Vintage Computing

NCR390 Simulator and Compiler

Simulator and Compiler Programs for the NCR390 Computer from the 1960s

Ad for an NCR390 from the 1960s

DESCRIPTION

The NCR390(R) computer was designed and built in 1960, and was in worldwide use in banks and accounting departments during the 1960s and early 1970s. It was the world's first, low cost, mass marketed computer.

This freeware package enables you to run NCR390 programs on a computer simulator (SIM390), using original machine language.

A compiler is also included which you can use to compile programs written in the PL390 language and run them on the simulator, instead of coding them in machine language, if you prefer. The compiler was not available when the NCR390 was in use, but was written years later to make it easier to write programs for the simulator.

The simulator and compiler run on a PC under MS-DOS.
Native DOS can be used, or Microsoft Windows can be used by using a DOS prompt (a command prompt).

System Requirements:

- **486 PC or higher.**
- **MS-DOS or any Windows system with a DOS prompt (command prompt).**
- **15 megabytes of disk space (mostly for the optional pictures).**

The package includes the following items:

1. SIM390 - The NCR390 simulator program.
2. PL390C - The PL390 language compiler.
3. Documentation.
4. Sample programs (HELLO WORLD, TIC-TAC-TOE, etc).
5. NCR390 photos and ads from the 1960s.

Topics in this document are color coded:

- **General information**
- **Installation, machine specs, acknowledgements**
- **HELP informations (commands, etc)**
- **Getting started**
- **The PL390 language and compiler**
- **PL390 quick reference and Op codes cross reference**
- **Historical notes**

WHAT IT LOOKS LIKE...

A short program is entered, using machine language, and then run.

```
-----------------------------------------------
         --- SYSLOG ---
 SIM390 STARTED AT  07-31-2004  19:23:42
 SIM390 VERSION 1.3S
 -----------------------------------------------
 19:23:42
 19:23:42  SIM390 TO INTERRUPT 390 PROG, PRESS SHIFT, CTRL, OR ALT.
 19:23:42  SIM390 FOR HELP, TYPE 'HELP' OR 'H'.
 19:23:42  SIM390 READY.
 19:23:42  SIM390 IDLE.
 19:23:44  k     <=== Simulate the KIE key (K or KIE) to enter data using the keyboard
 19:23:44  SIM390 (000):   KIE - CELL 000
 19:23:53  000088890001    <=== Enter data into the 390's memory (cell 000)
 19:23:53  SIM390           (000)       888,900.01    <=== Echo of data entered for cell 000
 19:23:53  SIM390 (001):   KIE - CELL 001    <=== KIE advances to next cell - cell 001
 19:24:04  170088899902
 19:24:04  SIM390           (001) 1,700,888,999.02
 19:24:04  SIM390 (002):   KIE - CELL 002
 19:24:11  010099991403
 19:24:11  SIM390           (002)   100,999,914.03
 19:24:11  SIM390 (003):   KIE - CELL 003
 19:24:19  030000000000
 19:24:19  SIM390           (003)   300,000,000.00
 19:24:19  SIM390 (004):   KIE - CELL 004
 19:24:21  r     <=== Simulate the RESET key (R or RESET) - go into IDLE mode
 19:24:21  SIM390 RESET COMMAND ACCEPTED
 19:24:21  SIM390 IDLE.
 19:24:26  d 000-004    <=== Display the contents of cells 000 thru 004 (a short program)
 19:24:26            (000)  00 00 88 89 00 01   <=== Prog: Read console, cells 88-89
 19:24:26            (001)  17 00 88 89 99 02   <=== Prog: Add 88 and 89, answer in 99
 19:24:26            (002)  01 00 99 99 14 03   <=== Prog: Print cells 99 thru 99
 19:24:26            (003)  03 00 00 00 00 00   <=== Prog: Halt, go to cell 00
 19:24:26            (004)  00 00 00 00 00 00
 19:24:26  SIM390 IDLE.
 19:24:35  *00    <=== Simulate the 'START 00' key (*00) - start the program at cell 00
 19:24:35  SIM390 START 00
 19:24:35           (000): RDC INTO CELL 088.  AWAITING REPLY...
 19:24:39  123    <=== Enter the first number to be added
 19:24:39                   (088)             1.23   <=== Echo of reply (123) with cell number
 19:24:39           (000): RDC INTO CELL 089.  AWAITING REPLY...
 19:24:43  456    <=== Enter the 2nd number to be added
 19:24:43                   (089)             4.56   <=== Echo of reply (456) with cell number
 19:24:43                   (099)             5.79   <=== Total of 123 + 456 in cell 99
 19:24:43           HALT IN CELL 003. PRESS ENTER TO CONTINUE...  <=== HALT encountered at cell 003
 19:24:46           (000): RDC INTO CELL 088.  AWAITING REPLY...  <=== Program continues after ENTER
 19:24:49  eoj    <=== End-Of-Job command (EOJ) - End the 390 program gracefully
 19:24:49  SIM390 JOB          ENDED VIA CONSOLE COMMAND AT 07-31-2004  19:24:49.
 19:24:49  SIM390 IDLE.
 19:24:52  exit    <=== Exit/end the simulator
 19:24:52  SIM390 ENDSIM COMMAND ACCEPTED
 19:24:52  SIM390 REEL-READER RECORDS  (RR) =         0
 19:24:52  SIM390 STRIP-READER RECORDS (ST) =         0
 19:24:52  SIM390 CARD-READER RECORDS  (CR) =         0
 19:24:52  SIM390 INPUT RECORDS =         0
 19:24:52  SIM390 TAPE-PUNCH RECORDS   (TP) =         0
 19:24:52  SIM390 CARD-PUNCH RECORDS   (CP) =         0
 19:24:52  SIM390 ERRORS =         0
 19:24:52  SIM390 INSTRUCTION COUNT =         5
 19:24:52  SIM390 ELAPSED TIME =     70.6300 SECONDS =    1.17 MINUTES
 19:24:52  SIM390 CPU TIME     =      0.0006 SECONDS =    0.00 MINUTES
 19:24:52  SIM390 CPU BUSY     =     0.00%
 19:24:52  SIM390 STARTED AT 07-31-2004  19:23:42
 19:24:52  SIM390   ENDED AT 07-31-2004  19:24:52.
```

SIM390 System Log (console activity logged by the simulator)

```
 ------------------------------------------------
          --- SYSLOG ---
 SIM390 STARTED AT  02-23-2004  00:01:42
 SIM390 VERSION 1.3R
 ------------------------------------------------
 00:01:42
 00:01:42  SIM390 TO INTERRUPT 390 PROG, PRESS SHIFT, CTRL, OR ALT.
 00:01:42  SIM390 FOR HELP, TYPE HELP OR 'H'.
 00:01:42  SIM390 READY.
 00:01:42  SIM390 IDLE.
 00:01:46  D T      <=== Display the date and time (D T)
 00:01:46  DATE=02-23-2004  TIME=00:01:46
 00:01:46  SIM390 IDLE.
 00:01:49  VER      <=== Display the SIM390 program version (VER)
 00:01:49  SIM390 VERSION = 1.3R  COMPILED ON FEB 23, 2004, 00.01.15
 00:01:49  SIM390 IDLE.
 00:02:00  D A      <=== Display active jobs (D A)
 00:02:00  NO ACTIVE JOBS
 00:02:00  SIM390 IDLE.
 00:02:07  M RR,TTT.OBJ    <=== MOUNT the tape with the TICTACTOE program
 00:02:07  SIM390 MOUNTING TAPE TTT.OBJ
 00:02:07  SIM390 TAPE TTT.OBJ  NOW MOUNTED ON RR. RECORDS=0001145.
 00:02:08  SIM390 IDLE.
 00:02:15  S RDR1      <=== Start Reader1 - the system reader for the Reel Reader tape drive
 00:02:15  SIM390 READER STARTED ON TAPE DRIVE RR.
 00:02:15  SIM390 RR: JOB TICTACA    TICTACA IS TIC-TAC-TOE WITH AUTOLOAD
 00:02:15  SIM390 JOB TICTACA  STARTED.
 00:02:15  SIM390 RR: PROGFMT        PROGRAM IS IN "PROGRAM" FORMAT
 00:02:15  SIM390 PROGRAM INPUT FORMAT SELECTION ACCEPTED.
 00:02:15  SIM390 RR: PRTA           PRINT IN "AUTHENTIC FORMAT" MODE
 00:02:15  SIM390 PRINT-AUTHENTIC MODE ACCEPTED.
 00:02:15  SIM390 RR: AUTOLOAD 05 10 01 99 00 02   <=== An AUTOLOAD command - first instruction
 00:02:15  SIM390 EXECUTING PROGRAM VIA AUTOLOAD...
 00:02:15
 00:02:15  SIM390 READING TAPE ON REEL READER.
 00:02:16  SIM390 READING TAPE ON REEL READER.
 00:02:16  SIM390 REWINDING TAPE ON REEL READER.
 00:02:16  TICTACA           (000)
 00:02:16  TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...  <=== Message from job TICTACA
```

PL390 Program Compile under DOS

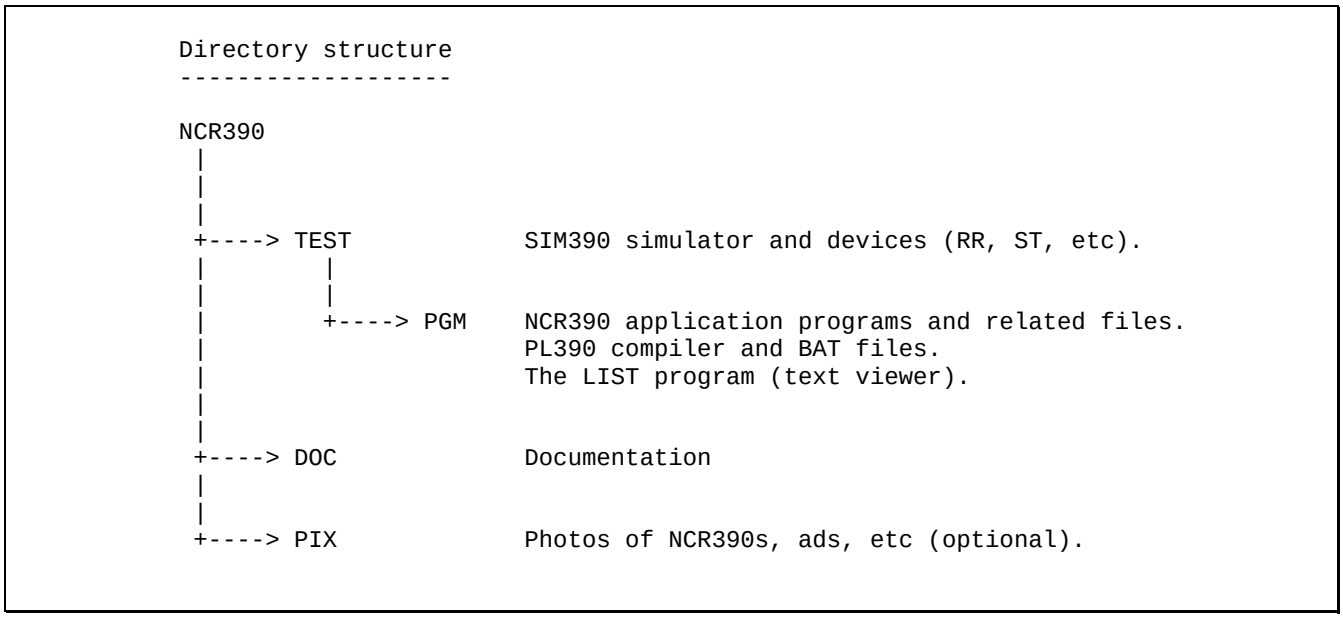
```
F:\NCR390\TEST>COMP TTT    <=== Compile the program TTT.TXT using COMP.BAT
 COMP.BAT   TTT
 Compile and Link for PL390/SIM390.
        1 file(s) copied.

 COMPILING PROGRAM IN MEMBER TTT.TXT

 PL390 COMPILER STARTED. VER=1.4Z   DATE=2004-02-23  TIME=21:05:05.65
 COMPILING PROGRAM TICTACTO    <=== TICTACTO is the program name, specified in the source
 SOURCE RECORDS READ =  1,135
 PASS1 ERRORS   =   0
 PASS2 WARNINGS =   1
 PASS2 ERRORS   =   0
 PL390 COMPILER ENDED.

        1 file(s) copied.
 Listing copied to  TTT.PRT   <=== The name of the listing file for the job
        1 file(s) copied.
 Object deck copied to  TTT.OBJ   <=== The name of the object deck for SIM390 (machine code)

Press any key to continue . . .

 Use the "W" key to toggle the display width of the listing.
 To exit the listing display, hit ESC.

Press any key to continue . . .
```

PL390 Program Listing after Compile (TTT.PRT)

```
0001  09 00 02 03 00 35   342 BEGIN  CLR BOARD THRU WORK3 GOTO CPYCOD    /*CLEAR BOARD, WORK3
                          343
                          344 *          002
                          345 *          CELL 002 = PROGRAM NI AFTER CELL 0 AUTOLOAD
0002  05 14 00 99 03 03   346 LOADUM RPT INTO 100 THRU 199 AT-END REWIND   /*LOAD UPPER MEMORY
0002                      347 BOARD  --- RENAMES LOADUM     <=== THE TTT BOARD AFTER LOAD DONE
                          348
                          349 *          003
0003  05 40 00 00 00 01   350 REWIND RWD GOTO BEGIN        /*REWIND TAPE ON REEL READER
0003                      351 WORK3  --- RENAMES REWIND    /*RE-USE CELL FOR WORK
                          352
                          353 *          004
0004  00 00 99 99 00 05   354 ENTRY  RDC REPLY            /*GET HUMAN'S MOVE
                          355....
```

CONTENTS OF THE PACKAGE

The Zipped files expand to files in several directories.



See the $README.TXT file for more details.

INSTALLATION PROCEDURE

1. Create a new folder (directory) for the package.
   Recommended name: NCR390.

2. Move the zipped package to that directory.

3. Unzip (decompress) the package inplace using any unzipper.
   WinZip, PKUNZIP, or other utilities can be used for this.
   WinZip is available for evaluation free of charge at:
      www.winzip.com

   Specify "include/expand subdirectories" when unzipping.
   The subdirectories should be automatically created.

   The amount of disk space used by the package after installation
   should be about 13-15 megabytes.

   Nothing will be added to the Windows Registry or copied into
   any Windows directories.
   No special DLLs are required or included.

4. The LIST program (text viewer) should be copied to a
   location in your DOS/Windows PATH, such as "C:\".
   It is used for displaying the compiled program listing
   with PL390C.
   Example: XCOPY LIST.EXE   C:\
   This will copy LIST.EXE to the C:\ directory.

UNINSTALL PROCEDURE

Delete the folder where you placed the programs and documentation.

We do not recommend ever deleting this package, since it represents the function of an antique computer, and might be worth something someday...

COMMENTARY

Playing with these programs is not a trivial excercise. Let's face it: Writing programs in machine language or a symbolic assembler type of language is not easy. Nor is learning a new computer language. With this package, you can do either or both - code in machine language or in a symbolic language (PL390).
Prior exposure to the NCR390 would be helpful.

I've tried to make the operation of SIM390 and PL390C as easy as possible, and tried to make the documentation as clear as possible. However, when all is said and done, it's complicated. But that's the nature of this computer, and programming at the "bare metal" level.

For those with NCR390 programming experience, this software will probably be nostalgic, new, and somewhat difficult.

For those with no NCR390 programming experience, this software will probably be new and difficult.

But the point of all this is to revive an obsolete computer via simulation, and to have fun. Writing machine code or symbolic code and running your programs can be a lot of fun. That's mainly why this software was written: To have fun.

OPERATIONAL INFORMATION

General:

"SIM390" is a DOS-based simulator program which runs on a PC and
simulates an NCR390(R) computer which was in use during the
1960s.

"PL390C" is a DOS-based compiler program which runs on a PC and
compiles PL390 source programs. The compiled output can be run
on the SIM390 simulator.

The PL390 language is a post-1960s creation, and was **not**
used on the NCR390 computer.
Its use is optional and is not required for running the simulator.

Both programs can run from DOS prompts under Windows, or
in native DOS.

COMPATABILITY:

SIM390 and PL390C have been successfully tested with:

Windows XP/Pro	(DOS prompt)
Windows NT	(DOS prompt)
Windows ME	(DOS prompt)
Windows 98SE	(DOS prompt)
Windows 98	(DOS prompt)
Windows 95	(DOS prompt)
PC/DOS 7.X	(Last PC/DOS release from IBM - 1994)
MS/DOS 7.X	(From Microsoft Windows 9x)

SIM390 and PL390C have *NOT* been tested with:

Windows 3.X	Windows 2000	Windows XP 64-bit			
OS/2	DR/DOS	FREEDOS	MFT	MVT	DOS/360
Apples/Macs	Commodores	TI/99s	MVS	OS/390	
Sinclair 1000s	Altairs	Atlas missile guidance systems			
ENIACs	Enigma Decoders	Babbage Difference Engines			
Tablet PCs	Programmable calculators	Cell phones			

DISK SPACE REQUIRED:
    About 5 megabytes for the system.
    With optional pictures, about 15 megabytes.

MINIMUM SYSTEM REQUIREMENTS:
    There are -NO- unusual or demanding requirements to
    install and run these programs as long as you have
    an IBM-compatable PC and DOS.

