DATE: September 22, 1983

TO: R and D Personnel

FROM: Jerry Kazin

SUBJECT: Software Interrupt Control Module Design Spec.

REFERENCE: PRIMOS Subroutines Guide (BREAK$ Function)
Software Interrupt Mechanism
PE-TI-879
Software Interrupt Control Module Proposal
PE-TI-1004
Software Interrupt Control Module Functional Spec.
PE-TI-1005
Specifications For The PRIMOS Condition Mechanism
PE-T-468

KEYWORDS: Software Interrupts, QUIT

ABSTRACT

Prior to Rev. 18, PRIMOS contained only one software interrupt known as the QUIT$ condition. This condition may be enabled/disabled by calling the BREAK$ module. At both Rev. 18 and Rev. 19 new software interrupts were added to the system. At Rev. 19 five new modules which allow users to enable/disable these new interrupts have been added to PRIMOS. They are SW$INT, SW$MKRCS, SW$ROOFF, SW$RAOF, and SW$ON.

This paper discusses both the internal details of these modules.

Please note that the information relating to the internal details of the software interrupt mechanism has been moved to Software Interrupt Mechanism PE-TI-879.

Table of Contents

1 INTRODUCTION.....3

2 SW$INT - THE SOFTWARE INTERRUPT CONTROL MODULE.....3

3 SW$MKRCS - TURNING ON SOFTWARE INTERRUPTS IN RING 0.....4

4 SW$ROOFF - TURNING OFF SOFTWARE INTERRUPTS IN RING 0.....5

5 SW$RAOF - TURNING OFF ALL SOFTWARE INTERRUPTS IN RING 3.....5

6 SW$ON - TURNING ON SOFTWARE INTERRUPTS IN RING 3.....6

7 MORE INFORMATION ON THE SOFTWARE INTERRUPT MECHANISM.....6

1 INTRODUCTION

This document provides a description of the internals of the software interrupt control modules, SW$INT, SW$MKRCS, SW$ROOFF, SW$RAOF, and SW$ON.

At Rev. 19.0 six software interrupt types have been defined for PRIMOS. They are the following:

1) CPU watchdog timer (CPU_TIMER$ condition),

2) Real Time watchdog timer (ALARM$ condition),

3) Phantom Logout Notification (PH_LOGO$),

4) Cross Process Signalling (CPS$),

5) Logout (LOGOUT$), and

6) Terminal QUIT (QUIT$).

CPS$ is an internal condition and will never be released to the general public.

At Rev. 19.3 a seventh interrupt type has been added.

7) IPC Message Waiting (IPC_MSG_WAITING$)

SW$INT and its associated mechanism allow a user to independently and selectively enable/disable any or all of these interrupt.

SW$MKRCS, SW$ROOFF, SW$RAOF, and SW$ON allow a user to enable/disable all of these interrupts at one time.

Before the creation of these modules, a user was able to turn off only one kind of software interrupt, QUIT. This was done via the BREAK$ module. For more information on this module consult PRIMOS Subroutines Guide (BREAK$ Function).

The following are only descriptions of the internal workings of the software interrupt control modules. The calling interfaces are defined in Software Interrupt Control Module Functional Spec. PE-TI-1005.

2 SW$INT - THE SOFTWARE INTERRUPT CONTROL MODULE

The software interrupt mechanism control module, SW$INT performs three basic functions; reading the present state of the interrupt types, turning on interrupt(s), and turning off interrupt(s). In addition, when enabling interrupts SW$INT invokes SW$ABT to determine if any software interrupts are pending and should be taken.

The following algorithm briefly describes the operation of SW$INT:

Determine the ring in which SW$INT is to operate.

Get the already defered status.

Determine if the key is valid and if a read and/or enable/disable is to occur.

Determine if the selected interrupt type(s) is valid.

Determine if enough space has been provided for SW$INT to write to if it is going to do a read.

Determine if the ring in which SW$INT is valid.

If everything is valid up till this point

Perform the designated read and/or enable/disable operation(s) for each selected interrupt type.

If enabling interrupt(s) and an interrupt has already been defered then call BCKUPB to insure that the call into ring 0 will be retryed after the defered interrupt has been signalled.

