PROPOSAL: A System of Clocks for Multics.

TO: F. J. Corbató

FROM: J. H. Saltzer

DATE: October 19, 1965

This paper ties together a specific proposal out of a collection of ideas which have been suggested by many persons concerning clocks on 645 Multics. Contributors have included G. Oliver, G. Futas, and D. Dahm (GE); J. Ossanna and V. Vyssotsky (BTL); and F. J. Corbató and E. L. Glaser (MAC). Following a discussion of the objectives of the system of clocks, two proposals for hardware implementation of the clocks will be presented.

Objectives

The hardware clocks on Multics must provide four basic services:

Calendar Clock

Real-Time Interval Measurements

Real-Time Interval Interrupts (wake-up clock)

Processor Usage Meter and Interrupt

In detail, these services are as follows:

1. Calendar Clock: From this clock one may determine calendar date and standard time. It should be easy for an operator to set this clock to within 0.1 second of local standard time and

it should ~~not drift from~~ its initial setting ~~by more than 1 second per week.~~ The indicated accuracy is approximately what may be achieved by a crystal oscillator without elaborate technology; this accuracy provides the user with real time experiments (and data links between Multics systems) with a useful reference source without attempting to be a secondary time standard. The value of this clock is set by mechanical intervention by an operator. It is powered independently of the computer so that the clock continues to run when the computer is shut down. Resetting the clock should be an unusual event, probably connected with a power failure.

2. Real-Time 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 milliseconds, with an accuracy of ±0.1 ms. It would be very desirable, although admittedly expensive, to push this accuracy to ±1 microsecond. One constraint which is essential is that one processor may mark the beginning of an interval and 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 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 (wake-up clock): Many Multics supervisory, monitoring, and user processes must have interrupt signals based on real-time intervals. For example, a monitoring process may wish to sample the number of logged in users every 10 minutes; a traffic control program may wish to change its strategy at 4:30 p.m. Expected time intervals will run from a few milliseconds to several days.

4. Processor Usage Meter and Interrupt: The system scheduler must have a technique available for automatically switching between processes on the basis of processor usage. A countdown register in each processor is indicated. The 635 processor interval timer is adequate.

In providing these four services three considerations should be emphasized because of their fundamental importance to the operation of Multics.

1. Common time unit: All system accounting based on real time should be able to use a common time unit. Microseconds ~~are suggested to~~ should be as small a unit as ~~should be~~ necessary for this class of system.

2. Reliability: If any of the basic clock functions described above is not working, the Multics supervisor will not operate and the Multics system is down. Therefore reliability ~~techniques such as~~ duplication and simple reconfiguration ~~are~~ is essential. In addition, there should be little opportunity or reason for the calendar clock to appear to be operating but with an incorrect setting. It follows that the calendar

clock must run without routine human intervention to avoid human error.

3. Simple usage: The clocks will be used very frequently both by the Multics supervisor and by Multics users; a clean simple to use design which requires little special programming is mandatory.

Discussion.

The first two objectives of a calendar clock and a real time interval timer can be neatly met by a single hardware register of 52 bits containing the number of microseconds since January 1, 1900. This register increments once per microsecond, requires 140 years to overflow, and is immediately accessible for reading (but not writing) by any processor. Obviously, double-word integer logic must be used for all times and intervals obtained from this clock. The difficult problem is where to put this register to insure immediate 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 may make interlace among memory controllers 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. As an I/O device working through the GIOC. In this arrangement, a calendar clock register would be placed either in a GIOC channel adapter or in a free-standing box working into a GIOC channel adapter. Periodically the contents of the calendar clock register would be read into core memory via a direct channel. Updating the memory cell every time the calendar clock register changes (once per microsecond) would tax the capacity of both the GIOC and of the addressed memory module. A better source of a periodic signal to update the memory cell is the change of a selected bit of the calendar clock register. Thus if the sixth bit from the right is chosen, the clock will read into core memory every 64 microseconds, using about 2% of the memory cycles of one memory module, and about 5% of the capacity of one GIOC.

Of these two possible locations for a calendar clock register, the memory controller ~~seems~~ is by far the more desirable. The GIOC location, while producing a less desirable clock and using up GIOC capacity, has the virtue that the calendar clock "plugs in" to an interface designed to mesh with virtually any hardware device.

The third objective, a wake-up clock, can be met by providing a second 52-bit count-down register which is synchronized with the calendar clock. When the register reaches zero, it triggers a standard interrupt sequence. The wake-up clock may be loaded at any time by any processor working for the operating system. This register would be located wherever the calendar clock register is.

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 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.

Operator Intervention.