    Details are listed below.

1. IBM-compatable PC with 2 megabytes of memory, and
    a 486 CPU. (Not tested with a 386 or lower CPU, but
    will probably function correctly. The simulator and
    compiler were compiled at the "386" level.)

2. MS/DOS or PC/DOS.
    DOS can be used from a Windows prompt or in native DOS mode.
    PC/DOS 7.0 and MS/DOS 7.x (Windows/98) were used
    to develop and test the programs, but lower versions
    such as MS/DOS 6 or 5 will probably work fine.

3. A hard drive with about 5 megabytes of free space.
    (Some files are appended-to during operation causing
     them to grow larger with each execution, but they can
     be deleted or renamed, as desired.)

4. An editor to use for writing programs, etc.

The MS/DOS "edit" program works well.
(I use SPF/PC version 4.0.4 [1994 version] from the
CTC Corporation, which also works well).

5. A file viewer to view the disk files.
The program "LIST.EXE" is included for easy viewing
in DOS mode. Shareware. Author: Vernon Buerg.

A BAT file (L.BAT) is included which invokes
LIST.EXE using the format "L filename".
For example,

```
L RR
```

would display the contents of the RR file (the Reel
Reader paper tape file).

```
L SYSLOG
```

would display the contents of the SYSLOG file (the
System Log file).

6. 520k of available conventional DOS memory.

7. XMS memory: None needed.
"XMS memory" spec for Windows DOS icon: None (none needed).

8. DPMI memory: None needed.
"DPMI memory" spec for Windows DOS icon: Auto (none needed).

9. Environment: Whatever your tailored environment needs.
"Environment" spec for Windows DOS icon: 512 bytes (or more).

10. Screen size: 43 or 50 lines.
The field is "Screen initial size" in the Windows DOS icon.
The SIM390 "Status" output and some of the HELP screens are longer
than 25 lines. At a screen size of 25 lines, SIM390 will be difficult
to use, at times.

11. "Windows mode" spec for Windows DOS icon: Window or full screen.

SOFTWARE STATUS: Freeware.
AUTHOR: Dave Morton.
PROGRAMMING LANGUAGE: COBOL.
TRADEMARKS:
"NCR" is a trademark of the National Cash Register Company.
"NCR 390" is a trademark of the National Cash Register Company.

General

The NCR390 was first introduced in 1960 by the National Cash Register company (NCR) and used by hundreds of organizations worldwide. It was also used by the USAF in its Military Pay departments to process payroll transactions and compute pay entitlements. Paychecks were run by a different department using punched card output from the NCR390. It would probably be classified as a mini-computer for its time.

The computer used punched paper tape, punched cards, magnetic ledgers, a built-in typewriter, and a console printer.

According to a magazine ad from that era, the NCR390 sold for $75,000 and leased for $1,850/month.

The company billed it as the "National 390".

It was, according to the ad,
   an "electronic statistician",
   an "electronic accountant",
   an "electronic mathematician",
   an "electronic filing system", and
   an "electronic reporter".

The company's address listed in the ad was:

    The National Cash Register Company
    Dayton 9, Ohio

NCR was evidently so large a presence in Dayton that no street address or PO box was necessary. 5-digit zip codes had also not yet been invented.

Later documents referred to the company as "NCR", rather than "National" or "The National Cash Register Company", and the computer as the "NCR 390". Here, it's referred to as 1 word - the "NCR390".

Note: The model of NCR390 I worked with was slightly different from the one in the ad. The ad shows 2 reel readers (paper tape drives which used 2 reels). We had 1 reel reader and 1 strip reader (no reels, no rewind capability on the strip reader).

The NCR390 was almost purely a "numeric" machine with no internal alphabetic capabilities.
Using the typewriter keyboard, it was possible to type directly onto a ledger or wide roll of paper. The computer also had the capability to read punched cards on a modified IBM-024 keypunch and type the card contents onto a magnetic ledger. Neither of these operations stored any information in the computer's memory. Therefore, all of its computer instructions deal with numbers except for "read punched card - alpha" which is a valid 390 instruction.
But again, reading alphabetic characters from a punched card did not store them in the computer's memory: it merely typed the characters onto a ledger, using the computer's built-in typewriter.

Description of Equipment Which I Worked With

Physical pieces of equipment:
1. Operator's console - with integrated I/O equipment.
2. Paper tape "reel" reader (like a tape drive).
3. Paper tape "strip" reader (no reels, no tape rewind).
4. Card reader/punch - 1 unit - modified IBM keypunch.
5. Paper tape punch.

Support equipment:
1. Paper tape splicer from NCR (desk, splicer, glue, heater).
   The paper tape sometimes had to be spliced, when a section
   of data needed to be inserted or deleted.
   Glue and a splicer machine were used. The splicer machine
   would apply heat and pressure to the splice, curing the splice.

Electronics:
Solid state (no tubes), printed circuits but with
components soldered to the boards (tranistors,
resistors, etc).

Memory type:
Magnetic core, hand wired - probably 9600 cores
(12 x 200 x 4).

Equipment color:
Tan (including the modified IBM keypunch).

Paper tape color:
Green.

Magnetic ledgers:
About 2 feet long by 1.5 feet wide,
if I remember correctly...

Magnetic stripes:
4 vertical magnetic stripes on the back of each
ledger. They contained a "line find" location (for
the next line of printing on a newly-inserted magnetic
ledger), and data. I don't recall how much data the
stripes could hold...

The AF pay records (known as "MPR's" - Military Pay
Records) held 36 words (cells) of information.
according to AF form 1933 "MPR MAGNETIC STRIP DATA",
cells 164 thru 199 were the standard memory locations
for the data to be read into or written from.

Sometimes the data from the magnetic stripes couldn't
be read, and the current information from the pay
record would need to be keyed in, manually. AF form
1933 would then be used for receiving the handwritten
information, field by field, cell by cell, and would
serve as a source document for re-keying.

Console keyboards, buttons and keys:

Numeric keyboard (upper keyboard) -
        12-rows (columns) of 10 numbers, 0 - 9. 120 keys.

PLUS KEY - Enter numeric data or permit the program to continue.

MINUS KEY - Enter negative numbers.

KIE - Keyboard Immediate Entry - unlocked keyboard for
      entering data, similar to the old "request" key
      on a 1052 console for an IBM/360.

```
   MANUAL - Put computer into single-step mode (single
            instruction execution mode).

   SINGLE STEP - Executed 1 instruction.
         This was called "manual mode".

   RESUME PROGRAM - Discontinued manual mode (single step),
         resumed normal operation.

   START 00 - Started program execution at cell 0.
   START 01 - Started program execution at cell 1.
   START 02 - Started program execution at cell 2.
   START 03 - Started program execution at cell 3.
   START 04 - Started program execution at cell 4.
   START 07 - Started program execution at cell 7.

   CARRIAGE TAB RIGHT.
   CARRIAGE RETURN.
   CARRIAGE - Actually, this referred to the bar which
              held the ledger in place - like a typewriter
              bar and rollers holding the paper against
              the platen.
      First depression - Opened to accept a magnetic ledger,
         manually.
      Second depression - Closed for printing on the ledger.

Typewriter keyboard (lower keyboard) -
           Standard typewriter keyboard.
           Text was output in normal typewriter fashion,
           but not stored in the computer's memory or on
           punched cards or tape.


Console Switches:
   ON/OFF Switch. This turned the computer on or off.

   Punch Selector Switch - CARD (Card punch) or TAPE (paper tape punch).


Console Lights:
   RDC      - Read from console (numeric keyboard).

   HALT     - A Halt instruction has been executed.

   MANUAL   - Step mode.

   COMMAND  -  Known as a  "Command" in 390 parlance.
   (Opcode)    Known as an "Opcode"  in sim390 parlance.
               2 binary groups of thin orange lights as:

      x xxxx  <--- opcode lights on console (orange)

      Examples:
       "0 0011"  meant "03" (HALT opcode).
       "1 0100"  meant "14" (SUBTRACT opcode).
       "0 0000"  meant "00" (READ-CONSOLE opcode).


   ADDRESS  -  3 binary groups of thin orange lights as:

      x xxxx xxxx  <--- address lights on console

      Ex:  "1 1001 0011"   meant "current address = 193".
      IE:   1   9   3      in decimal

      If the "Halt" light was on, that would mean
      "halt at address 193" under program control.
      IE, the program has come to a halt at address 193.
```

A program halt was normal: It signaled
end-of-job if the Halt address (documented in the
run book) was correct.
The only way to "end" a program on the NCR390 was
to issue a HALT command (halt operation code).
There was no "wait" opcode or supervisor call,
no wait state, no psw per se, no "end-of-job"
message typed on the console, just a halt in cell
xxx to tell you the job was done.

Paper Tape Punch Buttons:
EOT's - end-of-tape symbols (EOT's) would be punched
until the key was released.

Printing:
The printer was more like a printing calculator.
It printed numbers on a roll of paper called the
"Tally Tape", or on magnetic ledgers when the carriage
was moved to the ledger position.

All lines of print were 12 digits or less, shown as
dollar amounts with commas and a decimal point,
right-justified and zero-suppressed.
This made dollar amounts easy to read, and programs
and other output difficult to read.

Examples:

Number	Printed As
1	.01
400	4.00
70,321	703.21
051564990002	515,649,900.02
123456789012	1,234,567,890.12

With a black and red ribbon, negative numbers could be
optionally printed in red under program control.

Positive Numbers:
The highest positive number which could be stored in a cell
was 89 99 99 99 99 99 or 899,999,999,999.
The above number was printed as:
8,999,999,999.99

Negative Numbers:
Negative numbers were stored as 10's-complement numbers.
The high-order (leftmost) digit was a 9.

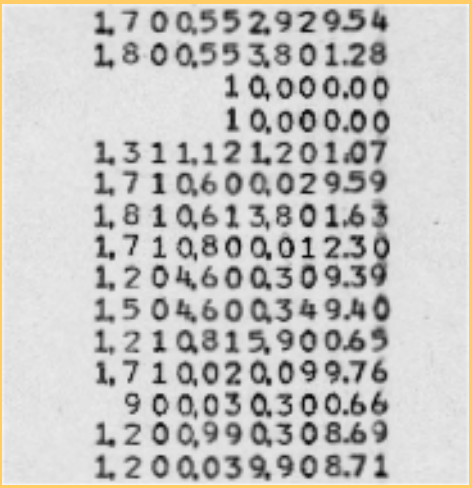
Example: Minus 1 (-1) was stored in a cell as...
99 99 99 99 99 99
Minus 2 (-2) was stored in a cell as...
99 99 99 99 99 98

A negative number was printed with the "credit" sign (CR)
following the number.

Example: Minus 2 (-2) was printed as...
.02CR

With a black and red ribbon, negative numbers could be
optionally printed in red under program control.

The lowest negative number which could be stored in a cell
was 90 00 00 00 00 00 or 900,000,000,000 .
The above number was printed as:
1,000,000,000.00CR



Portion of a Program Printed on a Tally Tape.
All numbers are "zero suppressed" to the left of the
decimal point, and expressed as dollars and cents.
Note that the cell numbers of the program
are not printed next to the instruction.
The programmer had to write the cell numbers
on the printout.

Digit Positions:
   Position numbers for each digit in a cell were numbered
   12 to 1, left to right.

   IE: 12 11 10 9 8 7 6 5 4 3 2 1.

Operating System:
   None.

Programming Languages:
   None.
   All computer programs were written using machine language,
   with all addresses hardwired and non-relocatable.

   Note: The PL390C compiler is now available, as of December, 2001,
         from Salzburg Consulting.
         Programs can be coded using symbolics, and 390 program
         addresses can easily be changed by changing the source code
         and recompiling.

Software:
   Software for the Military Pay operation was supplied by the
   Air Force Accounting and Finance Center, 3800 York Street,
   Denver, Colorado.

   These were all stand-alone programs, containing *no*
   operating system.
   The programs were on mylar tape (not paper tape), and
   were blue on one side and red on the other. Mylar tapes
   were extremely tough!

Procedure for running a Military Pay program:
   Mount the official Air Force program on the Reel Reader.
   Setup any other required devices.
   Reset.

```
    KIE.
    Key in something like: 05 15 14 98 00 99.....
    Enter.
    Reset.
    Start-00.
    The program would then run.

Reference Manuals:
    The NCR390 "gold" manual. I don't have one.
    I wish I did...
    All I have is a copy of the page of instructions (opcodes and
    modifiers), printouts, old forms, a magazine ad, some color
    slides and b&w prints from around 1968...

-------------------------------------------------------------
```

.... SIM390 ....

Many people have asked, "Now that the Personal Computer era is firmly established, and we're in at least the 5th generation of PCs, when is someone going to create a simulator for a really **OLD** and **EARLY** computer - like the NCR390 computer?"

Computer fans, the wait is finally over: SIM390 is here!!

"SIM390" is a DOS-based program which runs on a PC and simulates an NCR390 computer, to the extent possible. This program simulates as many NCR390 functions as possible, including:

Numeric keyboard entry (KIE and Read-Console)

PLUS KEY - Simulated as an "Enter" key, which achieves the same effect as a "Plus" key.

MINUS KEY - Negative numbers can be entered as 10's-complement numbers, or numbers preceeded with a minus sign, using the "Enter" key on the PC keyboard.
For other input sources (tape, card, magnetic ledger), negative numbers must be in 10's-complement format only.
For example, -2 can be entered from the keyboard as:

```
-2
```

```
999999999998
```

From tape, etc, -2 can be entered only as:

```
999999999998
```

Authentic print format

Start keys (START 00 through START 07, entered as *00 or S00, etc)

Manual Mode (no automatic program execution)
Instruction Step (when in Manual Mode)
Resume Program (from Manual Mode)

Idle state (Reset key)

Halt state

All 390 devices except magnetic ledgers

Rewind tape on Reel Reader

Unload tape on Reel Reader (rewind and dismount)

Mount new tape named xxx (copies file xxx to Reel Reader or
Strip Reader file)
Mount new cards named xxx (copies file xxx to card file)

New tape (manually-mounted new tape from another DOS prompt)
New cards (manually-replaced new cards in the hopper from
another DOS prompt).

Punching EOT's onto paper tape with the tape punch button

Punching ER's and EW's with printing

Selector switch for punched cards or punched paper tape

Carriage tabbing and returning
Carriage open/close functions
Forms-advance function

Manual or automatic printing and/or punching

Typewriter keyboard entry (manual typing)
Typing via the instruction "Read Card Alphanumerics"

Future.....
Simulate magnetic card operation, if I can locate the
programming specs.

Enhanced functionality supplied by the simulator but not available on the 390 includes:

* Identify a "Job" by coding a Jobname on the input tape (the tape containing the program).
  If a Jobname is specified, all console output from the 390 program will be prefixed with the Jobname.

* A "MOUNT" command for tapes and cards, to copy a file to to the simulated tape drive or card reader.

* Automatic Volume Recognition (AVR) at startup time.
  If tapes or cards are already loaded when the simulator starts, it detects them, and displays information on the console for each device.

* A "System Reader" function for reading programs and Job information from Reel Reader or Strip Reader tapes.
  The user enters "START RDRn" or "S RDRn" to start a System Reader.

* An "Autoloader" capability, eliminating the requirement to enter a 12-digit "Read Paper Tape" command for loading a program.
  Using the Autoloader feature, the machine language code for loading the program into the computer only needs to be entered once - not every time the program is executed.
  The Autoloader function can also be specified to load the first instruction found, load it into cell 0, and turn control over to cell 0 for execution.

* Ability to print in "authentic" mode, or in "program-format" mode. IE, the system will print using commas and periods (authentic mode), or print each cell as pairs of digits (program-format mode).

* A system log file (SYSLOG) to log the execution of the program and console activity.

```
*  Program run-time statistics at end-of-job including
   elapsed time and cpu time.

*  Enhanced runtime checking of program instructions and data
   with a "program check" operation if an error is found.
   For example, "division by zero" is suppressed, reported,
   and causes the program to fail, rather than proceeding with
   incorrect results.
   Attempting to execute the opcode "99", for example, would
   also trigger the program check sequence.

*  Directives to the simulator in the tape input file.
   Examples:
     TRACE ON
     *  comment lines
     blank lines
   Supplying directives in the tape input file with the program
   eliminates having to manually key them in.

*  A "HELP" command and a "PRINT HELP" command.
   The HELP command lists all SIM390 commands and their syntax.
   The PRINT HELP command writes the HELP file to HELPPRT.TXT on disk.

*  A "CLEAR" command to clear all memory cells, setting them to zero.

*  Debugging features such as:
     Display  cells
     Edit     cells
     Display memory in groups (lower, upper, all)
     Take a snapshot dump (save memory and status to disk)
     Display a flow-trace of the last 20 instructions
     Exhibit the contents of 1 cell repeatedly during
        program execution
     Address-stop function (stop at address nnn)
     Address-reference-stop function
     Start a program at **any** address, not just 00, etc.
     Manual mode (step) for single-instruction rate
     Print-step mode - 1 line of printing per
       "Enter key" depression
     Trace instructions on the console
     Log the trace output on disk
     If available, include source code with the trace
     Display the current status of the program

*  CPU-loop detection with optional program resumption.
   This prevents programs from looping endlessly during
   unattended operation.

*  Output-limit detection and warning.
   This feature prevents a malfunctioning program from
   erroneously writing large quantities of data to disk,
   filling the disk drive. The user can cancel the job,
   or permit the operation to continue until the next
   warning.

*  The ability to interrupt a 390 program during execution
   by simply depressing the "shift" key.
   This is especially useful for looping programs, or for
   debugging.

*  The ability to resume a program from where it left off,
   after being interrupted by an external interrupt (intentional)
   or running out of cards in the card reader.

*  An "EOJ" (end-of-job) command and opcode to end a 390 program
   now, from the program or from the console.
   On the NCR390, if you wanted to end a program early, you
   could RESET the computer, or power it OFF.
   If the program wanted to end itself, it would issue a HLT
```

(Halt) command, or rewind the RR tape endlessly, causing the system to freeze.
Neither method was very elegant.
The SIM390 EOJ command is a more user-friendly way of ending a program.

