Proposal:

A system of clocks for Multics.

This paper ties together a collection of specific proposals that have been suggested by many people concerning clocks on the 645 Multics system. Contributors have included G. Oliver and G. Fuchs, D. Dehn (CE), J. Ossanna + V. Vyssotsky (BTL), F.J. Corbató + E.L. Glaser (MAC).

Objectives

The hardware clocks available on Multics should provide the following services:

1. Calendar clock: From this clock one may determine calendar date and standard time. It should be accurate to within ±1 sec of local standard time. It should be easy for an operator to set this clock to within ±1 sec of local standard time and it should not drift more than ±1 sec/week. The value of this clock is read by mechanical interaction by an operator; it is independently powered and is read only reset in the event of power failure.

Real-Time

2. Interval measurements: For purposes of time accounting, statistical monitoring, and program speed evaluation, it must be possible to measure time intervals as small as 0.5 ms, with an accuracy of ± 0.1 ms. It would be very desirable, although admittedly expensive, to push this accuracy to ± 1 μsec. One constraint which is essential to note is that one program (or active device) must be informed of the beginning of an interval, a second the end of the interval. It follows that a common system clock accessible to all active devices is necessary for real-time interval measurements. It is also important that the identical interval measurement technique be available for long intervals (e.g. a week or a month) since at the beginning of an interval one may not know whether it will be short or long.

3. Real-Time Interval Interrupt: Many Multics supervisory processes will have need for interrupt signals based on real-time intervals or on time of day. Since a process issuing this will service is not necessarily be running when the interrupt signal occurs, the interrupt signal is best handled as a single system interrupt rather than a fault per process. Expected Time Intervals will run from a few ms. to several hours; the interrupt signal should occur at the system within ± 0.1 ms of the desired time.

4. Process ~~time interval measurements~~ Usage measurement and Interrupt: The system scheduler needs a technique for ~~getting control of a processor from a long~~ switching running process ~~And for a process which is running too long to a different processor in the basis of time used.~~ to automatically bumping long-running users from their processor. ~~Time intervals for this application may typically range from a few milliseconds to a few hundred ms, and should be given priority within a range for terms of process. The time interval used may be real time, but can be based on many cycles used~~ A per-process countdown register is initialized in each processor is initialized.

5. ~~Add~~ Common Time unit: All system accounting based on real time should be able to use a common time unit. Microseconds are suggested to be ~~this~~ small enough or ~~five~~ one should be necessary for the class of system.

6. Reliability: ~~The Multics supervisor will depend on the system of checks in order to operate~~ If any of the basic check functions described above is not working, the Multics supervisor will not operate correctly. Therefore, reliability techniques such as duplication and simple reconfiguration are essential.

~~and the same~~ warning techniques for abort and long intervals.

Discussion:

The first two objectives, (age calendar clock, and setting interval timer) can be readily met by a single hardware register. This register increments once per μsec, and will require 100 years to overflow, and is (indirectly (if reading only)) accessible by any processor. Since a 52-bit register is required for a 100-year count, (see appendix) double-word integer type must be used for all times and intervals obtained from the clock. The different problem is where to put this register to ensure indirect accessibility by any processor. Two possibilities appear feasible for the location of a microsecond calendar clock:

1. In the memory controller. A clock here could be either wired in as a specific memory location (note that this approach makes interface any memory controller awkward)) or as a special register similar to the memory file protect register which is read by a special processor instruction. The objective of reliability is easily met by building one clock register into each memory controller.

2. ~~In the GI&C~~ As an I/O device working through the GI&C. In this arrangement, a calendar clock register must ~~be read through~~ be placed either in a GI&C channel adapter or in a free-standing box working into a GI&C channel adapter. Periodically the contents of the calendar clock register must be read into core memory via a direct channel. Updating the memory cell every time the calendar clock changes (once per minute) must tax the capacity of both the GI&C and of the channel memory module(s). ~~The accuracy of the clock must be compromised with the data rate~~ A better source of a periodic signal to update the memory cell is the change of a selected bit of the ~~clock~~ calendar clock. Thus if the 8th bit from the right is chosen, the clock will be read into core every 64 msec, using about ~~1%~~ 2% of the memory cycles of one memory module, and about 5% of the capacity of one GI&C.

Of these two possible locations for a calendar clock, the memory controller seems the more desirable; unfortunately its design has been frozen too long to incorporate such features. Placing the GI&C location, while producing a less desirable clock, has the virtue that the calendar clock "plugs in" to an interface designed to work with virtually any hardware device.

The third objective, a wake-up clock, can be met by providing a second 32-bit count-down register which is synchronized with the calendar clock. When the register reaches zero, it ~~signals~~ triggers a standard interrupt sequence. ~~Although~~ the wake-up clock may be loaded at any time by any processor working for the operating system. ~~And~~

The processor usage meter is a special function which gets involved in time-accounting problems. The desire to obtain a load-independent measure of processor usage has prompted the suggestion that this ~~interval timer~~ register should count memory cycles used by the processor, rather than real time. Assuming that the DIS instruction is disabled, this approach seems very appealing.

[illegible]

Operator information

The calendar clock will occasionally need to be ~~set~~ adjusted by ~~the~~ an operator. One can invent elaborate mechanical or electrical aids to ~~the job~~ figure out the # of years since Jan 1, 1900, ~~[illegible]~~ ~~[illegible] this job easy~~, ~~but it appears that the following techniques~~ but it seems easier to use the capacity of the main computer as a crutch for such an infrequent operation. A small table of clock settings can be used to obtain a ball-park estimate (within a few days) ~~for~~ If the clock is far off, ~~an initial setting~~. This value can be placed in the clock ~~and the system bringing up~~ to allow the system to run; ~~the system~~ A simple program can then perform a precise computation ~~under control of~~ for the operator. ~~He~~ He gives a date and time; or receives an octal number ~~[illegible]~~ or a reply. He then keys the ~~resulting~~ octal number into the clock toggles, waits for the "exact" time, and presses a "read-in" button. Another approach to ~~clock~~ fine adjustment of the clock is to allow fine tuning of the crystal oscillator ~~which~~ on which the clock is based. It ~~should~~ must be emphasized that clock adjustment should be a rare operation performed only following a power failure.

