SIMULATION OF THE HMS 5050 COMPUTER SYSTEM

Leonard H. Weiner; Steven L. Huyser;

and Bernhard Weinberg

Michigan State University

Abstract

The HMS 5050 Computer System was designed to allow Computer Science students at Michigan State University access to a computer which has an interrupt system and programmable data channels. The HMS 5050 simulator, which runs on MSU's CDC 6500, enables the students to implement a major component of an operating system, and has been used successfully there since fall term 1972. In this paper we present the design goals of the HMS 5050, discuss its architecture and the structure of its simulator, and summarize classroom experience with the simulator.

Introduction

Undergraduate students in the Department of Computer Science at Michigan State University are required to take a systems programming course during their junior year. As part of a comprehensive study of computer operating systems, each student writes a simple but realistic monitor. In order to do this, the students must have access to a computer which has, among other features, programmable data channels (I/O processors) and an interrupt system. Unfortunately, the academic computers at MSU (a CDC 6500 and a CDC 3600) are operated on an around-the-clock closed shop basis, and the required student access is just not available.

The first attempt to solve this problem, as reported by Forsyth [2], led to the design of the MSU 6507, a modification of the CDC 6500, and the implementation of a simulator (an interpreter) that gave the students the needed machine access. This machine was, essentially, a CDC 6000-series computer with seven additional instructions. These instructions allowed student-written monitor programs to process interrupts, set a time slice, move block storage, and get the simulated time of execution. In giving I/O commands, the monitors communicated with the MSU 6507 in the same manner (through RA+1 requests) as the students would have communicated with SCOPE, the standard CDC 6500 operating system [1]. The similarity between the MSU 6507 and the CDC 6500 caused considerable confusion among the students; they were never quite sure whether their monitor or the SCOPE monitor was in control of the MSU 6507 user's jobs.

The MSU 6507, although a limited machine, did simulate the fundamental characteristics of a programmable interrupt system and demonstrated that the great majority of undergraduate Computer Science students are capable of writing a simple monitor system.

Because of the shortcomings of the MSU 6507, a new and complete computing system, the HMS 5050, was designed. An HMS 5050 simulator was implemented in time for use during Fall term, 1972.[1]

Design Goals of the HMS 5050

The first objective was to make the HMS 5050 simulator appear to the students to be a self-contained computer system; the host computer (CDC 6500) was to be completely transparent. Some other design goals of the HMS 5050 were:

1. The architecture of the main frame and the machine language instruction set were to be similar to those of the CDC 6400. Since MSU CPS students have extensive experience with CDC 6000-series assembly language programming prior to the junior year, it was felt that they should be allowed to use this experience rather than learn a new assembly language for this course.

2. The input-output instructions were to be realistic; this implied that the HMS 5050 would have programmable data channels. Although the CDC 6500 input/output system does not have such channels, the CDC 3600 and many medium- and large-scale computer systems do. Familiarity with, and availability of, the CDC 3600 determined its selection as a model for the HMS 5050 input/output system.

3. The interrupt system was to be representative and have detectable interrupt conditions in the main frame and the data channels. Because the input/output and interrupt systems are intimately related, the model for the HMS 5050 interrupt system was again that of the CDC 3600.

4. The peripheral devices were to be modeled on devices currently in use. As a minimum, the HMS 5050 was to include high speed card readers (1000 cpm), line printers (1000 lpm), and an operator's console; a medium size disk

storage device and tape drives were to be provided when convenient. CDC 3000 peripheral devices were chosen as models for all but the disk.

5. A realistic software support system was to be available to the HMS 5050 systems programmer. This was to include an assembler, various library and editing routines and a bootstrap loader, so that the programmer might create and load an HMS 5050 deadstart program containing his monitor. CDC 6000-series standard software system was chosen as the model.