* A "CANCEL" command and opcode (similar to the EOJ command).

* A "VER" command to display the current version of SIM390.

* A "D T" command to display the current date and time.

* A "D A" command to display the names of active jobs.

* A "D U" command to display the names and status of the I/O units.

* SIM390 stats including elapsed time, CPU time, and CPU-busy percentage time.

* The ability to input paper-tape data in "data" format (12 contiguous digits as done with the original 390) or "program" format (6 pairs of digits separated by blanks for ease of viewing, as a SIM390 enhancement).
The "program" format is the same format as described in the programming manual.

* Upper or lower-case console entries.

* Y2K compatability for presentation of the current date and the compile date.
(Date window: 19xx = 60-99. 20xx = 00-59.)

* "Always On" advanced technology (the 390 can't be turned off). The simulator (SIM390) can be ended, but the computer has no power-off switch.

Operational Notes:
Compiler: Microfocus COBOL, 3.2.20, 1994.
The SIM390 source is named "SIM390.CBL".

The SIM390 program can be run from **any** drive in **any** directory.

To run SIM390, at the C:> prompt, enter the following:

=====> C:>sim390 (or sim390.exe)

This will execute the DOS program "SIM390.EXE".
The disk drive can be any disk drive.

The dynamically called program "SIM390H.EXE" is also used by SIM390 (for the "HELP" command).
It is not linked with the main program, and must be located in the same directory as SIM390.EXE .

SIM390H uses the SYSHELP file to display the help, which must also be in the same directory.

390 application programs and data may be located anywhere, but the SIM390 "MOUNT" command (for mounting tapes and cards) expects the programs and data to be located in directory "PGM\".

Input to your 390 program can come from several sources.
1. The Reel Reader (file "RR").
2. The Strip Reader (file "ST").
3. The Card Reader (file "CR").
4. The keyboard via the directive "KIE" from the keyboard.

```
  Note: Magnetic ledgers are not supported at this time
        due to a lack of documentation.


Interrupting a 390 program:
  To interrupt an executing 390 program, do one of the following:

     Depress the left or right SHIFT key or CTRL key or ALT key.

  This is useful in program testing, looping, etc to break out of
  the program, and return control to the SIM390 OS.


Ending a 390 program:
  To end a 390 program which has not halted or needs to
  be terminated for some reason, enter:
     eoj
  ...at a halt or any other stopped state (Idle, KIE, etc.).

Cancelling a 390 program:
  To cancel a 390 program which has not halted or needs to
  be terminated for some reason, enter:
     cancel   or   can
  ...at a halt or any other stopped state (Idle, KIE, etc.).


Ending SIM390 (ending the simulator):
  To end SIM390 and obtain statistics, enter:
     exit    --or--
     endsim, es, quit, bye
  ...at a halt or any other stopped state (Idle, KIE, etc.).


Cancelling SIM390:
  Depress "Ctrl-c" or "Ctrl-break".
  You must have "break on" in DOS mode for this to function.
  To turn breaks on, at a DOS prompt, enter the following
  DOS command:   break on

  Example of turning on breaks:  C:>break on  <ENTER>


Inputs:
   Keyboard
   Reel Reader          RR
   Strip Reader         ST
   Card Reader          CR
   Magnetic ledger      ML
  *New tape or cards   (dynamic name from MOUNT command)
                        Disk file copied to RR/ST/CR for tape mount or
                        new cards
  *System help          SYSHELP

Outputs:
   Console              CRT display
   Punched paper tape   TP
   Punched cards        CP
   Reel Reader          RR - when mounting a new tape
   Strip Reader         ST - when mounting a new tape
   System printer       SYSPRINT
   System typewriter    SYSTYPE
  *System log file      SYSLOG
  *Trace log            SYSTRACE - optional
  *Snapshot dump        SYSSNAP - optional


  * = Additional file for SIM390 usage.
```

Device Equates:

-------390-------	-------PC-------	Record Length --PC--
Numeric keyboard	Keyboard	80
Printer	CRT display	80
Printer	Disk file "SYSPRINT"	81
Typewriter KB	Keyboard	80
Typewriter	CRT display	80
Typewriter	Disk file "SYSTYPE"	81
Reel Reader	Disk file "RR"	80
Strip Reader	Disk file "ST"	80
Card Reader	Disk file "CR"	80
Card Punch	Disk file "CP"	80
Tape Punch	Disk file "TP"	80
Magnetic ledger	Disk file "ML"	80
*New tape or cards	(dynamic name from cmd) Disk file copied to RR/ST/CR for tape mount or new cards. Used by the MOUNT command to mount a new tape on the Reel Reader or Strip Reader, or place new cards in the card reader. Example: M RR,TTT.OBJ	80
*System help	SYSHELP	81
*System log	SYSLOG	81
*Trace log	SYSTRACE	81
*Snapshot dump	SYSSNAP	81

* = Additional file for SIM390 usage.

Note: The MOUNT (M) command copies a file to RR, ST, or CR, and informs SIM390 of the new tape (or cards). The "NEWTAP" (new tape) command tells SIM390 that a new "tape" file has been copied to RR or ST by the operator using DOS or Windows, or that the old tape can be used. Copying the file plus NEWTAP is the equivalent of a MOUNT command. The same is true for the "NEWCARDS" command.

The NEWTAP/NEWCARDS and MOUNT commands OPEN their respective devices. If the tape or card device was unloaded (ULD RR/ST/CR), the file on the device is flushed and must be refreshed.
If a programming error caused the system to close the device, the file is flushed and must be refreshed.
Entering NEWTAP/NEWCARDS without refreshing the file using DOS or Windows will not accomplish anything if the file has been flushed by SIM390. However, a MOUNT command can be issued to mount a new file on the device.

Operating system note for NEWTAP/NEWCARDS:
Under Windows/XP and other operating systems which utilize file protection, a SIM390 file (such as RR) cannot be replaced/refreshed by an outside method, such as copy-and-paste, or the COPY command of MS/DOS when SIM390 is running and the card reader is OPEN. Therefore, with more modern operating systems, copying must occur when the file has been closed via the ULD (UNLOAD) command. If an external refresh of the file is desired, first UNLOAD the file, then copy the new file to the existing one, then issue the NEWTAP/NEWCARDS command. The alternate

solution for refreshing the device with a new tape or set of cards is
to issue a MOUNT command for the device, from the SIM390 console.
For example: M ST,TEST1.OBJ .

The NEWCARDS command can be used to refresh the hopper if the card reader
runs out of cards without resorting to COPY or cut and paste.
The system closes the card reader but does not empty the file in that
case, and will reload whatever cards were present.
Running out of cards is not considered to be a programming error.

Tip: To close a file for doing cut and paste or copying, UNLOAD the device.
Example: ULD RR (unloads the Reel Reader, clears and closes the file).

Tip: To start with fresh input files, UNLOAD all devices prior to shutdown
of SIM390, or after it starts up. UNLOAD will empty the files, and they will
appear on DOS/Windows directory listings as having zero bytes.
For example: CR   0 KB File 12-16-2004 9:30PM.

Program Segmentation:
   Yes.

Called Programs:
   SIM390H.EXE - The "HELP" processor.

TAPE SYMBOLS

SPECIAL SYMBOLS WERE PUNCHED INTO THE PAPER TAPE AND
USED BY THE 390 FOR END-OF-WORD, END-OF-RECORD, AND
END-OF-TAPE.
THESE SYMBOLS ARE REPLACED BY LETTERS FOR SIM390 USE
(IN THE CASE OF "ER" AND "EOT"), AND NOT USED AT ALL
IN THE CASE OF "EW".
IN DOCUMENTATION, THEY ARE REFERRED TO AS "EW, ER,
AND EOT".

```
EW  (END-OF-WORD)  - NOT USED BY SIM390.
                     THIS SYMBOL MEANT THE SAME THING AS
                     "END OF CELL". IE, 1 CELL = 1 WORD.
                     IN SIM390 FORMAT, 2 BLANKS
                     FOLLOWING THE INPUT DATA SIGNIFIES
                     END-OF-WORD.


ER  (END-OF-RECORD)- COLUMNS 1 AND 2
                     MUST CONTAIN THE LETTERS "ER".
     EXAMPLE:   ER [COMMENTS...]

EOT  (END-OF-TAPE) - LOGICAL END-OF-FILE IS DENOTED BY
                     END-OF-TAPE SYMBOLS.
                     EOT'S MAY ALSO BE PRESENT AT THE
                     BEGINNING OF A FILE.
                     COLUMNS 1-3: THE LETTERS "EOT".
      EXAMPLE:   EOT [COMMENTS...]
```

```
   EXAMPLE 1: PORTION OF SIM390 EXECUTABLE PROGRAM EXAMPLE
   -------------------------------------------------------

              OBJECT PROGRAM - COMPILED OR KEYED
              ----------------------------------



EOT
EOT           <=== LEADING EOT SYMBOLS FOR REALISM
EOT

              <=== BLANK LINES PERMITTED AND IGNORED


* THIS IS A COMMENT....  <=== COMMENT LINE
* TRACE OFF     <=== COMMAND COMMENTED OUT
* STEP ON       <=== COMMAND COMMENTED OUT


JOB TICTACA   <=== OPTIONAL JOBNAME
TRACE ON      <=== OPTIONAL DEBUGGING COMMAND


PROGFMT  <=== SPECIFIES "PROGRAM FORMAT" INPUT TO SIM390.
              THIS FORMAT CONSISTS OF A 3-DIGIT PROGRAM ADDRESS
              IN COLUMNS 1-3, FOLLOWED BY THE INSTRUCTION AS
              6 PAIRS OF DIGITS AS SHOWN BELOW.
              THE ADDRESS FIELD IN COLUMNS 1-3 IS TREATED AS A
              COMMENT BY SIM390.

              END-OF-WORD (EW) SYMBOLS NEED NOT BE USED.
              END-OF-WORD IS ASSUMED FOLLOWING EACH 12-DIGIT FIELD.

              END-OF-RECORD (ER) SYMBOLS, WHERE PRESENT, IMMEDIATELY
              FOLLOW ON THE NEXT OUTPUT RECORD IN COLUMN 1.

              END-OF-TAPE (EOT) SYMBOLS BEGIN IN COLUMN 1.



AUTOLOAD 05 10 01 99 00 02  <=== OPTIONAL AUTOLOAD COMMAND
                                 FOR LOADING PROGRAM, INSTEAD
                                 OF KEYING IT IN EVERY TIME.
                                 "S RDR1" OR "S RDR2" CONSOLE COMMANDS
                                 WILL START THE READER, LOAD
                                 THE AUTOLOAD INSTRUCTION, AND HONOR
                                 ALL OTHER COMMANDS, SYMBOLS, AND
                                 DIRECTIVES.


COLUMNS 1-80 - "PROGRAM FORMAT" (SIM390 ENHANCEMENT)
.
.
. . . . . .  COL 1 - ADDRESS FIELD.
. .                  THE ADDRESS IS TREATED AS A *COMMENT* BY SIM390.
. .
. .  . . . . . COL 6 - INSTRUCTIONS AND DATA
. .  .
. .  .
. .  .                . . . . . COL 26 - OPTIONAL COMMENTS
. .  .                .
. .  .                .
. .  .                .
. .  .                .
ADR  INSTRUCTION      OPTIONAL-COMMENTS
```

```
001  09 00 02 03 00 35   CLEAR BOARD. INITIATE NEW-GAME RESET
002  05 14 00 99 02 03   BOARD.  XXX123456789.
003  05 40 00 00 00 01   WORK3
004  00 00 99 99 00 05   RDC: GET SQUARE NUMBER FOR HUMAN ENTRY
005  16 00 99 08 22 25   CHECK FOR VALID REPLY (1 - 9)
006  15 00 12 99 86 13   DETERMINE SQ NBR. FOUND, OR INCREMENT.
007  17 13 94 00 04 25   FOR TYPED PRINTOUT OF COMPUTER WIN

008  00 00 00 00 00 10   CONSTANT 10
009  00 00 00 00 00 01   CONSTANT 1
010  00 00 00 00 09 00   CONSTANT FOR INSERT "A"
011  00 00 00 00 09 00   CONSTANT FOR INSERT "B"
 .
 .
 .
197  17 10 02 00 99 23   COPY...
198  15 03 59 27 20 05   COMPARE...


ER    <=== END-OF-RECORD SYMBOL

EOT   <=== END-OF-TAPE SYMBOLS
EOT
EOT

end of file
```

EXAMPLE 2: PORTION OF 390 PROGRAM EXAMPLE

```
COLS 1-80 - DATA FORMAT...
.       NOTE THAT "TRACE ON", ETC, ARE SIM390 ENHANCEMENTS.
.       THE CELL ADDRESS ON THE RIGHT IS A COMMENT, NOT AVAILABLE
.       ON THE ORIGINAL NCR390.
.
.       END-OF-WORD (EW) SYMBOLS NEED NOT BE USED.
.       END-OF-WORD IS ASSUMED FOLLOWING EACH 12-DIGIT FIELD.
.       END-OF-RECORD (ER) SYMBOLS, WHERE PRESENT, IMMEDIATELY
.       FOLLOW ON THE NEXT OUTPUT RECORD IN COLUMN 1.
.       END-OF-TAPE (EOT) SYMBOLS BEGIN IN COLUMN 1.
.
.

DATAFMT      <=== SPECIFIES "DATA FORMAT" INPUT TO SIM390


TRACE ON     <===
TRACE OFF    <===  SIM390 COMMANDS
TRACE ON     <===


. . . . . . . . . . . . . . COL 1
.
.
.    . . . . . . . . . .  COLS 1-12 - INSTRUCTIONS AND DATA
.    .
.    .
.    .         . . . . .  COL 16 - OPTIONAL COMMENTS
.    .         .
.    .         .
.    .         .
.    .         .

030000000002   1
030000000003   2
```

```
  030000000004   3
  010001100005   4
  010422220006   5
  010025270007   6
  030000000008   7
  000097980009   8
  170097989910   9
  010097990011   10
  010000990012   11
  010400050013   12
  171001002014   13
  150001021516   14
  030000000018   15
  030000000018   16
  000000000000   17
  010000250001   18
  000000000000   19
  0000000000000ER 20
  ER    <=== END-OF-RECORD SYMBOL
  EOT   <=== END-OF-TAPE SYMBOL

  end of file
*============================================================
```

