, a flag bit is set and the start time is increased by the value of the period. When the scan of the time table is completed, the interrupted task program is resumed.

Time-sharing among active task programs is controlled by another interval timer as follows: every 2.5 milliseconds, the current task program is interrupted and the contents of the working registers and instruction counter are stored in a dump table. The monitor scans the flag bits cyclically until it finds another active task program, which is then started. After another 2.5 milliseconds, control is transferred to a third program, and so on.

When a task program is finished, it calls the monitor asking it to turn its flag bit off, after which the program does not receive computing time until the next scheduled run.

Switching from one task program to another is also performed, whenever a program must wait for the completion of an input/output operation or whenever a common facility is occupied by another program. Here the restart address in the dump table is adjusted to make the task program repeat the call of the input/output procedure or the reservation procedure the next

time it receives a time quantum. Thus the monitor is relieved of having to keep track of queues of shared facilities.

The selection of a time quantum was influenced by the following considerations. The quantum had to be at least as great as the average response time required by a task program for a single input/output operation. At Pulawy this was about 1 millisecond. The upper limit was determined by the number of programs using the whole time quantum for computing. Too large a quantum would slow down the task programs limited by input/output, and thus degrade the performance of the low-speed devices. At Pulawy, the self-checking program was the only one of this type. Experiments showed that a time quantum 2–3 milliseconds resulted in the shortest completion time for all task programs.

7 Shared Facilities

We shall now consider the problem of mutual exclusion that arises, whenever two or more independent programs demand access to a common facility. Our understanding of this problem has been profoundly influenced by the monograph of E.W. Dijkstra (1965), *Cooperating Sequential Processes*. In the following we discuss his technique of binary semaphores as applied to our system.

The task programs at Pulawy can be regarded as independent programs, in as much they do not depend on explicit knowledge of one another's structures and speed ratios. The programs communicate with one another only for short intervals to ensure mutual exclusion from shared facilities. This communication implies inspection of and assignment to common booleans, called binary semaphores. Each semaphore is associated with a shared facility. It has the value zero if the facility is available, and one if it is busy.

When a program wishes to reserve a facility, it must inspect the corresponding semaphore. If the facility is available, the program will immediately occupy it by assigning the value one to the semaphore; otherwise the program must wait until the facility has been released. In the RC 4000 computer, this reservation can be made by the following sequence of instructions:

```
RESERVE:  LOAD, SEMAPHORE
          SKIP IF EQUAL TO, 0
          JUMP TO, RESERVE
          LOAD ADDRESS, 1
          STORE, SEMAPHORE
```

Consider now the case where program A is inspecting a semaphore. It may happen that the program is interrupted after the loading of the semaphore, but before inspection and assignment to it. The working register containing the value of the semaphore is then stored in the dump table within the monitor, and program B is started. B may load the same semaphore and find that the facility if available. Accordingly, B assigns the value one to the semaphore and starts using the facility. After a while B is interrupted, and at some later time A is restarted with the original contents of the working registers reestablished from the dump table. Program A continues the inspection of the original value of the semaphore and concludes erroneously that the facility is available.

This conflict arises because the task programs have no control over the interrupt system. The only indivisible operations available to the task programs are single instructions such as load, compare, and store. The reservation sequence can, however, be made an indivisible entity by incorporating it in the monitor program. The monitor is protected in the store and can only be called by a task program by provoking a program interruption (for example by executing a privileged instruction). This will transfer control to the monitor, with the interrupt system disabled. The monitor is now able to perform any sequence of instructions as an indivisible entity, before it reenables the interrupt system.

In our system, all semaphores are implemented as bits in a single storage word. The monitor can perform two primitive operations on the semaphores. The reservation procedure (called P by Dijkstra) examines a number of semaphores, selected by a mask, in parallel. If they are all zero, their values are changed to one, and a return is made to the calling program. If some of them are ones, the current task program is interrupted and another task program is started. When the interrupted program receives a new quantum of computing time, it repeats the call of the P procedure.

The releasing procedure (called V) sets a number of semaphores to zero, and starts another task program. The transfer of control is necessary to prevent a task program from monopolizing a facility. Most of the programs perform cyclic reservations of the same facility in the following way:

```
Program A:  P(semaphore);
            critical section;
            comment: common facility reserved by A;
            V(semaphore);
            remainder of cycle;
            goto Program A;
```

At Pulawy, the probability of program A being interrupted in the remainder of the cycle before the next reservation is roughly equal to the execution time of about 100 instructions divided by the time quantum, i.e. 500 μsec/2.5 msec = 1/5. Thus program switching on the V function is vital for ensuring that the programs receive access to common facilities on equal terms.

In our system 13 semaphores are associated with common data tables, procedures, and input/output devices.

Two semaphores prevent the pulse and flow integration programs from updating the tables of integrated data, while they are used by the balance evaluation program.

To avoid a duplication of code, a number of procedures are shared by all task programs. They perform the control typewriter input/output and the input and conversion of analog values to proper engineering units. A shared procedure executes a normal P function on entry, and a modified V operation on exit. This V function ensures that the release of the procedure and the return jump are made an indivisible entity.

The remainder of the semaphores are associated with the log typewriters, the strip printers, and the paper tape punch.

8 Size and Performance

The time-sharing monitor and the process control programs for Pulawy were designed, programmed, and tested in 18 man-months. The size of the programs and the data tables are as follows:

	Words
Monitor	410
Common procedures	940
Operator control program	400
Pulse integration program	45
Flow Integration program	45
Balance evaluation program	415
Log program 1	55
Log program 2	55
Alarm scan program	110
Trend log program	25
Self-check program	215
Data description tables	1000
Data integration tables	300
Total system	4015

The real-time performance of the multiprogramming system has been evaluated by measuring the execution times obtained by sequential and time-shared execution of the task programs. In the sequential run-mode, the computer executes one task program at a time. In the time-sharing mode, all task programs were executed simultaneously to obtain worst-case figures.

	Sequential execution (seconds)	Time-shared execution (seconds)
Pulse integration program	< 1	< 1
Flow integration program	9	21
Alarm scan program	13	32
Log program 2	94	105
Log program 1	120	128
Balance evaluation program	147	153
Operator control program	infinite	infinite
Self-check program	infinite	infinite

The log and balance evaluation programs are mainly limited by the speed of the typewriters. The multiprogramming system makes it possible to run these at 90–96 percent of their maximum speed.

The bottleneck of the system is the analog/digital converter. At present, this device is shared in a sequential maner among the flow, alarm, and log programs. The scanning rate of flows and alarms thus drops to 41–43 percent of the maximum speed.

In a system with a bigger internal store, this could have been improved by introducing another task program that would scan the analog variables and store them in a table, say, every five minutes. The other task programs would then reference this table instead of repeating the analog measurements.

Acknowledgements

The design of the time-sharing monitor for Pulawy is the work of Peter Kraft and the author. Later, we were joined by Karoly Simonyi, Jr., who contributed valuable ideas to the project and did the programming along with Peter Kraft. We are indebted to John Saietz of Haldor Topsøe for his continuous support in the specification of the process control tasks.

References

Brinch Hansen, P. 1967. The logical structure of the RC 4000 computer. *BIT* 7, 3, 191–199.

Dijkstra, E.W. 1965. Cooperating sequential processes. Technological University, Eindhoven, The Netherlands, (September).