Else if enabling interrupt(s) and it is alright to invoke SW$ABT invoke it.

Return the already defered status.

All reading, enabling, or disabling is acheived through the use of either "and", "or", or "and with inverted target" operations. This is done for the sake of fast performance.

It should be noted that SW$INT set or clears the appropriate bits in the interrupt control words in PUDCOM for all interrupt types except terminal quits. This type is handled by BREAK$.

3 SW$MKRCS - TURNING ON SOFTWARE INTERRUPTS IN RING 0

SW$MKRCS enables the receipt of software interrupts in ring 0. The area of code in ring 0 which is enabled is known as a reverse critical section. (MKRCS stands for make a reverse critical section.)

Basically, SW$MKRCS is responsible for two things; enabling all software interrupts not presently enabled and returning a description of the interrupts which were already enabled at the time of the call to SW$MKRCS. (This description will later be used to end the reverse critical section. It must be possible start and end a reverse critical section in the same state with respect to software interrupts.)

The following algorithm oriefly describes the operation of SW$MKRCS:

Getting present software interrupt status

Inverting found status

Checking terminal_quit state

Enabling all presently off interrupts -

the result of steps 1, 2, and 3.

If any interrupts are found pending SW$ABT is called to

handle the interrupt

The enabling of interrupts is achieved by an "or" operation. This is done for maximum speed.

It should be noted that SW$MKRCS directly modifies the ring 0 quit counter, PUDCOM.ROQUIT, and does not call BREAK$. This is done for performance sake.

4 SW$ROOFF - TURNING OFF SOFTWARE INTERRUPTS IN RING 0

SW$ROOFF allows code in ring 0 to disable a selected set of software interrupts. It is used in conjuction with SW$MKRCS to bound a reverse critical section.

Basically, SW$ROOFF is responsible for one thing; turning off the selected software interrupts.

The following algorithm briefly describes the operation of SW$ROOFF:

Turn off the selected interrupts

Checking terminal_quit state

The enabling of interrupts is achieved by an "and with inverted target" operation. This is done for maximum speed.

It should be noted that SW$ROOFF directly modifies the ring 0 quit counter, PUDCOM.ROQUIT, and does not call BREAK$. This is done for performance sake.

5 SW$RAOF - TURNING OFF ALL SOFTWARE INTERRUPTS IN RING 3

SW$RAOF allows code in ring 3 to quickly disable all software interrupts. It is used to create a region of code that cannot be software interrupted. This region is known as a critical section.

Basically, SW$RAOF is responsible for two things; turning off all software interrupts not presently inhibited and returning a description of the software interrupts that were already off prior to the call to SW$RAOF. (This description will later be used to end the critical section. It must be possible start and end a critical section in the

same state with respect to software interrupts.)

The following algorithm briefly describes the operation of SW$RAOF:

Getting the present interrupt status

Turning off all interrupts

Checking terminal_quit state

The disabling of interrupts is achieved by an "and with the inverse" operation. This is done for maximum speed.

It should be noted that SW$RAOF directly modifies the ring 0 quit counter, PUDCOM.ROQUIT, and does not call BREAK$. This is done for performance sake.

6 SW$ON - TURNING ON SOFTWARE INTERRUPTS IN RING 3

SW$RON allows code in ring 3 to quickly enable a set of selected software interrupts. It is in general used to end a critical section.

Basically, SW$RON is responsible for one thing; turning on the selected set of software interrupts.

The following algorithm briefly describes the operation of SW$ON:

Turning on the selected interrupts

Checking terminal_quit state

If any interrupts are found pending SW$ABT is called to

handle the interrupt

The enabling of interrupts is achieved by an "or" operation. This is done for maximum speed.

It should be noted that SW$ON directly modifies the ring 0 quit counter, PUDCOM.ROQUIT, and does not call BREAK$. This is done for performance sake.

7 MORE INFORMATION ON THE SOFTWARE INTERRUPT MECHANISM

For information on the details of the software interrupt mechanism consult Software Interrupt Mechanism PE-TI-879.

For information relating to why this new software interrupt control module was built consult Software Interrupt Control Module Proposal PE-TI-1004.

For information relating to the use of the software interrupt control

mechanism consult Software Interrupt Control Module Functional Spec. PE-TI-1005.