Output from the SIM390 "HELP" command

```
----------  SIM390 COMMANDS  ----------         SYSHELP LVL 08-27-2006

CONSOLE COMMANDS - OPERATION:

   XXX/Y    MEANS   TYPE EITHER 'XXX' OR 'Y'.
   EXAMPLE: HELP/H  MEANS
                    TYPE EITHER 'HELP' OR 'H'.

COMMAND              DESCRIPTION
---------------      ----------------------------------------
HELP/H               DISPLAY HELP INFORMATION
PRINT HELP           WRITE HELP INFO TO FILE FOR PRINTING:  HELPPRT.TXT

VER                  DISPLAY SIM390 PROGRAM VERSION
? OR STATUS          DISPLAY 390 STATUS
D T  OR  DT          DISPLAY CURRENT DATE AND TIME
D A  OR  DA          DISPLAY ACTIVE JOBS
D U  OR  DU          DISPLAY I/O UNITS
D SYS                DISPLAY SYSTEM FILE OUTPUT COUNTS

ENDSIM  OR...        END THE SIMULATOR
  ES/EXIT/QUIT/DONE/END/BYE

EOJ                  END    A 390 JOB
CANCEL/CAN           CANCEL A 390 JOB

JOB XXXXXXXX         GIVE A JOBNAME TO A PROGRAM WHICH WAS
                     ENTERED FROM THE CONSOLE SO THAT PROGRAM
                     RESPONSES ARE EASIER TO FOLLOW.

RESET/RES/R          RESET 390 (SIMULATES THE 390 'RESET' KEY)
KIE/K                REQUEST KEYBOARD ENTRY (KB IMMEDIATE ENTRY)
                       STARTING AT CELL 0, INCREMENTING BY 1.
                       UNLOCKS KEYBOARD FOR DATA ENTRY INTO MEMORY.
                       USE 'RESET' COMMAND TO END INPUT.
                       SEE ALSO 'E' COMMAND (EDIT) TO SPECIFY CELL(S)
                       FOR ENTERING DATA INTO MEMORY.

(START PROGRAM EXECUTION AT CELL...)
  *00 / S00          START 00
  *01 / S01          START 01
  *02 / S02          START 02
  *03 / S03          START 03
  *04 / S04          START 04
  *07 / S07          START 07
  **NNN              START AT ANY ADDRESS (NNN)

STEP/ST  [ON/OFF]    TURN MANUAL MODE ON/OFF. DEFAULT = STEP OFF.
                        'INSTRUCTION STEP' MODE VIA ENTER KEY.
MANUAL/MAN [ON]      TURN MANUAL MODE ON (SAME AS 'STEP ON').
                        'INSTRUCTION STEP' MODE VIA ENTER KEY.
RESUME/RP  [PR]      RESUME PROGRAM FROM MANUAL MODE (STEP OFF)


* NOTE: THE 'MOUNT' COMMAND CAN BE USED IN PLACE OF ULD/copy/NEWxxx.
RWD    [RR]          REWIND REEL READER TAPE. LEAVE RR FILE OPEN.
UNLOAD  XX           (SEE ULD)
ULD RR/ST/CR         UNLOAD TAPE OR CARD FILE, CLOSE AND CLEAR
                        THE FILE.
                        TO OPEN A NEW FILE, USE MOUNT OR
                        NEWTAP/NEWCARDS COMMAND.

NEWTAP  RR           NEW TAPE IS ON REEL  READER. OPEN THE FILE.
                        DO 'ULD RR', COPY FILE TO RR WITH DOS/WIN,
```

```
                        THEN 'NEWTAP RR'.
NEWTAP  ST           NEW TAPE IS ON STRIP READER. OPEN THE FILE.
                        DO 'ULD ST', COPY FILE TO ST WITH DOS/WIN,
                        THEN 'NEWTAP ST'.
NEWCARDS             NEW CARDS ARE IN HOPPER. OPEN THE FILE.
                        DO 'ULD CR', COPY FILE TO CR WITH DOS/WIN,
                        THEN 'NEWCARDS'.

*NOTE: TAPES AND CARDS MOUNTED USING THE 'MOUNT' COMMAND ARE
OBTAINED FROM THE 'PGM' DIRECTORY.
FOR EXAMPLE:  C:\NCR390\SIM\PGM\TTT.OBJ
MOUNT EXAMPLE:  M RR,TTT.OBJ

M RR,XXXXXXXX[.EEE]  MOUNT "TAPE" WITH FILENAME XXXXXXXX[.EEE]
                       IN THE 'PGM' DIRECTORY ON REEL  READER AND
                       OPEN THE 'RR' FILE.
                       TAPE REMAINS MOUNTED UNTIL REPLACED OR
                       UNLOADED.
M ST,XXXXXXXX[.EEE]  MOUNT "TAPE" WITH FILENAME XXXXXXXX[.EEE]
                       IN THE 'PGM' DIRECTORY ON STRIP READER AND
                       OPEN THE 'ST' FILE.
                       TAPE REMAINS MOUNTED UNTIL REPLACED OR
                       UNLOADED.
M CR,XXXXXXXX[.EEE]  MOUNT "CARDS" WITH FILENAME XXXXXXXX[.EEE]
                       IN THE 'PGM' DIRECTORY IN CARD READER HOPPER AND
                       OPEN THE 'CR' FILE.
                       CARDS REMAIN MOUNTED UNTIL REPLACED OR
                       UNLOADED.

PEOT/PE              PUNCH EOT'S ON TAPE (EOT BUTTON ON TP PUNCH)

PTAPE                SET PUNCH SWITCH TO "TAPE" (DEFAULT SETTING)
PCARD                SET PUNCH SWITCH TO "CARDS"

TYPE                 GO INTO TYPING MODE (390 TYPEWRITER).
                        STORE TYPED DATA IN DISK FILE 'SYSTYPE'.
/*                   END TYPING MODE

CAR/C                CARRIAGE OPEN/CLOSE TOGGLE
TAB                  CARRIAGE TAB TO NEXT TAB
RET                  CARRIAGE RETURN TO TAB 0
ACF/FF               ADVANCE CONTINUOUS FORMS (FORM FEED)

PRTAUTH/PRTA         PRINT IN AUTHENTIC MODE (NNN,NNN,...) (DEFAULT)
PRTPROG/PRTP         PRINT IN PGM FORMAT MODE   (NN NN NN NN...)

START/S RDR1         START SYSTEM READER1 FOR AUTOLOAD FUNCTION.
                     RDR1 READS TAPE FROM DEVICE RR (REEL READER).
                     (EXAMPLES: S RDR1 - START RDR1)
                     [NOTE: MEMORY IS CLEARED BEFORE READER BEGINS
                        READING TAPE.]

START/S RDR2         START SYSTEM READER2 FOR AUTOLOAD FUNCTION.
                     RDR2 READS TAPE FROM DEVICE ST (STRIP READER).
                     (EXAMPLES: S RDR2  -  START RDR2)
                     [NOTE: MEMORY IS CLEARED BEFORE READER BEGINS
                        READING TAPE.]

PTC   [ON/OFF]       PROCESS TAPE COMMANDS / DON'T PROCESS TAPE
                       COMMANDS (TAPE COMMANDS IN TAPE INPUT STREAM).
                     DEFAULT = PTC ON.
                     NOTE: "AUTOLOAD" IS ALWAYS HONORED WHEN READ BY
                     A READER, REGARDLESS OF THE PTC SETTING.

*** CONSOLE COMMANDS -  PROGRAMMING/DEBUGGING ***

COMMAND                 DESCRIPTION
--------------------   --------------------------------------
TRACE/TRC/TR/T [ON/OFF] TURN TRACE ON/OFF (TRACE ON SCREEN).
```

```
                          DEFAULT = TRACE OFF.
LOGTRACE/LT  [ON/OFF]   LOG TRACE INFO ON/OFF IN SYSTRACE (DISK).
                          DEFAULT = LOGTRACE OFF.
SOURCE/SRC   [ON/OFF]   TURN SOURCE STMT TRACE ON/OFF (WITH TR OR LT).
                          DISPLAYS PORTION OF SOURCE AFTER INSTR.
                          DEFAULT = SOURCE OFF.

STEPP [ON/OFF]          TURN PRINT-STEP MODE ON/OFF. DEFAULT=STEPP OFF.
                        "ENTER KEY" DISPLAYS 1 PRINT LINE IF "STEPP ON".
RPP                     TURN PRINT-STEP MODE OFF

D NNN [-NNN]            DISPLAY CELL(S) NNN-NNN
E NNN [-NNN]            EDIT    CELL(S) NNN-NNN

CLEAR/CLR               CLEAR MEMORY - SET ALL CELLS TO ZERO

DUMP  [ALL]             DISPLAY ALL CELLS (000-199)
  DUMP L                DISPLAY LOWER MEMORY (000-099)
  DUMP U                DISPLAY UPPER MEMORY (100-199)

SNAP                    SAVE A SNAPSHOT DUMP OF MEMORY IN SYSSNAP

D FLOW                  DISPLAY LAST 20 INSTRUCTIONS
EX [NNN/OFF]            EXHIBIT CELL NNN/OFF DURING EXECUTION
ASTOP [NNN/OFF]         ADDRESS   STOP ON CELL NNN/OFF
RSTOP [NNN/OFF]         REFERENCE STOP ON CELL NNN/OFF

RXX                     RESUME THE PROGRAM AFTER AN EXTERNAL INTERRUPT
                          FROM THE POINT OF INTERRUPTION

*** CONSOLE COMMANDS -  EXTERNAL INTERRUPTS ***

COMMAND                  DESCRIPTION
--------------------     ----------------------------------
(SHIFT/CTRL/ALT KEY)     INTERRUPT 390 PROGRAM
(SHIFT + ENTER KEY)      INTERRUPT PRINTING OR READ-CONSOLE

THE 390 APPLICATION PROGRAM IS INTERRUPTED, AND CONTROL IS
RETURNED TO THE OPERATOR. THE SIMULATOR WILL THEN ACCEPT
CONSOLE COMMANDS.
THIS IS AN "EXTERNAL INTERRUPT".
ANY ONE OF THE KEYS - SHIFT OR CTRL OR ALT - MAY BE USED.

THE 390 PROGRAM CAN BE RESUMED FROM THE NEXT INSTRUCTION
WITH THE "RXX" COMMAND.

FOR CAUSING AN INTERRUPT DURING PRINTING OR READ-CONSOLE,
THE *SHIFT* KEY MUST BE DEPRESSED AT THE SAME TIME AS THE
*ENTER* KEY IS DEPRESSED.


*** VALID COMMANDS, SYMBOLS AND DIRECTIVES IN TAPE INPUT ***
    NOT ALL COMMANDS, ETC, ARE VALID IN TAPE INPUT.
    THESE ARE THE VALID ONES FOR TAPE INPUT.

---TAPE INPUT: VALID COMMANDS---

ITEM            DESCRIPTION
-----------     ----------------------
PRTA/PRTP       PRINT IN AUTHENTIC MODE / PROGRAM FORMAT MODE
PTAPE/PCARD     SET PUNCH SWITCH TO TAPE/CARDS
TRACE...        TRACE ON/OFF (TRACE PROG EXECUTION ON SCREEN)
LOGTRACE...     LOGTRACE ON/OFF (DISK OUTPUT OF TRACE)
SOURCE...       SOURCE   ON/OFF (FOR TRACE OR LOGTRACE)
STEP/MANUAL     MANUAL MODE (INSTRUCTION STEP) ON OR OFF
RESUME          RESUME PROGRAM (MANUAL MODE OFF)
EX NNN/OFF      EXHIBIT CELL NNN DURING PROGRAM EXECUTION, OR OFF
AUTOLOAD...     FOR AUTOLOADING PROGRAMS WITH A SYSTEM READER.
```

```
                FORMAT:  AUTOLOAD RR/ST   OR....
                FORMAT:  AUTOLOAD NN NN NN NN NN NN

                WHERE NN... IS A VALID "READ TAPE" INSTRUCTION
                FOR THAT TAPE DRIVE, SUCH AS "05 10 01 99...".

                "AUTOLOAD RR/ST" READS RR OR ST, AND LOADS THE FIRST
                390 INSTRUCTION FOUND ON THE TAPE INTO CELL 0 AND
                EXECUTES IT.

                CAN BE INVOKED ONLY BY A 'START READER' COMMAND.
                AFTER THE AUTOLOAD COMMAND IS EXECUTED, THE READER
                ENDS, AND THE PROGRAM IS IN CONTROL.
                ANY SUBSEQUENT AUTOLOAD COMMANDS ARE IGNORED UNTIL
                A READER IS STARTED FOR THAT DEVICE.
                AUTOLOAD MUST BE THE *LAST* COMMAND IN TAPE INPUT.

---TAPE INPUT: VALID SYMBOLS---
ER              END-OF-RECORD SYMBOL
EOT             END-OF-TAPE SYMBOL

---TAPE INPUT: VALID DIRECTIVES---
*               COMMENT LINE
(SPACES)        BLANK LINE
JOB XXXXXXXX    JOBNAME (8 CHARS)
PROGFMT         FOR TAPE INPUT FORMAT '12 34 56 78...' (DEFAULT)
DATAFMT         FOR TAPE INPUT FORMAT '12345678...'

*** RUNNING SOME NCR390 PROGRAMS ***

NOTE: ALL PROGRAMS ARE DEVICE-DEPENDENT AND MUST BE RUN ON THE
DEVICE SPECIFIED (RR, ST, OR CR).
'RDR1' READS TAPE FROM THE REEL READER (RR).
'RDR2' READS TAPE FROM THE STRIP READER (ST).


ITEM        COMMAND             DESCRIPTION
----------- ------------------  -------------------------------------
SIMULATOR   C:\NCR>SIM390       EXECUTES SIMULATOR FROM A DOS PROMPT
COMPILER    C:\NCR>COMP xxx     COMPILES PL390 PROGRAM xxx.TXT


---390 APPLICATION PROGRAMS---

HELLOWORLD                      DISPLAYS 'HELLO WORLD' ON CONSOLE.
                                USES REEL READER AND CARD READER.
            M RR,HELLO.OBJ      MOUNT 'HELLO WORLD' PROGRAM TAPE ON RR
            M CR,HELLOCR.TXT    MOUNT CARDS IN CARD READER HOPPER
            S RDR1              START READER TO READ PROGRAM ON RR

TICTACTOE                       PLAYS TICTACTOE. 111111111=END OF GAME.
                                USES REEL READER AND CARD READER.
            M RR,TTT.OBJ        MOUNT TICTACTOE PROGRAM TAPE ON RR
            M CR,TTTCR.TXT      MOUNT CARDS IN CARD HOPPER
            S RDR1              START READER TO READ TTT PROGRAM ON RR

--END OF HELP--
```

How to play TicTacToe using SIM390

Print these instructions for handy reference.

This procedure explains how to play TicTacToe running a program
written for the NCR390 computer, running under SIM390.

The program was written in NCR390 machine language in 1968.
The program name (file name) on disk is RRTTTA.
.....

1. Go to a DOS prompt from Windows or native DOS.
   For example: C:\>

2. Change directories to the directory where SIM390 is located.
   For example: C:\>cd NCR390
                 C:\NCR390>cd TEST
                 C:\NCR390\TEST>

3. Type "TTTA" without the quotes.
   The batch file TTTA.BAT will be executed.
   Program RRTTTA will be loaded.

4. Follow the instructions on the screen.
   The human always moves first.

   Square numbers are: 1 2 3
                        4 5 6
                        7 8 9

5. To display the program's status, type "?" or "STATUS".

   To display memory, type "DUMP [ALL]" or "DUMP L" or DUMP U".

   To display 1 cell, type "D nnn" (cell number).

   To edit   1 cell, type "E nnn" (cell number), and type the new
   contents of the cell.

   To single-step the program using the Enter key, type "STEP ON".

   To display a trace of the program as it executes, type "TRACE ON".

   For help, type "H" or "HELP".

6. To end the 390 program, type "EOJ".

7. To end the SIM390 simulator, type "EXIT".

8. SIM390 will write output to SYSLOG and SYSPRINT.
   After SIM390 has ended, to view SYSLOG, SYSPRINT, type:
     L SYSLOG
     L SYSPRINT