Proposal

This section presents two

For the design of the above description, ~~two slightly different~~

proposals for implementation of a system of clocks, ~~with the same~~

~~proposals differ~~ one proposal putting the calendar clock in the Memory

Controller, the second in the CPU. ~~Both of these proposals~~ We

begin by describing ~~the~~ certain specifications which are common to

both proposals.

First, there is a device known as a "calendar clock":

this is a free-standing box, ~~independently powered~~ powered directly from the ~~standard~~ 3-wire Teletype M-G set. It contains

a ~~72~~ 52-bit ~~register~~ counter register driven by a ~~one~~ 1 mc. crystal

oscillator. Thirty ~~two~~ three toggle switches and a read-in button allow

an operator to set the left-most ~~32~~ 33-bits of the counter register to any

desired initial value. The remaining 19 bits are set to zero by the read-in button.

The 1 mc. crystal oscillator frequency, protected from accidental probes,

may be adjusted by a Product Service Engineer to keep the ~~frequency to within~~ the

Without adjustment of the crystal oscillator, the clock should drift less than 1 sec/week. ~~[illegible]~~

~~Drift~~ ~~It should be within~~ ~~and the~~

calendar clock within 1 sec of standard time. It must be possible to synchronize the crystal oscillator with an external 1 mc. frequency standard if such a standard is available.

The cable interface to the calendar clock consists of 53 lines, ~~one for each~~ ~~consisting of the 52~~ carrying the values of the 52 bit register, and a red flag cell ~~register~~.

A single cell in the ~~clock~~ calendar clock acts as a "red flag" to indicate that power has failed and that the calendar register may be in error. This "red flag" cell is set on ~~any power coming~~ whenever power comes up on the calendar clock; it can be turned off by ~~an~~ ~~on button~~ an operator's button. An indicator lamp is ~~a~~ ~~[illegible]~~ ~~the~~ actuated by the "red flag" cell.

~~[illegible]~~

③

There must also be at least 1 signalling line which is up whenever the calendar clock register lines are changing.

The cable interface to the calendar clock contains lines for the 52-bit calendar clock register and for the "red flag" call, △ Cable delay restrictions may be relaxed by the following convenient coin: ~~The exact time of arrival of the calendar clock value is not important; the interval between successive clock values is important.~~ A delay of up to 0.5 sec is tolerable, but the delay must be constant to within a microsecond or less. (Otherwise, hardware tolerances will be made tighter ?); then we need software tolerances.)

The above proposed calendar clock is assumed in both of the two

following alternative proposals.

Proposal I (Memory Controller)

The "Memory File Protect Register" (64 bits) is removed from the Memory Controller, In

its place are two 32-bit registers, the calendar clock register and

the Calendar Interval register. The contents of the calendar clock register

are set via a cable interface from the calendar clock described above.

A "set flag" call is set from the corresponding call in the calendar clock.

The Calendar Interval register simply counts down under control of

the signal line from the calendar clock, once/microsecond. When the

Calendar Interval Register power reaches zero, it generates an interrupt signal

which may be directed by switch to any of the 32 memory interrupt calls

in that memory controller, ~~via any backplane~~.

1 ···· 52
0 ——— 71

these two registers are accessible to any 645 processor by two special instructions. ~~Since the memory file protect register setting and reading instructions have not been~~ the instruction "Read Memory File Protect Register" is renamed "Read Calendar Clock Register"; it operates as follows: the contents of the calendar clock register of the addressed memory controller are placed in the AQ register bits 20-71. If the "real flag" in the memory controller call is set on, the requesting processor is not on. This instruction may be executed in Slave Mode.

The Instruction "Set Memory File Protect Register" is renamed "Set Calendar Interval Register". It operates as follows: the contents of the AQ register bits 20-71 are placed into the Calendar Interval Register of the addressed Memory Controller. This instruction may only be executed in ~~Slave~~ Master Mode; it causes an illegal procedure fault in Slave mode. A writing Interrupt from the Calendar Interval Register is not reset by this Instruction.

Proposal IV (GI&C)

"Calendar Clock Adapter"

A special direct channel adapter is placed in the GI&C. This adapter contains two 52 bit registers, a calendar clock register and a calendar Interval register. The contents of the calendar clock registers are set via a cable interface from the Calendar Clock described above. A "ready flag" cell in each of the corresponding cell in the calendar clock. The Calendar Interval Register simply counts down under control of the signal line from the calendar clock, one /μs increment. When the calendar Interval Register passes zero, ~~it generates an Interrupt signal~~ a status word is generated which is passed to the appropriate GI&C status channel.

~~the calabo clock~~

A word in top of the calabo clock register controls the

routing of its contents to a core memory location. This top may be set by switch

~~to~~ any of the right most 11 bits of the register. Whenever that bit

changes, the calabo clock contents are sent to core memory via a

~~the "Bad Flag" cell is also sent to core memory in bit position 0.~~

direct channel. The core memory location is ~~sent~~ chosen by the

operating system ~~at initialization time when it issues a~~ instruction a connect for the calabo clock channel.

The calabo interval register may be set from core memory at any time

by the operating system by ~~issuing~~ instructing a connect for the calabo clock channel.