The calendar clock will occasionally need to be adjusted by an operator. One can invent elaborate mechanical or electrical aids to figure out the number of microseconds since January 1, 1900, but it seems easier to use the capacity of the main computer as a crutch for such an infrequent operation. If the clock is far off, a small table of clock settings can be used to obtain a ball-park estimate (within a few days) This value can be placed in the clock to allow the system to run; a simple program then performs a precise computation for the operator. He gives a date and time as input and receives an octal number as a reply. He then keys the octal number into the clock toggles, waits for the "exact" time, and presses a read-in button. Another approach to fine adjustment of the clock is to allow fine tuning of the crystal oscillator by capacitive loading. It must be emphasized that clock adjustment should be a rare operation performed only following a power failure.

Proposal.

In the light of the above discussion, this section presents ~~two~~ proposal~~s~~ for implementation of a system of clocks, ~~one~~

proposal putting the calendar clock register in the Memory Controller, the second in the GIOC. We begin by describing certain specifications which are common to both proposals.

First, there is a device known as a "Calendar Clock." This is a free-standing box, powered ~~directly from~~ the system motor generator set. It contains a 52 bit counter register driven by a 1 mc. crystal oscillator. Thirty-~~three~~ six toggle switches and a read-in button allow an operator to set the most significant ~~33~~ 36 bits of the counter register to any desired initial value. The remaining ~~19~~ 16 bits are set to zero by the read-in button. The read-in button is protected from accidental bumps. The crystal oscillator frequency may be adjusted if necessary by a Product Service Engineer to keep the Calender Clock within 1 second of standard time, without adjustment of the crystal oscillator, the clock should ~~drift less than~~ never gain or lose 1 second ~~per week~~. It must be possible to synchronize the crystal oscillator with an external 1 mc. frequency standard if such a standard is available.

A single cell in the 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 whenever power comes up on the calendar clock; it can be turned off only by an operator's button. An indicator lamp is activated by the "red flag" cell.

The cable interface of the calendar clock contains lines for the 52-bit calendar clock register and for the "red flag" cell. There must also be ~~at least~~ one signalling line which ~~is active whenever the calendar clock register lines are changing~~ carries a pulse every microsecond. Cable delay restrictions may be relaxed by the following consideration:

A delay of up to 0.5 seconds is tolerable, but the delay must be constant to within a microsecond or less. (Obviously, hardware tolerances will be much tighter, these are only 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 52-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 "red flag" cell is set from the corresponding cell in the Calendar Clock.

The Calendar Interval Register simply counts down under control of ~~the~~ signal line from the Calendar Clock, once per microsecond. Whenever the Calendar Interval Register passes zero, it generates an interrupt signal which may be directed by switch to any of the 32 memory interrupt cells in that memory controller.

These two registers are accessible to any 645 processor by two special instructions. 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 "red flag" cell in the memory controller is on, the negative indicator in the processor is set 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 Master Mode; it causes an illegal procedure fault in slave mode. A waiting interrupt from the Calendar Interval Register is not reset by this instruction.

Proposal II (GIOC)

A special direct channel adapter (the "Calendar Clock Adapter") is placed in the GIOC. This adapter contains two 52 bit registers, a Calendar Clock Register, and a Calendar Interval Register. The contents of the Calendar Clock Register are set via a cable interface from the Calendar Clock described above. A "red flag" cell is set from the corresponding cell in the calendar clock. The Calendar Interval Register simply counts down under control of ~~the~~ signal line from the Calendar clock, once per microsecond. When the calendar Interval Register passes zero, a status word is generated which is passed to the appropriate GIOC status channel.

A movable tap on the Calendar Clock Register controls the sending of its contents to a core memory location. This tap may be set by switch to any of the right-most eleven bits of the register. Whenever the selected bit changes, the Calendar Clock contents are sent to core memory via a direct channel. The "red flag" cell is also sent to core memory in bit position 0. The core memory location is chosen by the operating system at

initialization time when it initiates a connect for the Calendar Clock Channel.

The Calendar Interval Register may be set from core memory at any time by the operating system by initiating a connect for the Calendar Clock Channel.

A typical ~~645~~ Multics system with 4 memory controllers would include two calendar clocks ~~Under~~ ~~when the system is~~ ~~reconfigured,~~ ~~One~~ ~~of~~ which are plugged into two ~~one~~ of the memory controllers. The calendar clock register and calendar interval register in the remaining memory controllers would be nonexistent, and ignored by the operating system. The operating system would use only one of ~~the~~ ~~clocks~~ ~~for~~ the two ~~timing~~ working calendar clocks ~~register~~, ~~treating~~ and treat the other as a backup. A reconfiguration of the system to remove a memory controller might require reconnection of one of the calendar clocks to a different memory controller.