   To view the contents of the Reel Reader (file RR), type:
     L RR

```
   To view the contents of the Card Reader (file CR), type:
     L CR

 ========================================================


C:>cd ncr390\test     <==== CHANGE DIRECTORIES TO THE SIM390 PROGRAM
C:\NCR390\TEST>ttta   <==== KEY "TTTA" TO EXECUTE THE TICTACTOE BAT FILE
-----------------------------------------------
SIM390 STARTED AT  10-27-2003  12:19:27
SIM390 VERSION 1.3N
-----------------------------------------------

SIM390 TO INTERRUPT 390 PROG, PRESS SHIFT, CTRL, OR ALT.
SIM390 FOR HELP, TYPE 'H' OR HELP.
SIM390 READY.
SIM390 IDLE.
START RDR1  <==== START READER1 (REEL READER) TO READ IN THE PROGRAM.
SIM390 READER STARTED ON TAPE DRIVE RR.
SIM390 RR: JOB TICTACA    JOBNAME (TICTACA IS TIC-TAC-TOE WITH AUTOLOA
SIM390 JOB TICTACA  STARTED.
SIM390 RR: PROGFMT        DATA IN THIS FILE IS IN "PROGRAM FORMAT" (DI
SIM390 PROGRAM INPUT FORMAT SELECTION ACCEPTED.
SIM390 RR: PRTA           PRINT IN "AUTHENTIC FORMAT" MODE
SIM390 PRINT-AUTHENTIC MODE ACCEPTED.
SIM390 RR: AUTOLOAD 05 10 01 99 00 02   FORMERLY KIE ENTRY BY OPERATOR
SIM390 EXECUTING PROGRAM VIA AUTOLOAD...

SIM390 READING TAPE ON REEL READER.
SIM390 READING TAPE ON REEL READER.
SIM390 REWINDING TAPE ON REEL READER.
TICTACA           (000)
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
5   <==== YOU SELECT SQUARE 5 ON THE TIC-TAC-TOE BOARD
TICTACA           (099)              .05  <==== THE COMPUTER ECHOS YOUR MOVE
TICTACA           (009)              .01  <==== THE COMPUTER RESPONDS WITH SQ 1
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
1   <==== YOU SELECT SQUARE 1 ON THE BOARD
TICTACA           (099)              .01  <==== SQUARE 1 AGAIN - ALREADY OCCUPIED
TICTACA           (098)     1,234,567.89  <==== PICK A DIFFERENT SQUARE
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
2
TICTACA           (099)              .02
TICTACA           (139)              .08
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
3
TICTACA           (099)              .03
TICTACA           (138)              .07
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
4
TICTACA           (099)              .04
TICTACA           (140)              .09  <==== THE MACHINE'S WINNING MOVE
TICTACA           (021)     1,111,111.11  <==== THE MACHINE WON
TICTACA           (002)     3,111,103.33  <==== SQUARES OCCUPIED AND VACANT
TICTACA  HALT IN CELL 118. PRESS ENTER TO CONTINUE...
TICTACA           (000)
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
?
SIM390 -------------------------------------------
SIM390 STATUS  AT  10-27-2003  12:20:26
SIM390 STARTED AT  10-27-2003  12:19:27
SIM390 VERSION = 1.3N  COMPILED ON SEP 30, 2003, 13.18.12
SIM390 JOBNAME = TICTACA
SIM390 SYSTEM STATE = RDC
SIM390 SOURCE DISPLAY = OFF
SIM390 TRACE DISPLAY = OFF
SIM390 LOGTRACE = OFF
SIM390 EXHIBIT = OFF
```

```
SIM390 REFERENCE STOP = OFF
SIM390 RESUME ADDRESS = NONE.
SIM390 MANUAL MODE (SINGLE-STEP) = OFF
SIM390 PRINT-STEP MODE = OFF
SIM390 PRINT MODE = AUTHENTIC
SIM390 PROCESS TAPE COMMAND INPUT = ON
SIM390 PUNCH SWITCH POSITION = TAPE
SIM390 CARRIAGE = CLOSED, POSITION = TAB 0
SIM390 CPULOOP-OPS-CT-WARN    =   7,000,000
SIM390 DISK OUTPUT BYTE COUNT =       7,776
SIM390 DISK OUTLIM-CT-WARN    =  25,000,000
SIM390 CURRENT-INST = (004) 00 00  99  99  00  05
SIM390 EXTNDED-INST = (004) 00 00 099 099 000 005
SIM390 RR FILE STATUS: OPEN = Y, RR POSITION =        1
SIM390 ST FILE STATUS: OPEN = Y, ST POSITION =        1
SIM390 CR FILE STATUS: OPEN = Y, CR POSITION =        1
SIM390 TP FILE STATUS: OPEN = Y, TP POSITION =        1
SIM390 CP FILE STATUS: OPEN = Y, CP POSITION =        1
SIM390 INSTRUCTION COUNT =       1,188
SIM390 ELAPSED TIME =     59.5900 SECS =    0.99 MINS
SIM390 CPU TIME     =      0.0003 SECS =    0.00 MINS
SIM390 RATE = 960,000 INST/CPU SEC
SIM390 ----------------------------------------------

TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
DUMP ALL
SIM390 DUMP REQUEST ACCEPTED.
         (000)  00 00 00 00 00 00
         (001)  09 00 02 03 00 35
         (002)  00 00 00 00 00 00
         (003)  00 00 00 00 00 00
         (004)  00 00 99 99 00 05
         (005)  16 00 99 08 22 25
         (006)  15 00 12 99 86 13
         (007)  17 13 94 00 04 25
         (008)  00 00 00 00 00 10
         (009)  00 00 00 00 00 01
         (010)  00 00 00 00 09 00
         (011)  00 00 00 00 09 00
         (012)  00 00 00 00 00 01
         (013)  12 00 02 03 08 20
         (014)  12 01 21 02 08 97
         (015)  00 00 00 00 00 01
         (016)  12 00 02 03 08 20
         (017)  12 01 21 02 08 97
         (018)  18 10 15 12 02 19
         (019)  09 00 03 03 00 04
         (020)  15 10 03 00 89 14
         (021)  00 01 11 11 11 11
         (022)  15 10 99 00 06 25
         (023)  15 01 02 24 26 49
         (024)  00 00 00 01 00 00
         (025)  01 00 99 99 14 04
                Etc....

         (197)  17 10 02 00 99 23
         (198)  15 03 59 27 20 05
         (199)  00 00 00 00 00 00

TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...

TICTACA           (099)              .00
TICTACA           (099)              .00
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
D 001
          (001)  09 00 02 03 00 35
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
HELP
TICTACA  (004): RDC INTO CELL 099.  AWAITING REPLY...
EOJ
SIM390 JOB TICTACA  ENDED VIA CONSOLE COMMAND AT 10-27-2003  12:21:33.
SIM390 IDLE.
```

```
 SIM390 ENDSIM COMMAND ACCEPTED
 SIM390 REEL-READER RECORDS  (RR) =       477
 SIM390 STRIP-READER RECORDS (ST) =         0
 SIM390 CARD-READER RECORDS  (CR) =         0
 SIM390 INPUT RECORDS =       477
 SIM390 TAPE-PUNCH RECORDS   (TP) =         0
 SIM390 CARD-PUNCH RECORDS   (CP) =         0
 SIM390 ERRORS =         0
 SIM390 INSTRUCTION COUNT =      1,192
 SIM390 ELAPSED TIME =    129.0700 SECONDS =    2.15 MINUTES
 SIM390 CPU TIME     =      0.0008 SECONDS =    0.00 MINUTES
 SIM390 CPU BUSY     =     0.00%
 SIM390 STARTED AT 10-27-2003  12:19:27
 SIM390   ENDED AT 10-27-2003  12:21:36.

C:\NCR390\TEST>L SYSLOG
C:\NCR390\TEST>L SYSPRINT
C:\NCR390\TEST>L RR
C:\NCR390\TEST>L CR
```

How to enter and run a program using SIM390

This procedure explains how to enter and run a program coded in
NCR390 machine language, and run it under SIM390.

The program is *NOT* saved.

This is for native **MACHINE LANGUAGE** programs - not
programs written in the PL390 symbolic language.
.....

1. Go to a DOS prompt. For example: C:\>

2. Change directories to the directory where SIM390 is located.
   For example: C:\>cd NCR390\TEST
                 C:\NCR390\TEST>

3. Type "SIM390" without the quotes.

4. Type "K" or "KIE".
   This will permit Keyboard Immediate Entry.
   The system will respond with:

      SIM390 (000): KIE - CELL 000

5. Enter your program, using 1 to 12 digits per cell, followed by
   the Enter key.
   The KIE routine will begin with cell 0, and increment 1 cell
   at a time, each time you press the Enter key.

   To help prevent unwanted loops, use a HALT instruction (Opcode 03)
   at the end of your program.

   Example: This program will add 2 entered numbers and print the
            result. It will then halt.
            Cells 18 and 19 will hold the 2 numbers.
            Cell 20 will hold the sum of the 2 numbers.
            Cell 00 reads the console for the 2 numbers
                     (operator entry).
            Cell 01 adds the 2 numbers, placing the sum in cell 20.
            Cell 02 prints the sum located in cell 20.
            Cell 03 halts the program. Press Enter to continue.
                     The next address after the HALT is cell 00.

          Cell Data
          ---- ------------
           000 000018190001   (00 00 18 19 00 01)
           001 170018192002   (17 00 18 19 20 02)
           002 010020201403   (01 00 20 20 14 03)
           003 030000000000   (03 00 00 00 00 00)

6. When finished, type "R" or "RESET".

7. Type "*nn" to run your program, where nn is the starting cell.
   NN must be one of the following: 00, 01, 02, 03, 04, or 07.

   To run the program above, key in "*00":
              SIM390 IDLE.
              *00

Ctrl key, or the Alt key, or the Shift key.

To display the program's status, type "?" or "STATUS".

To display memory, type "DUMP [ALL]" or "DUMP L" or DUMP U".

To display 1 cell, type "D nnn" (cell number).

To edit 1 cell, type "E nnn" (cell number), and type the new contents of the cell.

To single-step the program using the Enter key, type "STEP ON".

To display a trace of the program as it executes, type "TRACE ON".

For help, type "H" or "HELP".

9. To end the 390 program, type "EOJ".

10. To end the SIM390 simulator, type "EXIT".

Example:

```
-------------------------------------------------
SIM390 STARTED AT  09-03-2002  07:26:12
SIM390 VERSION 1.3C
-------------------------------------------------

SIM390 TO INTERRUPT 390 PROG, PRESS SHIFT, CTRL, OR ALT.
SIM390 FOR HELP, TYPE 'H' OR HELP.
SIM390 READY.
SIM390 IDLE.
R
SIM390 RESET COMMAND ACCEPTED
SIM390 IDLE.
K
SIM390 (000):   KIE - CELL 000
000018190001
          (000)       181,900.01
SIM390 (001):   KIE - CELL 001
170018192002
          (001) 1,700,181,920.02
SIM390 (002):   KIE - CELL 002
010020201403
          (002)   100,202,014.03
SIM390 (003):   KIE - CELL 003
030000000000
          (003)   300,000,000.00
SIM390 (004):   KIE - CELL 004
R
SIM390 RESET COMMAND ACCEPTED
SIM390 IDLE.
D 000-004
          (000)  00 00 18 19 00 01
          (001)  17 00 18 19 20 02
          (002)  01 00 20 20 14 03
          (003)  03 00 00 00 00 00
          (004)  00 00 00 00 00 00
SIM390 IDLE.
*00
SIM390 START 00
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
11
          (018)              .11
SIM390 (000): RDC INTO CELL 019.  AWAITING REPLY...
1
          (019)              .01
          (020)              .12
```

```
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
25
         (018)             .25
SIM390 (000): RDC INTO CELL 019.  AWAITING REPLY...
26
         (019)             .26
         (020)             .51
SIM390 HALT IN CELL 003. PRESS ENTER TO CONTINUE...
?
SIM390 ------------------------------------------
SIM390 STATUS  AT  09-03-2002  07:28:20
SIM390 STARTED AT  09-03-2002  07:26:12
SIM390 VERSION = 1.3C  COMPILED ON AUG 05, 2002, 05.04.17
SIM390 JOBNAME = (NONAME)
SIM390 SYSTEM STATE = HLT
SIM390 TRACE DISPLAY = OFF
SIM390 LOGTRACE = OFF
SIM390 EXHIBIT = OFF
SIM390 ADDRESS STOP = OFF
SIM390 REFERENCE STOP = OFF
SIM390 MANUAL MODE (SINGLE-STEP) = OFF
SIM390 PRINT-STEP MODE = OFF
SIM390 PRINT MODE = AUTHENTIC
SIM390 PROCESS TAPE COMMAND INPUT = ON
SIM390 PUNCH SWITCH POSITION = TAPE
SIM390 CARRIAGE = CLOSED, POSITION = TAB 0
SIM390 CPULOOP-OPS-CT-WARN    =   7,000,000
SIM390 DISK OUTPUT BYTE COUNT =       8,586
SIM390 DISK OUTLIM-CT-WARN    =  25,000,000
SIM390 CURRENT-INST = (003) 03 00  00  00  00  00
SIM390 EXTNDED-INST = (003) 03 00 000 000 000 000
SIM390 RR FILE STATUS: OPEN = Y, RR POSITION =        1
SIM390 ST FILE STATUS: OPEN = Y, ST POSITION =        1
SIM390 CR FILE STATUS: OPEN = Y, CR POSITION =        1
SIM390 TP FILE STATUS: OPEN = Y, TP POSITION =        1
SIM390 CP FILE STATUS: OPEN = Y, CP POSITION =        1
SIM390 INSTRUCTION COUNT =          8
SIM390 ELAPSED TIME =    128.4700 SECS =    2.14 MINS
SIM390 CPU TIME     =      0.0006 SECS =    0.00 MINS
SIM390 RATE =  13,333 INST/CPU SEC
SIM390 ------------------------------------------

SIM390 HALT IN CELL 003. PRESS ENTER TO CONTINUE...
TR
SIM390 INVALID REPLY. FOUND 'tr'. RE-ENTER.
SIM390 HALT IN CELL 003. PRESS ENTER TO CONTINUE...
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
TR
SIM390 REPLY MUST BE 0 TO 12 DIGITS, OR A SIM390 COMMAND.
SIM390 FOUND 'tr'.
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
TRACE ON
SIM390 TRACE DISPLAY NOW ON
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
2
         (018)             .02
SIM390 (000): RDC INTO CELL 019.  AWAITING REPLY...
3
         (019)             .03
SIM390 TRC (001) 17 00 18 19 20 02    (001) 17 00 018 019 020 002
SIM390 TRC (002) 01 00 20 20 14 03    (002) 01 00 020 020 014 003
         (020)             .05
SIM390 TRC (003) 03 00 00 00 00 00    (003) 03 00 000 000 000 000
SIM390 HALT IN CELL 003. PRESS ENTER TO CONTINUE...
SIM390 TRC (000) 00 00 18 19 00 01    (000) 00 00 018 019 000 001
SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
?
SIM390 ------------------------------------------
```

```
SIM390 STARTED AT  09-03-2002  07:26:12
SIM390 VERSION = 1.3C  COMPILED ON AUG 05, 2002, 05.04.17
SIM390 JOBNAME = (NONAME)
SIM390 SYSTEM STATE = RDC
SIM390 TRACE DISPLAY = ON
SIM390 LOGTRACE = OFF
SIM390 EXHIBIT = OFF
SIM390 ADDRESS STOP = OFF
SIM390 REFERENCE STOP = OFF
SIM390 MANUAL MODE (SINGLE-STEP) = OFF
SIM390 PRINT-STEP MODE = OFF
SIM390 PRINT MODE = AUTHENTIC
SIM390 PROCESS TAPE COMMAND INPUT = ON
SIM390 PUNCH SWITCH POSITION = TAPE
SIM390 CARRIAGE = CLOSED, POSITION = TAB 0
SIM390 CPULOOP-OPS-CT-WARN    =   7,000,000
SIM390 DISK OUTPUT BYTE COUNT =      14,013
SIM390 DISK OUTLIM-CT-WARN    =  25,000,000
SIM390 CURRENT-INST = (000) 00 00  18  19  00  01
SIM390 EXTNDED-INST = (000) 00 00 018 019 000 001
SIM390 RR FILE STATUS: OPEN = Y, RR POSITION =        1
SIM390 ST FILE STATUS: OPEN = Y, ST POSITION =        1
SIM390 CR FILE STATUS: OPEN = Y, CR POSITION =        1
SIM390 TP FILE STATUS: OPEN = Y, TP POSITION =        1
SIM390 CP FILE STATUS: OPEN = Y, CP POSITION =        1
SIM390 INSTRUCTION COUNT =          13
SIM390 ELAPSED TIME =    173.5700 SECS =    2.89 MINS
SIM390 CPU TIME     =      0.0609 SECS =    0.00 MINS
SIM390 RATE =     213 INST/CPU SEC
SIM390 -------------------------------------------

SIM390 (000): RDC INTO CELL 018.  AWAITING REPLY...
EOJ
SIM390 JOB (NONAME) ENDED VIA CONSOLE COMMAND AT 09-03-2002  07:29:16.
SIM390 IDLE.
ES
SIM390 ENDSIM COMMAND ACCEPTED
SIM390 REEL-READER RECORDS  (RR) =         0
SIM390 STRIP-READER RECORDS (ST) =         0
SIM390 CARD-READER RECORDS  (CR) =         0
SIM390 INPUT RECORDS =         0
SIM390 TAPE-PUNCH RECORDS   (TP) =         0
SIM390 CARD-PUNCH RECORDS   (CP) =         0
SIM390 ERRORS =         0
SIM390 INSTRUCTION COUNT =         13
SIM390 ELAPSED TIME =    186.9700 SECONDS =    3.11 MINUTES
SIM390 CPU TIME     =      0.0611 SECONDS =    0.00 MINUTES
SIM390 CPU BUSY     =     0.03%
SIM390 STARTED AT 09-03-2002  07:26:12
SIM390   ENDED AT 09-03-2002  07:29:19.
```

Note on numeric values: Negative numbers begin with a "9" in
the first digit position. For example, 999,999,999,999 = -1 on
the NCR390 (and SIM390).

Note: Cell 0 can be used by any program. However, if using the
PL390C compiler, the assumed beginning address of the program
is cell 1. Whether loading such programs manually or automatically,
loading must begin at cell 1 unless an ORG statement has been used
to offset the load address to a higher address.
This is simply to avoid clobbering cell 0, which was used for
loading the program.

How to code, save, and run a program using SIM390

This procedure explains how to enter and run a program coded in
NCR390 machine language, and run it under SIM390.

The program is *SAVED* for later modification and re-use.

This is for native **MACHINE LANGUAGE** programs - not
programs written in the PL390 symbolic language.
.....

Using any text editor, create a new text file in the NCR390\TEST
directory.

Code a program in NCR390 machine language using the format
described in $DOCSIM2 (see also below).
Columns 1, 6, and 26 are the important column numbers,
The Address (ADR) field is IGNORED by SIM390.
Example:

```
  COLUMNS 1-80 - "PROGRAM FORMAT" (SIM390 ENHANCEMENT)
  .
  .
  . . . . . .  COL 1 - ADDRESS FIELD (TREATED AS A COMMENT BY
  . .                  SIM390)
  . .
  . .  . . . . . COL 6 - INSTRUCTIONS AND DATA
  . .  .
  . .  .
  . .  .                 . . . . . COL 26 - OPTIONAL COMMENTS
  . .  .                 .
  . .  .                 .
  . .  .                 .
  . .  .                 .
  . .  .                 .
  ADR  INSTRUCTION       OPTIONAL-COMMENTS

  001  09 00 08 10 00 02   CLEAR CELLS 8-10.
  002  00 00 08 09 00 03   READ FROM CONSOLE INTO CELLS 8 AND 9
  003  17 00 08 09 10 04   ADD CELLS 8 AND 9, ANSWER IN CELL 10
  004  01 00 08 10 14 05   PRINT CELLS 8, 9, 10
  005  03 00 00 00 00 01   HALT. GO TO CELL 1.
  ER
  EOT
  EOT
```

Save the program (text file).

Copy the program to file RR (Reel Reader).

Go to a DOS prompt (eg, C:\>).