From this list of design decisions came the name of the HMS 5050: the designating number represents the arithmetic mean of the machine designations 6500 and 3600, while HMS is a somewhat immodest initialism for Huyser's Magnificent Simulator.[2]

Design of the HMS 5050

The HMS 5050, as stated previously, evolved from a marrying of concepts and features of Control Data 6000 and 3000 series computers. Basically, the central processor instruction set and operating registers of the CDC 6400 were integrated with the interrupt and input/output features of the CDC 3600. To facilitate such a scheme, additional operating registers had to be implemented. These allow for interrupt processing and include an interrupt mode register, interrupt (condition) register, main product register (which contains the instantaneous logical product of the contents of the interrupt mode and interrupt registers), a time limit register, and a free running clock register. The complete assemblage of HMS 5050 registers is shown in Table 1.

Instructions not found in the CDC 6400 instruction set were devised to manipulate these added registers, to process interrupt conditions, and to interface with the data channels. The new instruction set includes interrupt-protected "monitor only" instructions to clear channels, connect equipment to data channels, obtain channel status, transmit function codes to channels, initiate read or write activities, and clear interrupt conditions. Non-protected instructions allow register-to-register transfer or swap, sensing of interrupt conditions, copying the CPU register package to and from central memory, jumping to instructions not located in the first instruction parcel of a word (this allows interrupting the central processor at other than word boundries), and direct core-to-core data transfer.

The data channels, being programmable entities, have direct, unrestricted access to central memory, where both their instructions and the data they process are located. The operation of each of the fifteen data channels is asynchronous and independent of the central processor and other channels, although activity on a channel may be initiated only by the CPU.

The interrupt scheme of the HMS 5050 allows interrupts on conditions signalled by either a data channel or the central processor. These include interrupts on memory bounds, time limit, illegal instruction, illegal register reference, and illegal floating point operands. Three interrupt conditions are available for each channel: normal termination, abnormal termination and storage reference fault. When the interrupt system is activated, trapping to location 000001 occurs automatically; i.e., the CPU retains the position of the last instruction executed by

TABLE 1
Hardware Registers of the HMS 5050

Register Name	Mnemonic	Length (bits)
Index	B0 to B7	18
Address	A0 to A7	18
Operand	X0 to X7	60
Reference Address*	RA	18
Lower (Memory) Bounds*	LB	18
Upper (Memory) Bounds*	UB	18
Mode Selection*	MS	18
Time Limit*	TL	18
Lower Read-only Bounds*	LR	18
Upper Read-only Bounds*	UR	18
Interrupt Mask*	IM	60
Interrupt (Condition)*†	IR	60
Main Product*†	MP	60
Program Address*†	P	18
Instruction Position*†	IP	18
Clock*†	CK	60
Last Jump Address*†	LJ	18

*Protected registers. When not in monitor mode, any attempt to alter their contents will cause an illegal register interrupt condition.
†Read-only registers.

planting a (return) jump instruction in location 000000, and takes its next instruction from location 000001. After the interrupt condition has been processed, exit is made by jumping to location 000000; execution of the program that was in effect when the interrupt occurred is resumed.

HMS 5050 peripheral equipment may be connected to any of the fifteen programmable data channels through use of an equipment/unit designation unique to that channel. Up to eight equipments per channel and thirty-two units per equipment are permitted. An "equipment" is usually a peripheral controller; the "units" are devices connected to the controller. (The configuration of the HMS 5050 model currently in use includes three card readers, four line printers, one magnetic tape drive, and one disk system.)

Simulating the HMS 5050

The simulator for the HMS 5050 is a modular, table-driven processor written for execution on a Control Data 6500 under the standard SCOPE 3.2 operating system. The simulator, coded in COMPASS (the CDC 6000 assembly language), has three primary sections: the initialization or "deadstart" procedure, the central processor simulator and the input/output equipment simulator (including simulators for each type of I/O equipment).

The initialization procedure causes the absolute binary file of the deadstart program to be loaded into the HMS 5050 central memory, beginning at location 000000. All registers are cleared and all channels are deactivated. This section of code also produces a termination dump of the register package and the current channel status words when the HMS 5050 is halted (see Appendix).

