QUAESTIONES INFORMATICAE

Vol. 1 No. 1

June, 1979

Quaestiones Informaticae

An official publication of the Computer Society of South Africa
'n Amptelike tydskrif van die Rekenaarvereeniging van Suid-Afrika

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R2	$3	£1.50
Institutions	R4	$6	£3.00

Quaestiones Informaticae is prepared for publication by SYSTEMS PUBLISHERS (PTY) LTD for the Computer Society of South Africa.

A Hardware-Based Real-Time Operating System

M. G. Rodd

Dept. of Electrical Engineering, University of the Witwatersrand, Johannesburg, South Africa

Abstract

The efficient use of multiprogrammed industrial control computers is largely a function of the relationship between hardware and software A shift in this relationship is desirable, since multiprogrammed computers typically spend a large proportion of computing time in handling their own organization This situation is compounded in many time-critical industrial process-control applications

This paper proposes that a possible solution lies in the adoption of a hardware-based real-time operating system The system consists of a microcontroller working in close relationship with a conventional minicomputer To retain a high degree of flexibility, the microcontroller makes use of microprogrammable, bipolar, bit-slice microprocessor elements In essense, the unit executes the principal functions of a real-time operating system, acts as a pre-processor for all incoming requests, and ensures a high rate of task switching

The system has been applied in a series of real-time experimental configurations These were controlled successively by the conventional, software-implemented approach, and by the proposed system The respective performances were evaluated The new strategy is shown to result in a better and more economical industrial controller

1. Introduction

The introduction of the electronic computer has had an immense impact on society In almost every aspect of life, this product of man's power of innovation is exerting an influence One particular area in which the computer's ability to perform high-speed, repetitive tasks is used to an ever-increasing extent, is the industrial sector In an era of inevitable expansion of industrial capacity, industrialists look to the computer to aid them in achieving their goals of higher production and more constructive use of labour The demands made on the controlling computers have necessarily increased, from the execution of relatively simple sequential tasks, to an extent where total control of complex processes is invested in the computer

As the demands increase, the capabilities of the computers must increase likewise While dramatic technological advances have resulted in faster, more reliable, machines there is, seemingly, a limit to what a single computer is capable of performing The solutions offered have been many — multiply the complexity of the computer, add hardware, develop more sophisticated software, introduce additional, parallel processors

The majority of such complex systems have been produced by means of ad-hoc design procedures, based on experience and intuition This paper shows how a simplistic analysis of the performance of a single processor can reveal the vital criteria to be considered when defining a computer structure for a specific industrial application Based on this analysis, a system is proposed which provides for a highly efficient industrial controller [2] The heart of the structure is a microcontroller, executing control over a minicomputer and acting both as a partial pre-processor and as a hardwired operating system

2.1 Organizational Problems of Process-Control Computers

In broad terms, a computer applied in a real-time process-control situation must be capable of performing the following functions

(i) The scheduling of a variety of tasks in such a way as to effect overall control of the process ("Task" is taken to imply a sequence of computer instructions which perform a predetermined system function [1] Each task will have a certain time which it takes to complete and will, normally, require execution at certain prescribed intervals

(ii) The initiation of a specific task, indicated (or "requested") by any of the following occurrences

  (a) A signal derived from the plant

  (b) A request from another task currently in execution

  (c) A signal from one of the computer system's own external or internal peripherals (e g an event-timer or a real-time clock)

(iii) The temporary suspension of a task currently in execution, following a request for a task with a higher priority in the predetermined, pre-emptive priority hierarchy

Reviewing the above demands on such a system, one is struck by the fact that the problem which confronts the computer systems designer is by no means unique The situation is typical of many systems, having limited resources while being subjected to demands for service at varying intervals

Typical of such a system is a telephone exchange An Exchange might have thousands of subscribers, but can link only relatively few of them at any one time The requests for connections occur at some statistically evaluatable rate, and the lengths of calls (or holding times) may also be statistically predicted Such systems have been studied for many decades, and the results obtained can be of great value to the systems analyst in many other fields In the case under discussion in this paper, it is a single-server model which is appropriate, i e , only one request can be honoured at any one time, with all other requests occurring during that time being placed in a waiting queue [3]