Change directories to the directory where SIM390 is located.
For example: C:\>cd NCR390\TEST
              C:\NCR390\TEST>

Type "SIM390" without the quotes.

```
This will permit Keyboard Immediate Entry.
The system will respond with:

    SIM390  (000): KIE - CELL 000

Enter the 12-digit instruction to load your program from RR or ST
and hit ENTER.
For example:  051001990002  ENTER.

Type "R" or "RESET".

Type "*00" (for START-00) to run your program.
Your program will be loaded via the instruction you entered into
cell 00 via the KIE routine.
For example: SIM390 IDLE.
             *00

If you need to interrupt the program (looping, etc), depress the
Ctrl key, or the Alt key, or the Shift key.

To display the program's status, type "?" or "STATUS".

To display memory, type "DUMP [ALL]" or "DUMP L" or DUMP U".

To display 1 cell, type "D nnn" (cell number).

To edit    1 cell, type "E nnn" (cell number), and type the new
contents of the cell.

To single-step the program using the Enter key, type "STEP ON".

To display a trace of the program as it executes, type "TRACE ON".

For help, type "H" or "HELP".

To end the 390 program, type "EOJ".

To end the SIM390 simulator, type "EXIT".


-------------------------------------------------------------------
Note on numeric values: Negative numbers begin with a "9" in
the first digit position. For example, 999,999,999,999 = -1 on
the NCR390 (and SIM390).

Note: Cell 0 can be used by any program. However, if using the
PL390C compiler, the assumed beginning address of the program
is cell 1. Whether loading such programs manually or automatically,
loading must begin at cell 1 unless an ORG statement has been used
to offset the load address to a higher address.
This is simply to avoid clobbering cell 0, which was used for
loading the program.
-------------------------------------------------------------------
```

How to code, save, compile and run a program using PL390C and SIM390

HELLO WORLD

This procedure explains how to enter and run a program coded in
the PL390 symbolic language, and run it under SIM390.

The program is SAVED for later modification and re-use.

The BAT files used for compiling are:

   COMP.BAT   - Compile and Link.
   COMPGO.BAT - Compile, Link and Go.

COMP.BAT does not copy the object deck to the RR file.
COMPGO.BAT *does* copy the object deck to the RR file.

We will use COMPGO.BAT.
SIM390 should not be running for COMPGO to work correctly.
....

"HELLO WORLD" Program
----

1. Create a new file named HELLOCR.TXT. Edit the file and enter 1 line
   of text containing the following, starting at column 1:

   HELLO WORLD!!

   This will be your displayed message which will appear on the screen.

2. Using any text editor, create a new text file in the NCR390\TEST
   directory called HELLO.TXT .

   This will be your source program.

3. In the HELLO.TXT file, enter the program shown below.
   Stay within columns 1 through 76.

     ....

     "HELLO WORLD" program in disk file HELLO.TXT:

```
    [Columns]
    [1       8   12                                                  76]
     * THIS PROGRAM TYPES "HELLO WORLD" FROM A CARD.
           PROGNAME PHELLO
           PUNCH JOB JHELLO
           PUNCH PRTA        PRINT IN AUTHENTIC MODE
           PUNCH AUTOLOAD 05 10 01 99 00 01
    [1       8   12                                                  76]
     BEGIN  EQU *     /* THIS IS WHERE THE PROGRAM STARTS - CELL 1
     REWIND RWD       /* REWIND RR
     TYPEIT RCA       /* READ CARD APLHA (TYPE)
     HALT   HLT       /* HALT
     DONE   EOJ       /* END OF JOB
```

4. Save the program source file HELLO.TXT to disk.

5. Go to a DOS prompt. For example: C:\>

6. Change directories to the directory where SIM390 and
   your program are located.

C:\NCR390\TEST>

7. For a compile and execute (compile, link and go):
   Type "COMPGO HELLO" without the quotes. No TXT suffix.

   Example: C:\NCR390\TEST\>COMPGO HELLO

   The compiler will read the file HELLO.TXT and process it.
   The executable output (machine language) called HELLO.OBJ should be
   created by the compiler and copied to the Reel Reader (file RR).

   If the compile is OK, SIM390 will be invoked automatically.

8. You are now running SIM390.

   Mount the cards in the card reader:
   Type "M CR,HELLOCR.TXT" and <ENTER>

   Example: SIM390 IDLE.
             M CR,HELLOCR.TXT <ENTER>

   Type "START RDR1" and <ENTER>.

   Example: SIM390 IDLE.
             START RDR1 <ENTER>

   The "HELLO WORLD" executable program (HELLO.OBJ) will be loaded and executed.

   The message "HELLO WORLD!!" will be read from the card reader (file CR) and
   displayed on the screen.

9. If you need to interrupt the program (looping, etc), depress the
   Ctrl key, or the Alt key, or the Shift key.

   To display the program's status, type "?" or "STATUS".

   To display the system's unit names and status, type "D U".

   To display lower memory, type "DUMP L".

   To display 1 cell, type "D nnn" (cell number).

   To display cells zero thru five, type "D 000-005".

   To edit 1 cell, type "E nnn" (cell number), and type the new
   contents of the cell.

   To single-step the program using the Enter key, type "STEP ON".

   To display a trace of the program as it executes, type "TRACE ON".

   For help, type "H" or "HELP".

10. To end the 390 program, type "EOJ".

11. To end the SIM390 simulator, type "EXIT".

Note on numeric values: Negative numbers begin with a "9" in
the first digit position. For example, 999,999,999,999 = -1 on
the NCR390 (and SIM390).

Note: Cell 0 can be used by any program. However, if using the
PL390C compiler, the assumed beginning address of the program
is cell 1. Whether loading such programs manually or automatically,
loading must begin at cell 1 unless an ORG statement has been used
to offset the load address to a higher address.
This is simply to avoid clobbering cell 0, which was used for

```
---------------------------------------------------------------

     Columns   Usage
     -------   ---------------------------
     1  - 76   Text
     77 - 80   Optional sequence number.
               Printed on the input listing,
               not printed on the "compile"
               part of the listing (1st part).
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     1  -  6   Line Name (optional)
           7   Blank
     8  - 10   Symbolic Operation Code
          11   Modifier for some Symbolics
     12 - 76   Operands and comments
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     8  - nn   Compiler directive (such as "PUNCH")
     nn - 76   Operands and comments
     ..........................................................
     1      8   12
     aaaaaa sss ooooo--->76   Basic line format

     Example:
     PRTTOT PRT TOTAL1 THRU TOTAL9

                13
     aaaaaa sssf ooooo--->76  Line format with Symbolic modifier (f)
               -
     Example:
     PRTTOT PRTM TOTAL1 THRU TOTAL9

        aaaaaa = Line Name
        sss    = Symbolic Operation Code
        f      = Symbolic Modifier
        o      = Operand

     1      8       12 or 13
     name   opcode  operands and comments
```

--- The PL390 Language ---

```
0001  09 00 02 03 00 35   342 BEGIN  CLR BOARD THRU WORK3 GOTO CPYCOD    /*CLEAR BOARD, WORK3
                          343
                          344 *          002
                          345 *          CELL 002 = PROGRAM NI AFTER CELL 0 AUTOLOAD
0002  05 14 00 99 03 03   346 LOADUM RPT INTO 100 THRU 199 AT-END REWIND   /*LOAD UPPER MEMORY
0002                      347 BOARD  --- RENAMES LOADUM     <=== THE TTT BOARD AFTER LOAD DONE
                          348
                          349 *          003
0003  05 40 00 00 00 01   350 REWIND RWD GOTO BEGIN        /*REWIND TAPE ON REEL READER
0003                      351 WORK3  --- RENAMES REWIND    /*RE-USE CELL FOR WORK
                          352
                          353 *          004
0004  00 00 99 99 00 05   354 ENTRY  RDC REPLY             /*GET HUMAN'S MOVE
```

General

The NCR390 computer was programmed in machine language during its lifespan, which extended from 1960 to the early 1970s. There was no language or compiler available.

The PL390 language and compiler were written in 1968/1969 by Dave Morton for the NCR390 computer, for running on an IBM System/360 mainframe computer. In 2001, the compiler was transferred from the mainframe to the PC, and the compiler and language improved.

Although the language and compiler never saw use on the NCR390 computer, they can be used with the NCR390 simulator program, "SIM390".

The compiler (PL390C) is a DOS-based program which translates the PL390 language into NCR390 machine language, using a PC running MS/DOS or PC/DOS (native DOS, or in a Microsoft Windows DOS prompt window). The compiler produces both a listing and an object file. The object file can be read into the NCR390 simulator program and executed.

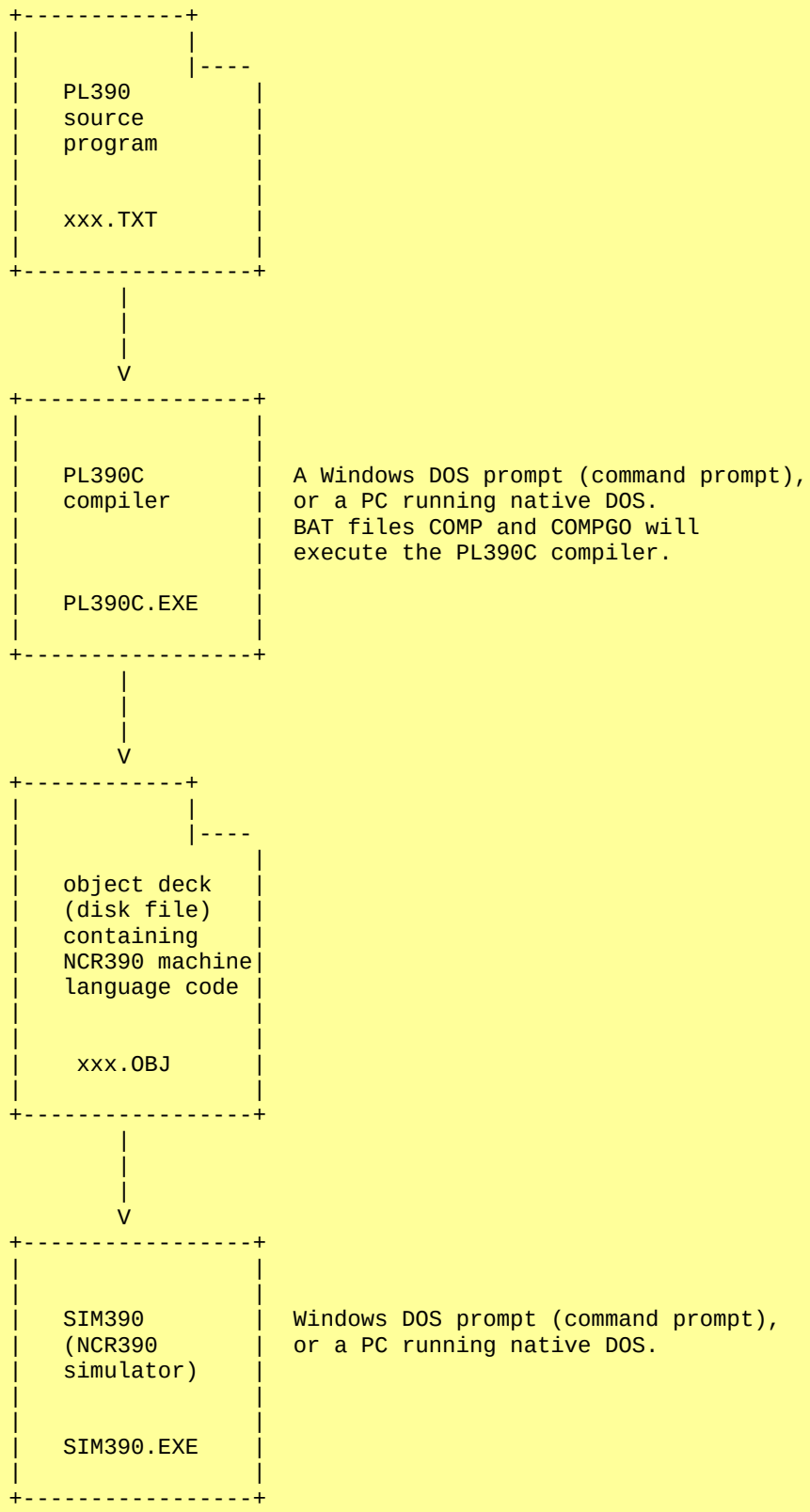


Figure 1.
Program data flow from source deck to NCR390 simulator.

BAT files available for compiling PL390 source programs

COMP.BAT - Compile - SIM390 can be up or down
COMPGO.BAT - Compile and Go - SIM390 should be down

When using COMP.BAT, SIM390 can be up and running. The object deck is copied to name.OBJ and can be loaded by the user from SIM390 by using the MOUNT command.

When using COMPGO.BAT, SIM390 should be down. The compiler copies the object deck to name.OBJ *and* to the Reel Reader (used by SIM390), and then invokes SIM390. If the simulator is running, the object deck cannot be copied unless the RR file is first closed. Also, a second instance of SIM390 will be invoked, causing problems.

Programs can typically be compiled in just a few seconds.

Note that it is not necessary to write a program in PL390 and compile it, in order to run a program on the simulator. Programs written in native NCR390 machine language can be run on the simulator. However, writing a program in PL390 is much easier than than writing a program in machine language.

The PL390 language approximates a blend of assembler language and COBOL, with 1 line of source code generating 1 line of object code (with some exceptions). The language does not support copied code, macros, or multi-line statements.

This document assumes that compiled programs will be run on the SIM390 simulator, and includes operational and programming notes to that effect. Differences between the operation of the SIM390 program and the NCR390 are listed where applicable.

The basic format of the language is:

```
     Columns   Usage
     -------   ---------------------------
     1  - 76   Text
     77 - 80   Optional sequence number.
               Printed on the input listing,
               not printed on the "compile"
               part of the listing (1st part).
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     1  -  6   Line Name (optional)               .
           7   Blank                              .
     8  - 10   Symbolic Operation Code            .
          11   Modifier for some Symbolics        .
     12 - 76   Operands and comments              .
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

     Compiler directives (such as "PUNCH")
     8
     directive  operand  comments (thru col 76)
     ..........................................................
     1      8   12
     aaaaaa sss ooooo------>76   Basic line format
     name   op  operands--->76   Basic line format
     1      8    13
     aaaaaa sssf ooooo------>76   Basic line format
     name   opm  operands--->76   Basic line format
```

```
Example:
1       8    12
PRTTOT PRT  TOTAL1 THRU TOTAL9   (without opcode modifier - PRT)

Example:
1       8     13
PRTTOT PRTM TOTAL1 THRU TOTAL9   (with opcode modifier - PRTM)

   aaaaaa = Line Name
   sss    = Symbolic Operation Code
   f      = Symbolic Modifier
   o      = Operand



1      8       12 or 13
name   opcode  operands and comments
.............................................................

Cols 1      8
            PROGNAME PLXLANG1

Cols 1      8
            PUNCH PRTA        PRINT IN AUTHENTIC MODE
            PUNCH AUTOLOAD 05 10 01 99 00 01   OPTIONAL

Cols 1       8   12
     BEGIN  EQU *     /* THIS IS WHERE THE PROGRAM STARTS
     READT2 RPT 100  THRU 199 LOOPED
     REWIND RWD GOTO INIT     /*REWIND RR
     RW1    ---               /*RW1 RENAMES REWIND
     WORK1  WRK               /*WORK CELL
     *THIS IS A COMMENT STMT
     *   AND THIS IS A COMMENT STMT  /*EMBEDDED COMMENT
     UNUSED                   /*UNUSED CELL
     STINIT EQU *
     INIT   CPY CON1 INTO FACTOR....
            ORG 007           /*ORG TO CELL 7
     ST07   EQU *
     CON1   DAT 123
     CON2   DAT 456
     FACTOR DAT 322
                     Figure 2.

 Example of PL390 source code EXAMPLE1.TXT with column numbers
```

```
                  PL390 COMPILER   VER 1.4U            PLXLANG1

     ADDR     OBJECT CODE     STMT   SOURCE STATEMENT      09-02-2002    01:17

                                1        PROGNAME PLXLANG1
                                2
                                3        PUNCH PRTA        PRINT IN AUTHENTIC
                                4        PUNCH AUTOLOAD 05 10 01 99 00 01
                                5
     0001                       6 BEGIN  EQU *     /* THIS IS WHERE THE PROGR
     0001  05 14 00 99 01 02    7 READT2 RPT 100  THRU 199 LOOPED
     0002  05 40 00 00 00 05    8 REWIND RWD GOTO INIT     /*REWIND RR
     0002                       9 RW1    ---               /*RW1 RENAMES REWI
     0003  00 00 00 00 00 00   10 WORK1  WRK               /*WORK CELL
                               11 *THIS IS A COMMENT STMT
                               12 *   AND THIS IS A COMMENT STMT  /*EMBEDDED
     0004  00 00 00 00 00 00   13 UNUSED                  /*UNUSED CELL
     0005                      14 STINIT EQU *

           WARNING - NI ADDRESS (006) POINTS TO A NON-PROCEDURE LINE.
   W-0005  17 10 07 00 09 06   15 INIT   CPY CON1 INTO FACTOR....

     0006  00 00 00 00 00 00     +
     0007                      16        ORG 007           /*ORG TO CELL 7
     0007                      17 ST07   EQU *
     0007  00 00 00 00 01 23   18 CON1   DAT 123
     0008  00 00 00 00 04 56   19 CON2   DAT 456
     0009  00 00 00 00 03 22   20 FACTOR DAT 322
                               21
                               22
                               23

              1 WARNINGS IN THIS COMPILE
             NO ERRORS   IN THIS COMPILE
```

Figure 3.

Example of PL390 compiled listing from example above, produced by the PL390C compiler. Text has been truncated to fit page.