The central processor simulator is composed of three units. The first unit forms the logical product of the contents of the interrupt mode and interrupt registers and puts it into the main product register. If the result is non-zero, an interrupt condition is present and the interrupt system is activated as described above. The code then checks for other possible interrupt conditions, such as the instruction address being out of bounds, and when one is found, sets the appropriate bit in the interrupt register. Otherwise an attempt is made to fetch the next instruction to be simulated.

The second unit parses the instruction. The code used for the parsing is selected according to an entry in a jump table. The actual simulation code for the instruction is then executed. Again, this code is selected by means of a jump table. The individual pieces of code used to simulate instructions are, for the most part, composed of macro calls. The use of macros here allows flexibility and modularity in adding or modifying instruction simulation. The instruction simulation code contains these basic elements:

1. incrementing the free-running simulated clock by the number of machine cycles required to execute the simulated instruction,
2. checking for interrupt conditions in the instruction operands and, if indicated, setting the appropriate interrupt condition bit,
3. simulating the instruction,
4. checking for interrupt conditions caused by the execution of the instruction and setting the appropriate interrupt condition bit.

Once the above steps have been completed, the CPU simulator loops back for the next instruction.

The third unit is the I/O simulator, of which the section that simulates data channel activity is the most complex. The main components are: device drivers, channel tables, equipment tables, unit tables and the action list. Macro constructed tables configure the system at assembly time. There is a device driver for each type of device available (e.g., card reader, printer, tape drive).

The configuration and current activity of all equipment is maintained in a tree whose depth is three levels: The root of the tree represents the HMS 5050 CPU; the nodes of the first level are the data channel tables, which branch to the equipment tables (the nodes of the second level). These, in turn, lead to the unit tables, the leaves. The availability of a certain device or channel is determined by the information found on the tree. See Illustration 1 for a typically structured tree.

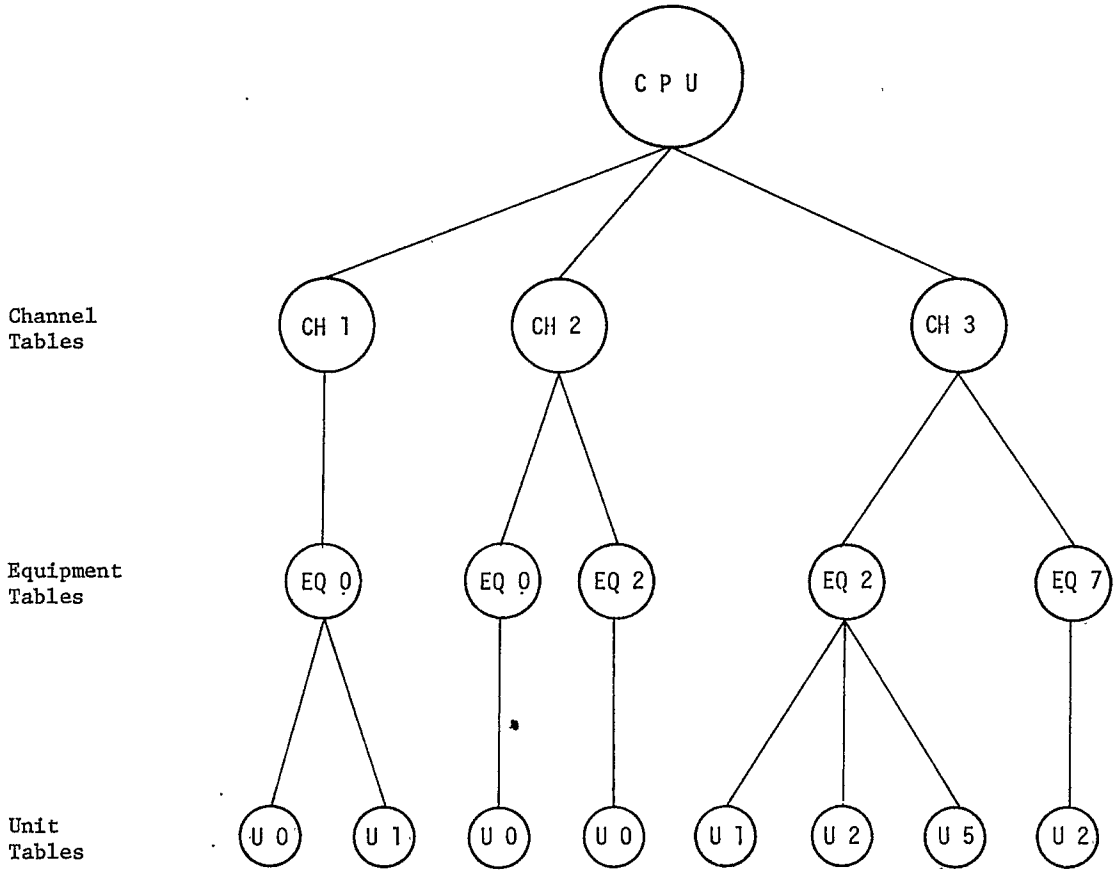
A two-way linked-list (the action list) is used to simulate asynchronous I/O activity. This list contains the simulated time at which a specified device, assigned to a specific channel/equipment/unit, is to continue its activity. For example, assume that a read is in progress on a card reader. Since each card takes a certain amount of time to move through the card reader, the simulated time (the time at which the next card image is to be transferred from the card reader buffer to central memory) is placed on the action list, along with information linking that action list entry with the specific unit. The entry is linked in time-order so that the first entry on the action list contains the time the next I/O activity is to take place. At the end of simulation of each CPU instruction, the current simulated time is checked against the top entry in the action list. When it is greater than or equal to the time of the next I/O activity, the required action, associated with the device driver on the channel/equipment/unit specified in the list, is simulated and control is returned to the CPU simulator.