The assumptions made are as follows

(i) That the incoming requests are characterised by a Poisson distribution, with an average request rate of K (per time-span)

(ii) That the distribution of task lengths may be characterised by a negative exponential function of the form

$$f(t) = \frac{1}{h} e^{-t/h}$$

where h is the mean execution time of the tasks, and t is the time variable

It may be shown (4), that the resulting mean queue length, L, is given

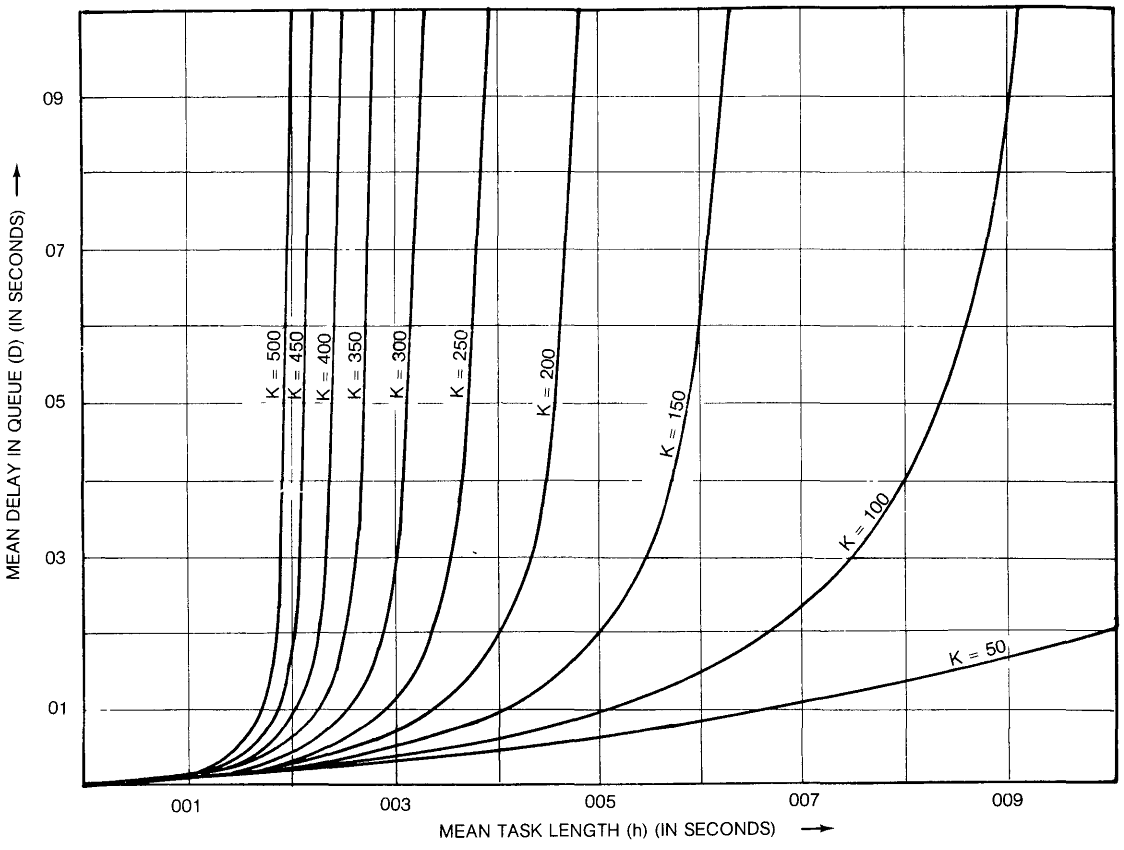
$$L = \frac{Kh}{1-Kh} \qquad (1)$$

and the mean delay, D, of tasks in the queue is given by

$$D = \frac{h}{1-Kh} \qquad (2)$$

Fig 1 plots expression (2) In the most basic terms, this gives the key to the performance of an industrial control computer The model is obviously highly simplified, as problems such as priority are not considered (Ref [4] shows how these can be included) The analysis does, however, provide a firm basis for viewing the requirements of a computer system performing such functions