Note that the beginning address by default is *CELL 1*, not cell 0.
A program may begin at Cell 0 if an ORG statement specifies it.
Namely: ORG 0

One line of PL390 text creates 1 line of object code, with the exception of the following statements: EQU, ORG, and "---". Those statements do not generate object code, and are explained in the Details section. (The ORG statement generates object code of all zeros for cells between its start and end). Additionally, comment statements, blank lines, and compiler directives do not generate object code.
Note: "Object code" in this context means 390 machine code. The PUNCH compiler directive generates a statement which is placed in the object deck, but it may or may not be a 390 machine instruction, and therefore, may or may not be "object code".

Starting Address

The assumed first address is *CELL 1* (not cell 0). If an AUTOLOAD statement is punched, the object code specified will be loaded into cell 0 by the simulator and executed, after a Reader is started to the device containing the object program. Cell 0 can be used by the program by explicitly referring to cell "0" or coding "ORG 0" at the beginning of the program.

Line Names (Cols 1-6)

A line name must begin with an alphabetic (A-Z) character or a dollar sign ($). The remaining characters may be alphabetic, numeric, a dollar sign ($), or a dash (-).

Examples:

```
      GOOD1  ADD A TO B
      $GOOD2 ADD C TO D
      GOOD-3 ADD E TO F
      GOOD$4 ADD G TO H
      D999-A ADD $GOOD2 TO GOOD-3

      9BAD   ADD T TO U  <--- invalid line name
      %BAD   ADD V TO W  <--- invalid line name
      BAD##  ADD X TO Y  <--- invalid line name
      BAD_1  ADD Z TO A  <--- invalid line name
```

Names vs Cell Numbers (Addresses)

Where location names are listed in the instruction formats (name1, name2, etc), cell numbers may be used instead, if desired. Since cell addresses are generated by the compiler (except for ORG), caution should be used in specifying hard-coded addresses. Except for special situations, the use of hard-coded (numeric) addresses should be avoided, and names used instead.

Example using names:

```
      1   BEGIN  RPT INTO CLRA THRU HILM
      2   CLRA   CPY ...
     99   HILM   WRK
```

Example using cell addresses:

```
      1   BEGIN  RPT INTO 2 THRU 99
      2   CLRA   CPY ...
     99   HILM   WRK
```

Example using cell addresses:

```
      1   BEGIN  RPT INTO 002 THRU 099
      2   CLRA   CPY ...
     99   HILM   WRK
```

Example using both:

```
      1   BEGIN  RPT INTO CLRA THRU 99
      2   CLRA   CPY ...
     99   HILM   WRK
```

Example using both:

```
      1   BEGIN  RPT INTO 2 THRU HILM
      2   CLRA   CPY ...
     99   HILM   WRK
```

GOTO is Optional

If a "GOTO" address is not specified in the line, the next line
(next cell) is assumed to be the GOTO address (D address or NI
address) for this line.

Example:

```
      1     BEGIN  RPT INTO 2 THRU 99 GOTO CLRA
      2     CLRA   CPY ...

      1a    BEGIN  RPT INTO 2 THRU 99
      2a    CLRA   CPY ...
```

Line 1 will generate exactly the same object code as Line 1a.

NXTLIN (Next Line)

The reserved word "NXTLIN" means "next line". If desired, it can
be used as an operand in place of a name.
It is not necessary to code "GOTO NXTLIN" as the compiler assumes
the next line to be the GOTO address, unless otherwise specified.

Examples:

```
      ADDA  ADD A TO B GOTO NXTLIN   /* "NXTLIN" NOT NECESSARY
            ADD C TO NXTLIN  GOTO PRINT
      TOT   DAT 31416
      A     WRK
      B     WRK
      C     WRK
      PRINT PRT TOT...
```

Embedded Comments

Embedded comments can be placed anywhere on a line.
There are 2 types of embedded comments:

1. "/*" required.
2. "/*" not required but recommended (trailing comment).

Some instructions (in PL390) do not use a GOTO address, such as
"WRK" and "DAT". A comment can be added after the opcode or operand
without using "/*" in such cases. However, it is safer to always
use "/*" to indicate an embedded comment.

Memory Plane Level Rules

The NCR390 memory is divided into 2 sections, or "planes".

```
    Lower memory plane: Addresses 000-099
    Upper memory plane: Addresses 100-199
```

Some instructions require that 2 of the "name" operands
be located in the same memory plane, ie both operands
must be located in lower memory (cells 0-99) or both in
upper memory (100-199). The compiler checks the validity of
the instruction and issues an error message if the Level Rules
for memory planes are not met.
The programmer must then modify the program so that the Level Rules
are met. Otherwise, the program will not function correctly.

See the document $MACH.PDF which documents the machine instructions
and rules for coding.

Unsupported Features

The PL390 language does not support COPY or INCLUDE statements, or macros.

The language does not support a CALL statement for calling a subprogram. However, one could be coded by hand, theoretically.

There is no Linkage Editor for PL390 programs since there are no CALLed subprograms.

Compiler Output

--Listing

The PL390C compiler produces a listing of the compile in the form of a sequential disk file, with records 124 bytes long.
The name of the listing file is source.PRT where "source" is the filename of the source program.

--Object Deck

The PL390C compiler produces an "object deck" in the form of a sequential disk file, with records 80 bytes long.

The name of the output file is "source.OBJ" where "source" is the filename of the source program.
Example: MYPROG.OBJ which was generated from MYPROG.TXT

5 types of records (lines) are produced:
1. Machine code (object code) for an instruction.
2. An instruction which produces no machine code, and is shown as a comment in the object deck.
   Examples: ORG, EQU.
3. A comment statement from the source code.
4. A comment statement produced by the compiler.
5. A command or directive produced by a PUNCH statement.

.....

Machine Code format:

```
     aaa  nn nn nn nn nn nn   Source
```

Cols 1-3   Address (comment only - ignored by SIM390).
Cols 6-22   Instruction as 6 pairs of digits.
Cols 26-80 First 55 bytes of source code from the source program.

Example of machine code format:

```
     005  17 10 07 00 09 06   INIT   CPY CON1 INTO FACTOR
```

.....

Commented instructions:

```
     *                       Source
```

Cols 1   Asterisk.
Cols 2-25   Blank.
Cols 26-80 Source.

These are produced when there is no object code associated with the instruction.

Example of commented instructions:

```
     *                        BEGIN  EQU *   /* THIS IS WHERE THE PROGRAM STARTS
     *                               ORG 007   /*ORG TO CELL 7
     *                        ST07   EQU *
.....................................
```

Source comments:

```
     *                        Source
```

```
Cols 1      Asterisk.
Cols 2-25   Blank.
Cols 26-80  Source.
```

Example of source comments:

```
     *                        THIS IS A COMMENT...
.....................................
```

Comments from the compiler:

```
     * text
```

```
Cols 1      Asterisk.
Cols 2-80   Text.
```

Example of a comment from the compiler:

```
     *ID TICTACTO OBJECT AUTO 12-01-2004 15:33:41 001139 SRCLINES
```

Note: The compiler automatically produces an identification record
(ID record) as the first line of output in the object deck.

....

Commands and directives in object deck:

```
     cccccccc Operands
```

```
Cols 1-8    Command (PRTA, AUTOLOAD, etc).
Cols 10-80  Operands (command dependent)
```

Example of commands and directives in object deck:

```
     JOB TEST1   TEST COMPILER AND SIMULATOR
     PRTA        PRINT IN AUTHENTIC MODE (NNN,NNN,...)
     AUTOLOAD 05 10 01 99 00 01   OPTIONAL AUTO LOAD
```

```
(blank line) - Blank lines are permitted and are treated as
               comment lines.


* - Comment Statement
      Format (column 1):
         ------------------------------------------------------------
         *comments
         ------------------------------------------------------------

      Examples: * THIS IS A COMMENT STATEMENT.
                *AND  THIS  IS  A  COMMENT STATEMENT....



/*  - Embedded Comment
      Format (column 12 or later):
         ------------------------------------------------------------
         text...     /*Embedded Comment - Column 12 or later
         ------------------------------------------------------------
      Operation:
         Supplies a comment on a line of valid PL390 code after all
         operands have been specified. No PL390 operands may be
         specified after an embedded comment.

      Examples: WORK1  WRK                  /*THE WORK AREA
                ADDUP  ADD WORK1 TO WORK2   /* ADD THE NUMBERS
                DATA1  DAT 999    /* MAX NUMBER FOR WORK1
                A1     ADR *      /* ADDRESS OF HERE
                   /*INVALID COMMENT   <===Invalid



PROGNAME - Supply a name for the program

      Type: Compiler directive.

      Format (column 8):
         ------------------------------------------------------------
                PROGNAME name
         ------------------------------------------------------------

      The PROGNAME statement causes the "name" to be printed on
      the compiled listing and punched into the object deck, for
      program identification.
      The PROGNAME is not the same as the JOB name (if used).
      "Name" can be up to 8 characters long.

                 COL 8
      Examples:      PROGNAME MYPROG
                     PROGNAME TEST23BB



PUNCH - Punch Data into the Object Deck

      Type: Compiler directive.

      Format (column 8):
         ------------------------------------------------------------
                PUNCH text...
         ------------------------------------------------------------
```

```
       The PUNCH statement will punch the contents of text following
       the statement into the object deck at compile time.
       1 blank separates the word "PUNCH" from the text.
       The output will be placed into the "OBJ" file, beginning at
       column 1.
       The output can be read into the SIM390 simulator and processed
       by the simulator.

       For the Autoload feature of SIM390, the PUNCH statement is used
       to create the initial instruction to load the program into memory.
       The initial instruction is loaded by SIM390 into cell 0 if the
       "AUTOLOAD" keyword and program-format machine language
       instruction are present.
       The Autoload feature eliminates the requirement of manually
       keying in the program loading instruction each time the program
       is run. Use of the Autoload feature is optional, and requires
       a Reader to be started to that device.

       Column numbers are fixed.

                        COL 8     14
       Examples:            PUNCH JOB TEST1
                            PUNCH TRACE ON
                            PUNCH PRTA   WE WILL PRINT IN AUTHENTIC MODE
                            PUNCH AUTOLOAD 05 10 01 99 00 02

       Output in      COL 1
       object deck        JOB TEST1
       from examples:     TRACE ON
                          PRTA   WE WILL PRINT IN AUTHENTIC MODE
                          AUTOLOAD 05 10 01 99 00 02

-------------------------------------------------------------



 ACL - Accept Ledger
       Format:
          ------------------------------------------------------------
          [L-name] ACL [GOTO G-name]
          ------------------------------------------------------------
       Operation:
          Accepts and positions a magnetic ledger for
          typing on the ledger.
          No data is read into the computer's memory.
          Note: The SIM390 program treats this Symbolic as a no-op.
                After being positioned, the next logical step was
                to type on the ledger, or the program would read
                a punched card (alpha mode) and cause the typing
                on the ledger, on the NCR390.

                Using the simulator, if manual typing is to follow
                the ACL instruction, a HLT instruction should be
                issued after the ACL. TYPING mode can then be entered
                by keying the command "TYPE" on the console.

                To summarize: To type on the ledger after ACCEPTING
                it, either:
                1. Halt (HLT) and enter typing mode manually using
                   the SIM390 command "TYPE", or...
                2. Read a card alphabetically (RCA) which will
                   display the contents of the card on the screen.
                   On the NCR390, it would type the contents of the
                   card on the ledger.

                Typed data is written to SYSTYPE (a disk file) and
                the Console.

       Level Equality: None
```

```
      Machine Instruction: 06 4S -- -- -- NI

      Program Example:
                GETLED ACL    /*ACCEPT THE LEDGER (NO-OP)
                TYP1   RCA    /*TYPE PUNCHED CARD FROM CARD RDR
                TYPMOR HLT    /*GO INTO TYPING MODE, HERE...
                              /*MANUALLY TYPE MORE, IF DESIRED



ADD - Add  (single or block)

      Format1 (ADD format):
         ------------------------------------------------------------
         [L-name] ADD name1 TO name2 [GOTO G-name]
         ------------------------------------------------------------


      Format2 (ADD format):
         ------------------------------------------------------------
         [L-name] ADD name1 AND/+ name2 GIVING name3 [GOTO G-name]
         ------------------------------------------------------------
      Operation of Format1 and Format2:
         Adds name1 and name2, placing the result in the "TO" address
         or the "GIVING" address.
         The numbers are considered to be signed positive or negative
         integers.
      Level Equality:
         Format 1: None.
         Format 2: Name2 and name3 (B and C addresses)
      Machine Instruction: 17 0R AA BB CC NI

      Examples: ADDONE ADD CON1 TO WRK1
                ADDX   ADD CON1 AND CON2 GIVING CON3


      Format3 (BLA format):
         ------------------------------------------------------------
         [L-name] ADD name1 THRU name2 TO name3 THRU name4
                     [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         This is the equivalent of the "BLA" (Block Add) instruction.
         Adds a block of cells to a second block of cells,
         pair by pair.
         The size of the source block must equal the size of the
         receiving block.
      Level Equality: None
      Machine Instruction: 18 0R AA BB KK NI

      Examples: BLOCKA ADD CON1 THRU CON15 TO WRK1 THRU WRK15



ADR - Address
      Format:
         ------------------------------------------------------------
         [L-name] ADR */name1
         ------------------------------------------------------------
      Operation:
         Generates the address designated by name1 or "*" as a
         3-digit, right-justified number.
         For example, the address of cell 3 is 00-00-00-00-00-03,
         and the address of cell 157 is 00-00-00-00-01-57.
         If "*" is specified, the address generated is that of
         the current location counter (ie, the address of this
         line).
      Level Equality: N/A
```

```
      Machine Instruction: None (data)

      Examples: HERE   ADR *
                WRK9   WRK
                WRK10  WRK
                ADDUP  ADD WRK9 TO WRK10 GOTO TOTALS
                ADRA   ADR WRK9
                ADRB   ADR *
                ADRC   ADR 1
                ADRD   ADR 99
                ADRE   ADR 173
                ADRF   ADR ADDUP
                ADRG   ADR ADRA



CAN - Cancel Job (SIM390 Extended Instruction)
      Format:
         ------------------------------------------------------------
         [L-name] CAN
         ------------------------------------------------------------
      Operation:
         The current job is canceled, and the tape on the reel reader
         (if any) is rewound. SIM390 then enters an IDLE status.
         Level Equality: N/A
         Machine Instruction: 21 00 00 00 00 00

      Examples: CAN1   CAN   /* THIS WILL CANCEL THE SIM390 JOB



CAR - Carriage Control
      Format:
         ------------------------------------------------------------
         [L-name] CAR  FEED/OPEN/TAB/TAB1/TAB2/TAB3/RETURN
                       [GOTO G-name]
         [L-name] CARM FEED/OPEN/TAB/TAB1/TAB2/TAB3/RETURN
                       [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Moves the carriage or performs a related action.

         Operands
         --------
         FEED: Paper feed, no tabbing.
         OPEN: Open the carriage.
         TAB:  Tab to a #1 block.
         TAB1: Tab to a #1 block.
         TAB2: Tab to a #2 block.
         TAB3: Tab to a #3 block.
         RETURN: Return the carriage to a #1 insert.

         For "Advance Continuous Forms, see "FRM".

         If the optional modifier "M" is specified (CARM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
      Machine Instruction: 02 ER -- -- -L NI
         FEED:   02 0R -- -- -L NI
         OPEN:   02 8R -- -- -4 NI
         TAB:    02 2R -- -- -L NI
         TAB1:   02 2R -- -- -L NI
         TAB2:   02 4R -- -- -L NI
         TAB3:   02 5R -- -- -L NI
         RETURN: 02 3R -- -- -L NI

      Examples: CAR1   CAR OPEN
                CAR2   CARM TAB3  /*MANUAL OPERATION
```

```
               CAR3   CAR  RETURN GOTO PRINT1
               CAR4   CAR  FEED   GOTO PRINT2
```

CLR - Clear Cells

Format:

```
   ------------------------------------------------------------
   [L-name] CLR name1 [THRU name2] [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

Clears one or more cells to zero.

Level Equality: Name1 and name2 (A and B addresses)

Machine Instruction: 09 0S AA BB -- NI

Examples:

```
               CLEAR1 CLR WRK4
               CLEAR3 CLR WRK1 THRU WRK3
               DONE   HLT GOTO DONE
               WRK1   WRK
               WRK2   WRK
               WRK3   WRK
               DAT1   DAT 123
               WRK4   WRK
```

CPY - Copy (single or block)

Format1 (CPY format):

```
   ------------------------------------------------------------
   [L-name] CPY name1 TO/INTO name2 [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

Copies name1 to name2.

Level Equality: None

Machine Instruction: 17 1R AA -- CC NI

Examples:

```
               CPYONE CPY CON1 INTO WRK1
               CPYTO  CPY WRK5   TO WRK6
```

Format2 (BLC format):

```
   ------------------------------------------------------------
   [L-name] CPY name1 THRU name2 INTO name3 THRU name4
              [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

This is the equivalent of the "BLC" (Block Copy) instruction.
Copies a block of cells to a second block of cells,
pair by pair.
The size of the source block must equal the size of the
receiving block.

Level Equality: None

Machine Instruction: 18 1R AA BB KK NI

Examples:

```
               BLOCKC CPY CON1 THRU CON3 INTO WRK1 THRU WRK3
               CON1   DAT 111
               CON2   DAT 222
               CON3   DAT 333
               WRK1   WRK
               WRK2   WRK
               WRK3   WRK
```

```
DAT - Data
      Format:
         ------------------------------------------------------------
         [L-name] DAT MINUS/-/nnnnnnnnnnn/
                      (FLOAT-L)/(FLOAT-R)/
                      (VARY-A)/(VARY-B)/(VARY-C)/
                      (VARY-AB)/(VARY-AC)
         ------------------------------------------------------------
      Operation:
         Generates 12 digits of data in this cell.

