DATE: November 14, 1984

TO: List

FROM: Joel H. Ball Jr.

SUBJECT: SLEEP$I an Interruptible SLEEP$

REFERENCE: Subroutines guide, SLEEP$
SLEEP$I Proposal, OSDOC-PROP-120.0

KEYWORDS: SLEEP$, Software Interrupts

ABSTRACT

This document describes the functionality and design of the SLEEP$I routine. For information on why this routine was written, please refer to the SLEEP$I proposal, OSDOC-PROP-120.0. Please return all comments by Friday, 11/30.

1 Introduction

The need for an interruptible SLEEP$ routine is compounded by the advent of more and more software interrupts. This document describes the routine calling sequence and its usage. This document also describes the details of how the routine will be implemented in Primos.

2 Functional Description

The following sections describe the subroutine interface to the new interruptible sleep routine.

2.1 SLEEP$I

Interruptible SLEEP$

SLEEP$I		SLEEP$I

Subroutine NOV. 14, 1984

Name: SLEEP$I (SLEP$I)

Purpose:

This routine allows the caller to be interrupted by a software interrupt, and return from the interrupt without over SLEEP$ing. This is done by the caller providing the timer to use.

Usage:

```
dcl SLEEP$I entry(fixed bin(31));

call SLEEP$I(some_time);
```

some_time - Amount of time (in milliseconds) to sleep.
Note: this must be a variable and may not be a literal value.

3 Design Notes

The following sections describe how the SLEEP$I routine will be implemented, and how the SLEEP$ routine will be modified to make use of it.

3.1 SLEEP$I Description

The current problem with SLEEP$ is that it calls STIMER which makes a local copy of its argument (never mind that SLEEP$ also does the same thing). Therefore the need for a new STIMER that does not make a copy of the "timer", but, instead uses the variable passed to it to while away the time with. SLEEP$I will be a simple routine which calls a new STIMER, STIMER_I, that will modify the caller's argument. Modifying STIMER will not do, as there are too many places in Primos that STIMER is called from with a literal. STIMER_I will be a duplicate of the code for STIMER, except that it will not copy its argument.

The implication here is that the caller's argument to SLEEP$I will be the one passed to STIMER_I and no copy will be made of it. So, writers of programs using writable shared memory, beware! They should continue using the SLEEP$ routine.

3.2 Modifications to SLEEP$

The entry point SLEEP$ (module SVCAL$.PMA) will be deleted, and a new routine will be written that continues to make a copy of its argument, but will no longer call STIMER. Instead it will call the new SLEEP$I routine to do its napping. It will also be moved into the Ring3 load instead of remaining in the Ring0 load.

4 Compatibility Issues

SLEEP$ will be fully compatible with its previous version. SLEEP$I is new and has no compatibility issues. The documented entry point for SLEEP$I will be SLEP$I, not SLEEP$I. This will allow calling from Fortran routines. There are no other known compatibility issues.