NICOBUG

NICOLET INSTRUMENT CORPORATION OCTAL DEBUGGING ROUTINE

INTRODUCTION

Nicobug is an octal debugging program specifically designed for Nicolet 1080 series computers. It can be used to examine and change memory locations and as a simple method of stepping through a program a few instructions at a time.

STORAGE

At present, Nicobug occupies locations 6000 - 6413, but can be reassembled to occupy any desired locations.

EXAMINATION AND CHANGING OF LOCATIONS

Nicobug accepts only octal numbers as arguments and single characters as commands.

nnnnnn/	causes the contents of location nnnnnn to be typed out following the slash. Until either Return or Line Feed is typed, this location is open for modification. A new octal number can be entered while the location is open, causing this number to be deposited in memory when Return or Line Feed is typed.
Line Feed	If a location is already open, it is closed and the next sequential location and its contents are typed. If no location is open, the next sequential location is opened for examination and modification.

For example, in the following, the underlined numbers are typed by Nicobug:

123/<u>000506</u> 100506 (Return)	Location 123 is opened and changed from 506 to 100506.
123/<u>100506</u> (Line Feed) <u>000124/3240120</u> (Line Feed) <u>000125/0001307</u> (Return)	Locations 123 - 125 are examined sequentially

BREAKPOINTS

Ideally, when one is debugging a complex program, it is desirable to be able to step through the code a few instructions at a time, and examine the various registers at each step. This could be accomplished by pushing the Single Step button and Executing each instruction by thumb, but this can prove extremely tiresome if a large amount of code is involved.

Alternatively, one could insert STOP commands in various locations and allow the computer to execute instructions until it encountered a STOP. Then one would have to re-enter the instruction replaced by the STOP, toggle in a new STOP command, and then proceed from the original stop. This unduly tedious procedure can be simulated more conveniently by using Nicobug.

At the location where one wishes to examine the state of various registers, one need only insert a breakpoint. Then the computer is started, executing instructions. When it encounters the breakpoint, it jumps to Nicobug, and types out the contents of the Link and AC. It is then possible to examine and change desired locations and continue from the breakpoint.

> NOTE: In order to make use of this breakpoint feature, it is necessary that one location on each memory page be reserved as a pointer. The location chosen for use in Nicobug, is the LAST location on the page, location 1777. Thus, in the lowest 4K, locations 1777, 3777, 5777 and 7777 are reserved for breakpoint pointers. This location is replaced with a pointer to Nicobug when a breakpoint is inserted. When the breakpoint is removed, the break location is restored, but location 1777 is not. Therefore, no program information can be contained in the last location of any page where a Nicobug breakpoint is to be inserted.

The breakpoint instructions are as follows:

B	Remove the breakpoint, if any. (There can be no breakpoint at location 0.)
nnnnnnnB	Insert a breakpoint at location nnnnnnn.
C	Continue from the last breakpoint. If, while in Nicobug, one changes the breakpoint, continuation is still from the old breakpoint. The saved AC and Link are restored before continuing.
nnnnnnnG	Begin executing instructions at location nnnnnnn. The saved AC and Link are not used. Note that typing G by itself starts the program at location 0.
A	Open the contents of the saved AC for modification

BREAKPOINT USE EXAMPLES

4106B	A breakpoint is placed at location 4106.
4000G	The program is started at location 4000
0004106 1;0000240	When the program reaches location 4106 it jumps to Nicobug and prints out the location of the break and the contents of the Link and the AC.
4076/0005606	Location 4076 is examined
5173B	A breakpoint is inserted at 5173
C	The program is restarted at 4106
0005173 0;0000000	The break at 5173 occurs
B	The breakpoint is removed
C	Program execution continues