Published: 10/28/66

Identification

Pseudo-Drum Module
S. Kidd, G. F. Clancy

Purpose

The drum simulator module is a complete core simulator of the MSU-32 drum written to help check out the drum DIM. It simulates the drum completely, including all timing dependencies and detectable hardware-generated interrupts.

Introduction

The pseudo-drum is a set of routines allowing a 6.36 or 64.5 user to simulate the Firehose Drum (see GE pps M50EB00098 for complete specifications). The caller sets up the hardware queue exactly as he would for the real drum and issues a "connect". As the program continues from the time of the connect, the current status word (CSW) is periodically updated; the commands in the drum queue are simulated; abnormal status words (ASW's) are placed in the status queue when pathological conditions or programmed interrupts are encountered in the command queue; and interrupt-type commands result in simulated traps by the channel. The caller can "disconnect" the channel by the appropriate call.

The pseudo-drum works by reserving a block of core for the "drum". Every "n" micro-seconds (where n is given by the caller), the pseudo-drum takes a clock trap, steals enough CPU time to do its dirty-word and then returns to the interrupted program. At each clock trap (i.e., at the end of drum sector latency) the "channel" simulates a program interrupt if the previous DCW was an interrupt-type command. It then looks at the next DCW, updates the CSW, and executes the command exactly as the channel would, modifying the status queue as necessary.

Usage

There are four calls the user must make to use the pseudo-drum.

1. call setup;

This call does some initialization which must precede any of the other pseudo-drum calls.

2. call defmsu (base, sectors, tracksets, delay);

This call serves to define the MSU being simulated. Assume the following declarations:

```
dcl base fixed bin (18),      /* MSU base address (18 bits)--
                                 normally set manually on the
                                 DSC */,

sectors fixed bin (17),       /* number of sectors/track.
                                 Always 128 or 256 */

tracksets fixed bin (17),/*      number of track sets on MSU */

delay fixed bin (17);         /* sector time in us. of the
                                 simulated drum.  The real drum
                                 takes 135 us./sector at 3600 rpm.
                                 The simulated drum would normally
                                 have a value of at least 600 in
                                 order that the drum simulator
                                 not steal an inordinate percentage
                                 of the machine. */
```

3. call defivt (intvectorg);

Assume the following declaration:

```
dcl intvectorg pointer;
```

It is necessary for the user of the pseudo-drum to create a dummy interrupt vector and fill in its entries (SCU, TRA pairs). This call defines the origin of that interrupt vector for the benefit of the pseudo-drum. When the pseudo drum channel program module recognizes a fault of type i, it executes an XED of the ith pair in the interrupt vector, simulating the trap. Type 1 traps are programmed interrupts; type 2 are data faults like parity errors (never recognized by the drum simulator); and type 3 traps are control faults (the simulator will recognize types c4 and c6. (See page 3.30 of the reference document). If the interrupt vector has not been loaded, the 645 simulator will probably fault on an attempt to execute a 0-opcode if the pseudo-drum tries to simulate an interrupt.

4. call cioc (pcw);

Assume the declaration

```
dcl pcw bit (36) /* PCW is the operand of the CIOC instruction
                    the user would have executed in order to
                    disconnect, increment the service pointer,
                    or start the MSU */
```

The PCW is assumed to have a channel command in bits 18-23 and the DCW relative address in bits 0--17. If the PCW is either a disconnect or increment service pointer, the pseudo-drum will make the appropriate adjustment in the CSW and status queue and will then return. If the command is a start MSU then the channel will execute the first element of the DCW chain and return. The remaining elements will automatically be executed, 1 every "delay" us., until the channel hits a DCW disconnect or the user program gives a CIOC call with a disconnect PCW.

Restrictions:

Since the pseudo-drum depends on the simulated 645 interval timer to get control, the user program must not alter it. In particular, timing-dependent programs which also depend on a timer runout fault will not be able to use the pseudo-drum.