Each device driver is composed of seven sections:

1. Function list and associated simulation code,
2. Read simulation code,
3. Write simulation code,
4. Clear channel simulation code,
5. Code to return the simulated channel/equipment/unit status,
6. Connect equipment simulation code,
7. Disconnect equipment simulation code.

ILLUSTRATION 1

Hypothetical Configuration of Channels, Equipments and Units for the HMS 5050.



The function list and associated simulation code is used to identify and simulate functions transmitted to the device by the CPU. A line printer, for example, might have a function code to cause the paper to eject to the top of the next page.

The read and write simulation codes initialize information in the unit tables, such as reading up the next channel instruction, calculating the next action time, and creating a linked entry in the action list. This simulation might consist, in the case of a printer, of accumulating a buffer of data and then transmitting it to the SCOPE operating system to be written on a file for later print disposition. A printer, of course, would not have any actual read simulation code; the code provided for reading would simply simulate a channel reject.

As their functions are more immediate, the other four sections of simulation code in each device driver do not utilize the action table. They clear a connected channel/equipment/unit, transmit the current status back to the CPU for any unit, connect an equipment/unit to a channel, and disconnect an equipment/unit from a channel, respectively.

In order to present a clearer picture of how these sections work together, we will follow an example through each of its I/O simulation steps. Assume that we are reading a set of cards from the card reader connected to channel 2 as equipment 3, unit 0. (See Table 2.)

TABLE 2

Card Reading

User Steps	Simulation Steps
1. Clear channel 2 of any connected units and interrupts.	a) All branches of the channel 2 equipment/unit tree are scanned for any connected equipment. b) If any is found, it is disconnected and the associated link in the action table is removed. c) Interrupt bits for channel 2 are cleared.
2. Connect equipment 3, unit 0 on channel 2.	a) All branches of the channel 2 equipment/unit tree are scanned for any connected equipment. b) If any connected unit is busy, the connection is rejected. c) Connect equipment 3, unit 0 by making appropriate table entries.
3. Request the status of that unit.	a) The current status word of the unit is updated and returned.
4. Initiate a read action on channel 2.	a) Check to see if unit is connected; reject if not. b) Get a copy of the data channel instruction (channel control word) from central memory (CM). c) Enter the control word in the unit table. d) Calculate time when card image is to be transmitted to CM. e) Form action list entry. f) I/O activity on that unit subsides until the CPU simulator returns control, the time of the requested action having come to pass. g) Characters are read from a file simulating a card reader and transferred to a CM word specified by the control word copy; the copy is updated. h) Loop back to d) until the activity requested by the current control word has been processed. i) Read new control word; if end, select normal termination interrupt and end processing, else loop back to d).