This paper was presented at the SACAC symposium on Real-time Software for Industrial Applications in Pretoria on 29-30 November, 1978



K is the average Task request rate (per second)

2.2 Organizational Strategies

The criterion of acceptability of any control system is that it must be capable of performing the required function[1] In terms of Fig 1, this may be interpreted as requiring that the execution of every task, when requested, should occur within a permissible limit (any delay would, of course, arise from the queue which may develop)

If the characteristics of a process under review are such that non-permissible delays will occur if a conventional single computer is used, how can the system be structured to produce an acceptable situation? Fig 1 indicates the way This shows that, to reduce the average delay time experienced for a specified number of tasks, only two parameters can be varied These are

(i) The average task request rate, and
(ii) The average task execution time

To achieve a reduction in either or both of these parameters, a variety of techniques may be employed Some of the more obvious are outlined below

(a) **Reduction of the Average Request Rate**
Two solutions are immediately apparent here Firstly, pre-processing may be employed to sift through demands from the system (i e , requests for attention, alarms, etc), and to select those which are absolutely vital for sending on to the main computer Secondly, processing power may be distributed among a hierarchy of processors Thus, the request rate (and therefore the load) to any one machine will be reduced

(b) **Reduction of the Average Execution Time**
The obvious indication is that actual run times of tasks should be kept to an absolute minimum This implies, firstly, that the processor should be made as fast as possible, and secondly, that programs should be optimally coded for minimum run-time There are, however, two important factors which should also be considered here Firstly, it is common knowledge that any multi-programmed computer spends much of its available computing time in executing internal operating system functions and virtually "trying to decide what to do next" Such operating system functions can be considered as tasks and should therefore be kept to a minimum Also, the question of swop-time (the time taken to switch execution from one task to another) must be taken into account Since the bringing into execution of any task will require a task-swop, this time-length may be included in the average task execution time Thus, reduction of the task-swop-time will effectively reduce average execution time

3.1 A Possible Solution

The factors discussed above can be considered collectively, and a multi-faceted approach adopted It is not intended to attempt a totally new architectural approach to computer design The area of immediate interest lies in the modification of existing minicomputer systems to produce more capable and efficient systems

It is proposed that the solution lies in the supervision of the functions of a minicomputer by means of a separate, but closely-linked, hardware unit The functions of the unit are

(i) To execute the primary executive functions of a real-time operating system.
(ii) To pre-process all incoming requests from the process and from the peripherals, as well as those generated by tasks currently in execution in the computer
(iii) To supervise task-swopping within the minicomputer

As well as meeting these requirements, it is essential that

(a) The unit may be interfaced to the minicomputer **without** modifications to the existing hardware
(b) It should be economically effective
(c) Although a hardware unit, the structure must retain a high degree of flexibility

The block diagram of a suitable structure is shown in Fig 2 The hardware unit comprises a hardwired operating system together with a local memory, an interrupt controller, and interfacing to the computer The interfacing to the minicomputer utilizes the computer's input/output bus, and requires no change in the minicomputer's existing hardware