Simulator Size and Operating Cost

The assembly language source deck for the HMS 5050 simulator contains 4000 cards which, after macro expansion, represent about 12,000 CDC 6500 machine language instructions. The simulator uses about 12,800 memory locations of which 8,192 are for the HMS 5050 simulated memory.

Because the simulator checks for a number of possible interrupt and error conditions before it interpretively executes each HMS 5050 instruction, it executes an average of thirty CDC 6500 CPU instructions for each simulated HMS 5050 CPU instruction. This 30:1 performance degradation, however, does not show up in the job cost. Because the HMS 5050 assembler is a stripped down version of the CDC 6500 assembler, HMS 5050 source programs require less assembly time than do equivalent CDC 6500 programs; for short running programs, assembly and I/O costs far outweigh execution costs.

Test programs, which do not execute input/output instructions or do interrupt processing, have been run on both the HMS 5050 simulator and the CDC 6500. Job costs for these HMS 5050 programs average only about 20% more than for the CDC 6500 programs.

It is difficult to estimate the increased cost of running student monitors under the simulator, but regular CDC 6500 programs of similar size and complexity incur costs of no less than half those incurred by the monitors.

Classroom Use of the Simulator

The students are introduced to the HMS 5050 simulator by means of two documents:

1. The HMS 5050 Computer System Reference Manual describes the machine architecture which includes operating and special purpose registers, the interrupt system and conditions, HMS 5050 machine instructions not included in the CDC 6500 instruction set, and

channel control word instructions for input/output processing, along with other HMS 5050 features.

2. The HMS 5050 User's Manual describes the interface between the host computer and the HMS 5050 simulator, the special features of the HMS 5050 assembler, the deadstart procedure for initiating execution of the HMS 5050, interrupt processing, and input/output processing (data channel programming).

Early in the term, the students are given a preliminary HMS 5050 programming assignment designed to familiarize them with the machine. This is a simple program to read and echo-print, alternately, a deck of data cards using the simulated HMS 5050 card reader and line printer on channel 1. (An HMS 5050 program that does this is shown in the Appendix.)

The monitor assignment, the largest programming assignment the students have encountered thus far in the computer science curriculum, is given to them in two parts at the end of the second week of the term.

1. Monitor 1.0 is a simple batch monitor to load and execute user jobs from a single job queue, one job at a time. Although the jobs prepared for Monitor 1.0 do not perform input or output, they cause a variety of interrupt conditions that the monitor must recognize and process. Depending on the nature of the interrupt, it should either return control to the user job or terminate it, and then load the next.

2. Monitor 2.0, an extension of the 1.0, is supposed to manage three simulated job queues, each from a different data channel, in a multiprogramming environment. This means that Monitor 2.0 must maintain three separate user jobs in CM simultaneously, assigning the CPU to each in turn whenever an interrupt occurs. These user jobs, require the monitor to limit their total processing time to a specified number of simulated machine cycles, and vigorously exercise the interrupt system.

Those students who complete Monitor 2.0 early enough are encouraged to extend their monitor to perform any or all of the following functions for a varying amount of extra credit:

1. multiprogramming with memory compaction,
2. time slicing of the CPU among user jobs,
3. servicing dynamic user requests for a change in CM,
4. rollout, to disk, user jobs waiting for more CM,
5. aging of user job priority and managing a priority scheme for CPU assignment,
6. providing individual job logs appended to each user job's output,
7. job swapping to disk,
8. maintaining I/O queues on disk.

A brief discussion of student performance over this assignment set is given in the next section.