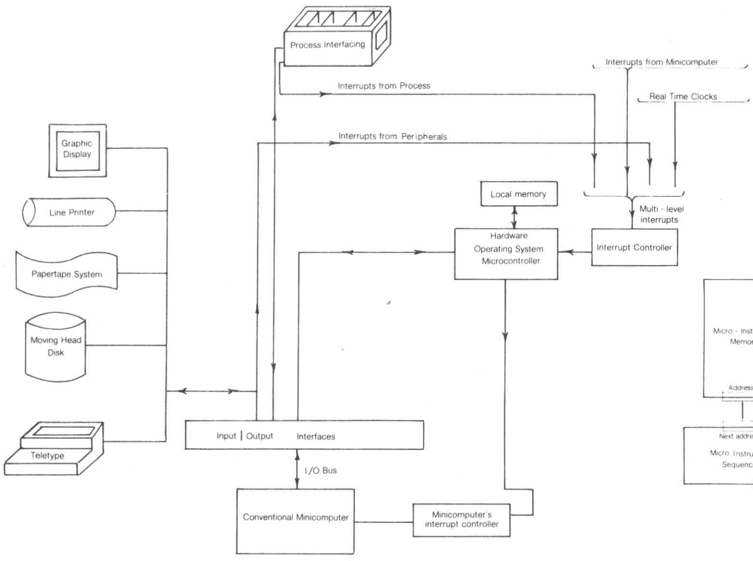


FIGURE 2 : SYSTEM BLOCK DIAGRAM

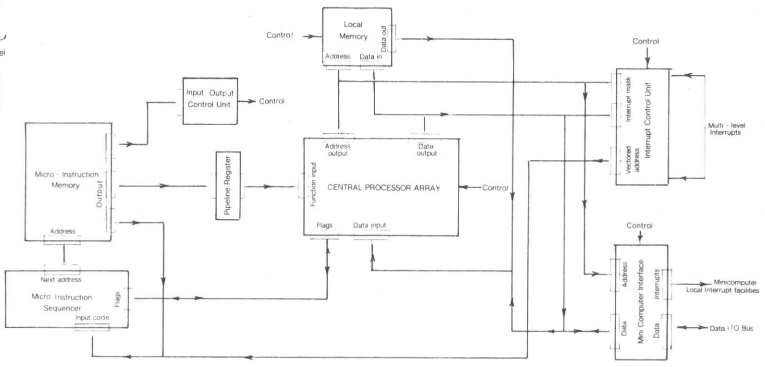


FIGURE 3 : MICROCONTROLLER BLOCK DIAGRAM

PARAMETER	COMPUTER I WITH MICRO-CONTROLLER	COMPUTER I WITH KERNEL IN-CORE REAL-TIME OPERATING SYSTEM	COMPUTER II WITH IN-CORE, REAL-TIME OPERATING SYSTEM (16 Bit)	COMPUTER III WITH REAL-TIME OPERATING SYSTEM (32 Bit)
Time to recognise the presence of an interrupt	1,8	7,2	10,6	6
Time to accept an interrupt and initiate the task requested. (I)	171	408,6	261,5	2106
Time to effect a task-swop. Note (i)	165,6	408,6	200	2100
Time to initiate a task called by another task. Note (i)	171	471,6	200	2300
Time to react to an interrupt generated by a peripheral. Note (i)	1,8	7,2	10,6	6
Time to initiate an input/output operation. (i)	171	408,6	±300 Note (iii)	1800
Time to terminate an input/output operation.	171	408,6	±300 Note (iii)	1700
Longest non-interruptable time. Note (ii)	7,7	495	Not available	97
Core required to implement a basic operating system (in actual locations.)	76	508	2805	±10K

3.2 A Practical Implementation

From the required system functions, it is clear that the hardware unit (referred to below as a "microcontroller"), is in essence, a high-speed computer with somewhat limited and specialized functions The high speed is essential, since it must be capable of rapid reactions to incoming requests, and must be able to control the minicomputer as effectively as possible It was estimated [2], that the microcontroller should be one order of magnitude faster than the minicomputer in processing speed This indicated a desired cycle-time of the order of 150-200 nano-seconds, which immediately ruled out conventional mircroprocessors The need for flexibility pointed to a microprogrammable unit To meet both these requirements, bipolar bit-slice microprocessor elements were used (see Fig 3) The central processor array is a 16-bit wide system, for convenient data exchange with the host computer The micro-instruction word is essentially vertical in structure, with a 24-bit width Two pages of micro-instruction memory, each with 512 words, are included Information relevant to each task (sometimes referred to as a Process Control Block) is held in a local, high-speed bipolar memory, consisting of 1 K x 16 bits RAM The interrupt control unit presently comprises 64 levels of interrupt, fully vectored, with programmable masking The minicomputer interface controls data flow, and routes signals to the minicomputer's interrupt system All communications with the minicomputer are effected via this interface, which also controls the necessary synchronization during data transfer To effect data transfer, approximately 70 locations are required within the minicomputer's memory, to act as a buffer store and interrupt catcher

3.3 Operation

The real-time operating system is effected by microprogrammes At the present time, its functions are limited to those of the executive of a conventional, in-core, real-time operating system As such it may be regarded as a Kernel operating system Only limited file-handling has been implemented The primary functions of the operating system are

(i) Scheduling of multiple tasks The tasks may be either real-time dependent and priority-based, or non-time-critical background tasks A limited degree of dynamic rescheduling is permitted
(ii) Control of multiple, vectored interrupts
(iii) Control of inter-task communication and synchronization, including buffer-passing, as well as communication between tasks and the executive
(iv) Supervision of all input/output operations

In simple terms, the microcontroller determines which task should be executed at any one time It examines all incoming interrupts, and uses a single-queue, priority-based, pre-emptive scheduling algorithm to determine which tasks should be swopped-in or-out Input/output requirements in the minicomputer are dealt with by regarding their driver routines as tasks, on the same basis as user tasks

3.4 System Evaluation

The system outlined above has been implemented in conjunction with a minicomputer, applied in a variety of real-time control situations [2] The results obtained have been compared with results measured using conventional techniques The same minicomputer system was used, under the control of an in-core, real-time operating system In addition, comparisons were made with other minicomputer systems operating under their current real-time operating systems Some of the results obtained are shown in Table 1

TAKE IN TABLE 1

TABLE 1 **Comparison of Some Key Real-Time Operating System** Parameters

NOTES A All times are in microseconds
B (i) Assumes the requested task is in memory, and has a higher priority than the current task
(ii) Only considers tasks which form part of the operating system, and excludes any user tasks
(iii) Dependent on the peripherals used

A detailed comparative study was also made of the Utilization Factor (UF), a factor which indicates what proportion of the total available computing time is actually used for valuable work (i e the processing of user tasks) In a typical, fast on-line control application, it was found that a UF of < 0,08 could be expected, i e less than 8% of total available computing time was actually devoted to the execution of user tasks The rest of the time was taken up by scheduling, task-swopping, interrupt handling, etc Under the supervision of the microcontroller, a UF of up to 0,7 (i e 70%) could be achieved in the identical situation

4. Conclusion

It has been described how a simple statistical analysis of a process-control computer system can indicate ways of ensuring efficient organization It is clear from the analysis that the provision of bigger and better facilities will not necessarily result in improved systems In order to improve the system and to provide for a high degree of utilization of the facilities available, there are certain clear-cut factors which must be considered These factors indicate that an unconventional, integral hardware/software approach to the management of industrial control computers would ensure highly efficient operation Such possibilities become feasible in view of current technological developments, specifically with the introduction of ultra high-speed microcomputer components

Indeed, the work described in this paper points to an important avenue, in which much research effort should be concentrated, that of the supervision of large, powerful mainframe computers by simple, low-cost controllers It is clear that any efficient system implies efficient use of resources Thus, simple, logical tasks such as scheduling and interrupt handling are not necessarily the forte of powerful, 'number-crunching' computers, and the delegation of such tasks to these machines is a misuse of potential power The solution lies in surrounding the mainframe machines with small, limited-capability computers (i e microprocessors), which will supervise and assist the central machines in using their potential to the utmost To quote Maurice Wilkes, "A computer operating system may be regarded as — and indeed is — a control system There would thus appear to be scope for implementing an operating system by means of a number of interconnected micro-processors dedicated to the purpose " [5]

Acknowledgement

The author wishes to express his appreciation to the University of Cape Town for assistance in the production of this paper

References

1 BARRON, D W (1971) *Computer Opeiating Systems,* Chapman and Hall, London

2 RODD, M G (1976) *Organization of Industrial Control Computers,* Ph D Thesis, University of Cape Town

3 RUBIN, M and HALLER, C E (1966) *Communication Switching Systems,* Rheinhold Publishing Corp , New York , 246-271

4 SYSKI, R (1960) *Congestion Theory in Telephone Systems,* Oliver and Boyd, London 298-340

5 WILKES, M V (1977) The Application of Microprocessors to Main Frame Design, in *Proc IERI Conference on "Computer Systems & Technology",* Brighton, U K

Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles or articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be in double-spaced typing on one side only of Henderson or Prof. M. H. Williams at

Rhodes University
Grahamstown 6140
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9,** 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.

Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

Quaestiones Informaticae

Contents/Inhoud

Spectrum Offset