Summary

Systems programming students at MSU have found the HMS 5050 simulator to be a realistic computer system with programmable interrupts and data channels. In the three terms it has been used, approximately 90% of the students have been able to complete Monitor 1.0 and 75% have completed Monitor 2.0. About a third of the latter group attempt some of the extra credit extensions to their monitor, with a wide range of success; one or two students in each class of thirty-five ordinarily complete a full-blown monitor which contains most of the features listed in the previous section.

These percentages are about the same as those achieved by earlier classes who used the MSU 6507. But recent students, by virtue of their HMS 5050 experience, have learned considerably more about interrupt and input/output processing than did their predecessors.

References

1. Control Data 6400/6500/6600 Computer Systems SCOPE Reference Manual, Control Data Corporation Publ. No. 60189400, Rev. L, 1971.

2. Forsyth, John, "The MSU 6507 Interpreter for the CDC 6500," Abstracts of the Computer Science Conference, page 80, 1973.

Notes

[1]S. Huyser, sponsored by L. Weiner, did the original design of the HMS 5050 and also implemented the simulator.

[2]Because Allan Moluf, a student at MSU, later developed the HMS 5050 I/O simulator, HMS now stands for the Huyser-Moluf Simulator.

```
                                        IDENT   EXAMPLE
                                        HMS                 MACRO TO,SELECT ABSOLUTE ASSEMBLY AND LIST OPTIONS.
                                        LIST    -8,-R       HMSTEXT OF  07/06/73  AT  .02.50.35.

 0  43103                               MX1     3           CREATE MASK IN X1 TO BE USED TO DETECT
        20122                            LX1     18          INTERRUPT CONDITIONS ON CHANNEL ONE
              0160002137                 RT      X1,IM       PLACE THE MASK INTO THE INTERRUPT MASK

 1  00101                    LOOP        CC      1           CLEAR CHANNEL 1
 2  00200000000400000026                CE      1,0,0,REJ   CONNECT CARD READER TO CHANNEL 1
 3  00310000000400000000                CS      X1,1        STATUS CHANNEL 1
 4  20164                               LX1     59-7        AND TEST EOF BIT TO SEE IF ALL CARDS READ
        0331000021                       NG      X1,FINISHED
 5  00500000270400000026                BR      CCW,1,REJ   ISSUE READ ON CHANNEL 1
 6  0100000014                          RJ      WAIT        WAIT FOR READ TO FINISH
 7  00101                               CC      1           CLEAR CHANNEL 1
10  00200000000440000026                CE      1,1,0,REJ   CONNECT LINE PRINTER TO CHANNEL 1
11  00600000270400000026                BW      CCW,1,REJ   ISSUE WRITE ON CHANNEL 1
12  0100000014                          RJ      WAIT        WAIT FOR WRITE TO FINISH
13  0200000001                          JP      LOOP        TRY TO READ ANOTHER CARD

14                           WAIT       BSSZ    1           ENTRY POINT OF SUBROUTINE
15  00400000000476000026                FT      1,768,REJ   FUNCTION CHANNEL 1 TO INTERRUPT
                                 *                               ON ALL CONDITIONS
16  00700                               WT                  IDLE CPU UNTIL OPERATION IS FINISHED
                                 *                                (BIT IS SET IN INTERRUPT REGISTER.
17  00400000000477000026                FT      1,778,REJ   FUNCTION CHANNEL TO CLEAR INTERRUPT
20  0200000014                          JP      WAIT        RETURN TO CALLING ROUTINE

21  00101                    FINISHED   CC      1           CLEAR CHANNEL 1
22  00200000000440000026                CE      1,1,0,REJ   CONNECT LINE PRINTER TO CHANNEL 1
23  00600000300400000026                BW      EOFCW,1,REJ ISSUE WRITE ON CHANNEL 1
24  0100000014                          RJ      WAIT        WAIT FOR WRITE TO FINISH
25  0000000000                          PS                  HALT THE CPU

26  0000000000               REJ        PS                  HALT THE CPU HERE ON AN ERROR

27  60000000010000000031     CCW        IOTR    1,BUFFER    ECHO PRINT CONTROL WORD
30  60000000010000000051     EOFCW      IOTR    1,EOFMSG    END-OF-FILE MESSAGE CONTROL WORD

31                           BUFFER     BSSZ    16          ECHO PRINT BUFFER
51  47051604461706460611     EOFMSG     DIS     12,*END-OF-FILE ENCOUNTERED

65                                      END
```

APPENDIX

HMS 5050 Program to Read and Echo-Print Cards

```
        ** HMS 5050 TERMINATION DUMP **

 P  000025    IP 000036    A0 000000    X0 00000000000000000000    IM 00000000000000700000
 RA 000000    B1 000000    A1 000000    X1 40000000000000000002    IR 00000000000000010020
 LB 000000    B2 000000    A2 000000    X2 00000000000000000000    MP 00000000000000000000
 UB 020000    B3 000000    A3 000000    X3 00000000000000000000    CK 00000000000014321446
 MS 400000    B4 000000    A4 000000    X4 00000000000000000000    CI 00000000006100046000
 TL 000000    B5 000000    A5 000000    X5 00000000000000000000    LJ 000014
 LR 000000    B6 000000    A6 000000    X6 00000000000000000000
 UR 000000    B7 000000    A7 000000    X7 00000000000000000000
-CH-    CCWADR    STATUS OP  PR-CNT    PR-ADR
 01 00000000300000000001 60000000000000000065
 02 00000000000000100000 00000000000000000000
 03 00000000000000100000 00000000000000000000
 04 00000000000000100000 00000000000000000000
 05 00000000000000100000 00000000000000000000
```

Channel 1 interrupts selected.

Clock register - number of cycles (octal) since deadstart.

Indicates last jump instruction.

Control word at end of operation on channel 1.

Channel 1 is connected and ready, the last control word was read from address 30. Channels 2-5 are shown not available or not connected.

Octal dump of registers and channel status produced by the HMS 5050 at termination.

```
***   PRINTER 110

      THIS IS THE FIRST CARD IMAGE FOUND ON THE INPUT DEVICE


[     THIS IS THE FORM OF THE FILE CARD FOR THE HMS5050 (7-8 PUNCH IN COLUMN ONE)
            A 7-8 PUNCH IN COLUMN ONE APPEARS ON OUTPUT AS THE GRAPHIC, [

MNEMONIC      INSTRUCTION

CC            (CLEAR CHANNEL)      HALT ANY I/O IN PROGRESS ON THE CHANNEL,
                                   CLEAR ANY INTERRUPTS ASSOCIATED WITH THAT CHANNEL,
                                   AND DISCONNECT ANY CONNECTED EQUIPMENT.

CE            (CONNECT EQUIPMENT) CONNECT THE SPECIFIED UNIT AND EQUIPMENT TO
                                   THE CHANNEL.  NOTE: NOT MORE THAN ONE EQUIPMENT AND
                                   UNIT MAY BE CONNECTED TO A CHANNEL AT ONE TIME.

CS            (COPY STATUS)        REQUEST COPY OF CURRENT STATUS OF CHANNEL.

FT            (FUNCTION TRANSMIT)  SEND FUNCTION CODE TO EQUIPMENT ON CHANNEL.

BR            (BEGIN READ)         INITIATES A READ ON THE CHANNEL.

BW            (BEGIN WRITE)        INITIATES A WRITE ON THE CHANNEL.

WT            (WAIT)               IDLES THE CPU UNTIL AN INTERRUPT OCCURS

    =IOTR= IS A CHANNEL CONTROL (COMMAND) WORD. THE FIRST PARAMETER SPECIFIES
THE NUMBER OF RECORDS TO BE TRANSFERRED, AND THE SECOND PARAMETER SPECIFIES
WHERE THE RECORDS ARE READ FROM OR WRITTEN TO.


         THIS IS THE LAST CARD TO BE ECHO PRINTED
*END-OF-FILE ENCOUNTERED
```

User Program Output

This example was programmed by Thomas P. Carr, the student who implemented the HMS 5050 disk simulator.