         Operands
         ----------
         MINUS or - : Indicates a negative number follows.
         nnn...     : A 1- to 12-digit number (contiguous digits).
         (FLOAT-L)  : 000000000100  (01 at "C" position)
         (FLOAT-R)  : 999999999900  (-01 at "C" position)
         (VARY-A)   : 000001000000  (01 at "A" position)
         (VARY-B)   : 000000010000  (01 at "B" position)
         (VARY-C)   : 000000000100  (01 at "C" position)
         (VARY-AB)  : 000001010000  (01 at "A" and "B" positions)
         (VARY-AC)  : 000001000100  (01 at "A" and "C" positions)
      Level Equality: N/A
      Op-Code: None.

      Examples: DATA1  DAT 0
                DATA2  DAT 427
                DATA3  DAT 000123456789
                DATA4  DAT -1   (generates 999999999999)
                DATA5  DAT -6   (generates 999999999994)
                DATA6  DAT MINUS 2  (generates 999999999998)
                DATA7  DAT (VARY-AB) (generates 00 00 01 01 00 00)



DIV - Divide
      Format:
         ------------------------------------------------------------
         [L-name] DIV name1 BY name2 [GIVING name3] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Name1 is divided by name2. If the GIVING operand is not
         specified, the quotient is placed in name2.
         If the GIVING operand is specified, the quotient is placed
         in name3.
         An assumed decimal point for the quotient is located
         between the 4th and 5th positions from the right.
         Ie, "nn,nnn,nnn . nnnn"
         The largest quotient is "99,999,999.9999".
         Division by zero is illegal and causes a simulated program
         check interruption, with the suppression of the operation,
         and the termination of the 390 program.
      Level Equality: Name2 and name3 (B and C addresses).
      Machine Instruction: 19 0R AA BB CC NI

      Examples: DIVID1 DIV WRK1 BY WRK3 GOTO PRTIT
                DIVID2 DIV WRK1 BY DIVSOR GIVING QUOTNT GOTO PRTIT



EOJ - End of Job (SIM390 Extended Instruction)
      Format:
         ------------------------------------------------------------
         [L-name] EOJ [condition code]
         ------------------------------------------------------------
      Operation:
         The current job is ended, and an optional condition code is
         displayed. SIM390 then enters an IDLE status.
```

```
         Any tape on the Reel Reader is left in place and not rewound.
         The condition code can range from 00 to 99, and must
         consist of 2 digits.
         The condition code is placed in the D-Address position.
      Level Equality: N/A
      Machine Instruction: 20 00 00 00 00 cc

      Examples: EOJ1   EOJ 00
                EOJ2   EOJ 12  /*EOJ WITH NON-ZERO CC



EQU - Equate
      Format:
         ------------------------------------------------------------
         L-name EQU */name1/n-nnn
         ------------------------------------------------------------
      Operation:
         Causes L-name to be associated with the current address
         (where "*" is specified), or a name or a numeric value.
         If a name is specified, it must have been previously
         defined (as a line name).
         If a numeric value is specified, it must have a value
         between 0 and 199; 1 to 3 digits are required.
         The address is printed by the compiler, and the line name
         (L-name) can be referred to as having that address or that
         numeric value.
         L-name is required.
         No object code is generated and no cell is created.
      Level Equality: N/A
      Machine Instruction: None

      Examples: HERE   EQU *
                BEGIN  EQU *
                START  EQU BEGIN
                ADDUP1 ADD A TO B
                ADDUP2 ADD B TO C GOTO CELL1
                ADAT   EQU A  <=== Invalid - "A" not previously defined
                A      DAT 7512
                B      DAT 2175
                C      WRK
                BDAT   EQU B
                COPYIT CPY FOUR4 TO HIGH
                CELL1  EQU 1
                FOUR4  EQU 44
                HIGH   EQU 99
                HIGHER EQU 199
                MAX    EQU HIGHER
                END    EQU *



FRM - Advance Continuous Forms
      Format:
         ------------------------------------------------------------
         [L-name] FRM    [GOTO G-name]

         [L-name] FRMM   [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Advances continuous forms via the tractor feed on the
         console.
         This instruction is simulated by displaying 5 blank lines
         on the screen.

         If the optional modifier "M" is specified (FRMM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: None
```

```
      Machine Instruction: 02 9R -- -- -L NI

      Examples: FORM1  FRM
                FORM2  FRMM GOTO TAB3



HLT - Halt CPU Processing
      Format:
         ------------------------------------------------------------
         [L-name] HLT [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Halts CPU processing. Processing will resume after the
         Enter key has been depressed.
      Level Equality: None
      Machine Instruction: 03 -R -- -- -- NI

      Examples: HALT1  HLT
                HALT2  HLT /*WE WILL HALT HERE - WAIT FOR ENTER KEY
                HALT3  HLT GOTO CONTIN



IFF - Compare For Zero
      Compare For Equality
      Compare For Magnitude


      Format1 (Compare For Zero):
         ------------------------------------------------------------
         [L-name] IFF name1 [IS] [NOT] ZERO/ZEROS/ZEROES/0
                      [THEN] GOTO G-name1
                      [ELSE/OTHERWISE GOTO G-name2]
         ------------------------------------------------------------
      Operation:
         Performs a Compare For Zero (CFZ) operation.
      Level Equality: None
      Machine Instruction: 15 1R AA -- NE EQ

      Examples: CFZ1   IFF ABC IS ZERO GOTO XYZ ELSE GOTO QRS
                CFZ2   IFF ABC IS NOT ZERO   GOTO XYZ



      Format2 (Compare For Equality):
         ------------------------------------------------------------
         [L-name] IFF name1 [IS] [NOT] =/EQ/EQUAL/NE [TO]
                      [THEN] GOTO G-name1
                      [ELSE/OTHERWISE GOTO G-name2]
         ------------------------------------------------------------
      Operation:
         Performs a Compare For Equality (CFE) operation.
         Note: The phrase "EQ ZERO" (or EQUAL, etc) will be
               converted to a Compare For Zero (CFZ), even
               if a line in the program has the name "ZERO".
               The phrase "EQ 0" (or EQUAL, etc) will result
               in a compare for equality with cell 0, since
               absolute addressing (0) is being used.

               Rule: For CFZ, code "IFF name1 *IS* ZERO...".
               Do not use =/EQ when doing a Compare For
               Zero, and remember that "=/EQ 0" compares the
               contents of cell 0 to the other operand in a
               Compare For Equality operation.
      Level Equality: B and C addresses.
      Machine Instruction: 15 0R AA BB NE EQ

      Examples: CFE1   IFF ABC IS NOT  EQUAL TO XYZ GOTO WHY
```

```
               CFE2   IFF ABC NE XYZ  THEN GOTO WHY
               CFE3   IFF ABC NOT = XYZ    GOTO WHY ELSE GOTO 3
               CFE4   IFF ABC =  WRK1 GOTO XYZ ELSE GOTO QRS
               CFE5   IFF ABC EQ WRK1 GOTO XYZ OTHERWISE GOTO 99

               CFE6   IFF ABC =  0 GOTO MATCH  /*ABC TO CELL 0
                      (CFE6 contains an unwise expression.
                       This will compare ABC to cell 0.)
                       It is not checking ABC for the value ZERO.)

               CFE7   IFF ABC =  CELL0 GOTO MATCH  /*ABC TO CELL 0
                      (CFE7 contains a  better expression, where
                       "CELL0" is the name of the cell at address 0.
                       CFE7 is comparing ABC to cell 0, and the
                       intent is clear.)



      Format3 (Compare For Magnitude):
         ------------------------------------------------------------
         [L-name] IFF name1 [IS] [NOT]  LT / LESS THAN /  <
                     [THEN] GOTO G-name1
                     [ELSE/OTHERWISE GOTO G-name2]
         [L-name] IFF name1 [IS] [NOT]  GE / GRTEQL /  >=  [TO]
                     [THEN] GOTO G-name1
                     [ELSE/OTHERWISE GOTO G-name2]
         ------------------------------------------------------------
      Operation:
         Performs a Compare For Magnitude (CFM) operation.
         Valid comparisons are "less than" or "greater than or
         equal to" (and their negatives via "not").
         Note: The symbol " < "  is a valid symbol to use for
               "less than".
               The symbol " >= " is a valid symbol to use for
               "greater than or equal to".
      Level Equality: B and C addresses.
      Machine Instruction: 16 0R AA BB LT GE

      Examples: CFM1   IFF ABC IS NOT < XYZ GOTO WHY
                CFM2   IFF ABC GE WRK1 GOTO XYZ ELSE GOTO QRS
                CFM3   IFF ABC >= WRK1 GOTO XYZ ELSE GOTO QRS
                CFM4   IFF ABC NOT >= WRK1 GOTO XYZ ELSE GOTO 99
                CFM5   IFF ABC LT WRK1 GOTO XYZ ELSE GOTO QRS
                CFM6   IFF ABC <  WRK2 GOTO DEF ELSE GOTO TUV



INS - Insert Digits

      Format1:
         ------------------------------------------------------------
         [L-name] INS PIC/PICTURE ddddddddddd OF name1
                     INTO name2 [GOTO G-name]
         ------------------------------------------------------------


      Format2:
         ------------------------------------------------------------
         [L-name] INS POS/POSITION(S) n [THRU n] OF name1
                     INTO name2 [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Copies digits from name1 and inserts them into name2,
         based on the PICTURE or POSITIONS operand.
         Digit positions are numbered 1 thru 12, from right to left.
         Up to 10 digits can be copied and inserted.
         The lowest starting position is position 1, and the highest
         starting position is 10.
         The insertion string must consist of contiguous digits.
```

```
         Insertion proceeds from right to left.

         For Format1, digits are selected for insertion by "X"s in
         the PICTURE, and ignored by dashes. A total of 12 dashes
         and X's must be present in the PICTURE.
      Level Equality: None
      Machine Instruction: 12 HR AA BB -J NI

      Program Example using PIC (PICTURE):
         INSRT1 INS PIC --------XX--   OF WORKA INTO WORKB
         WORKA  DAT     111111111111   <--Source (WORKA)

         WORKB  DAT     000000000000   <--WORKB before INSERT
         WORKB  DAT     000000001100   <--WORKB after  INSERT


      Program Example using POSITIONS:
         INSRT2 INS POSITIONS 7 THRU 12 OF WORKC INTO WORKD
         WORKC  DAT 111111111111   <--Source (WORKC)

         WORKD  DAT 000000000000   <--WORKD before INSERT
         WORKD  DAT 111111000000   <--WORKD after  INSERT

         Postions   .....7654321



LOK - Halt, goto this address if STEP Key depressed (Enter Key)
      Format:
         ------------------------------------------------------------
         [L-name] LOK
         ------------------------------------------------------------
      Operation:
         Halts CPU processing and locks the program.
         The D-address (Next Instruction) is the same as the
         address of the current instruction.
         Operands are not allowed, but embedded comments (/*COMMENT..)
         are permitted.
         If the Enter key is depressed (to simulate depressing
         the STEP key of the NCR390), the program will again
         halt at the same address.
      Level Equality: None
      Machine Instruction: 03 -R -- -- -- NI

      Examples: LOK1   LOK
                LOK2   LOK   /* THIS WILL LOCK THE PROGRAM



MDD - Multiply Dollar Decimal
      Format:
         ------------------------------------------------------------
         [L-name] MDD name1 BY name2 [GIVING name3] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Name1 is multiplied by name2. The product placed in
         name2 if "GIVING" is not specified, and in name3 if
         "GIVING" is specified.
         The product is then shifted right 2 places, and rounded.
         The value in name2 cannot exceed 11 digits.
      Level Equality: Name2 and name3 (B-address and C-address)
      Machine Instruction: 10 0R AA BB CC NI

      Examples:
         MDD1   MDD FACTOR BY WRK1
         MDD2   MDD FACTOR BY WRK1 GIVING WRK2 GOTO PRINT
```

```
MUL - Multiply and Shift
      Format:
         ------------------------------------------------------------
         [L-name] MUL name1 BY name2
                     [SHIFT LEFT NN PLACE(S)]
                     [GOTO G-name]

         [L-name] MUL name1 BY name2
                     [SHIFT RIGHT NN PLACE(S)]
                     [RSIGN / RSIGN ROUNDED]
                     [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Multiplies 2 numbers, and optionally shifts and rounds
         the product.
         Name1 is multiplied by name2 and the product placed in
         name2. The value in name2 cannot exceed 11 digits.
         Shifting of the product is optional.
         "RSIGN" and "RSIGN ROUNDED" pertain to right shifting.
         "RSIGN" means "retain sign" which is the original sign
         of the number before shifting.
         If no shifting or rounding is specified, none is done.
         There is no provision on the 390 for retaining the sign
         or rounding left-shifted products.
         Note: MUS can be used in place of MUL. Identical object
               code is generated.
      Level Equality: None
      Machine Instruction: 11 fR AA BB GG NI

      Examples:
         MULT1  MUL FACTOR BY WRK1
         MULT2  MUS FACTOR BY WRK1 LEFT 04 PLACES GOTO TOTALS
         MULT3  MUL FACTOR BY WRK1 SHIFT RIGHT 2 PLACES RSIGN ROUNDED



MUS - Synonym for "MUL" (Multiply and Shift).
      Can be used in place of MUL.


NOP - No Operation
      Format:
         ------------------------------------------------------------
         [L-name] NOP [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Performs a Compare For Equality (CFE) operation, comparing
         cell 0 to itself. The resulting compare will always be
         "equal", and the implied or specified GOTO address will
         always be taken. This is the D-address or NI-address in
         machine language.
      Level Equality: None
      Machine Instruction: 15 00 00 00 00 NI

      Examples: NOP1   NOP     /* THIS IS A NOP FOR A DELAY
                NOP2   NOP GOTO READ



ORG - Set Location Counter to the Specified Address
      Format:
         ------------------------------------------------------------
                ORG n-nnn
         ------------------------------------------------------------
      Operation:
         Sets the location counter to the specified address nnn,
         and generates cells containing zeros between the current
         address and the ORG address.
```

```
         A line name is not allowed on the ORG statement.
         1 to 3 digits are required for the operand (n-nnn).
         The address must be numeric and be between 0 and 199.
         An ORG to an address lower than the current address
         will cause the compiler to issue a warning message.
         An ORG to address 0 is permitted.
      Level Equality: N/A
      Machine Instruction: None

      Program Example:
                WORK1  WRK         /*THE WORK AREA
                       ORG 7       /*ORG TO ADDRESS 7
                       ORG 25      /*ORG TO ADDRESS 25
                WORK3  WRK         /*WORK3 STARTS AT ADDRESS 25
                       ORG 040     /*ORG TO ADDRESS 40
                       ORG 150     /*ORG TO ADDRESS 150



PER - Punch ER Symbol
      Format:
         ------------------------------------------------------------
         [L-name] PER  [GOTO G-name]

         [L-name] PERM [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Punches ER symbols (end of record) into paper tape or
         cards, depending on the setting of the Punch switch.
         The Punch switch is set to "Paper tape" by default, when
         using SIM390, and can be changed to "Cards" by using the
         the appropriate command.

         If the optional modifier "M" is specified (PERM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
      Machine Instruction: 01 0S -- -- -7 NI  (automatic)
                           01 0S -- -- -3 NI  (manual)
      Examples: PER1   PER
                PER2   PERM       /*MANUAL OPERATION
                PER3   PER  GOTO PRINT1



PRN - Print Numbers as Signed Values
      Format:
         ------------------------------------------------------------
         [L-name] PRN  [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]

         [L-name] PRNM [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         (See also "PRT").
         Print the contents of name1 thru name2 by displaying
         the contents on the screen and writing the line(s) in
         SYSPRINT, using signed values for numbers (positive or
         negative).

         Negative numbers are displayed with "CR" (credit) following
         the number. For example, "123,456.78CR".

         If "(EW)" is specified, punching "EW and data" is added
         to the instruction via the L modifier.

         If "(ER)" is specified, punching "ER and data" is added
         to the instruction via the L modifier.
```

```
         If negative numbers are punched, they are punched in
         "absolute value" format - not as signed numbers. Negative
         numbers cannot be formatted on tape or cards except as 10's
         complement numbers.

         If the optional modifier "M" is specified (PRNM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
      Machine Instruction: 01 0S AA BB 0L NI

      Examples: PRINT1 PRN  ANSWR1
                PRINT2 PRN  ANSWR1 THRU ANSWR2 GOTO PRINT3
                PRINT3 PRNM ANSWR1 THRU ANSWR2 /*MANUAL OPERATION
                PRINT4 PRN  (ER) TOTALS   /*PRINT AND PUNCH
                PRINT5 PRNM (ER) TOTALS   /*PRINT AND PUNCH MANUALLY



PRT - Print Numbers as Absolute Values
      Format:
         ------------------------------------------------------------
         [L-name] PRT  [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]

         [L-name] PRTM [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         (See also "PRN").
         Print the contents of name1 thru name2 by displaying
         the contents on the screen and writing the line(s) in
         SYSPRINT, using absolute values for numbers.

         Negative numbers are displayed as their absolute values.
         For example, "-17" would be printed as "999,999,999,983"
         which is the way it's stored in the 390 memory.

         If "(EW)" is specified, punching "EW and data" is added
         to the instruction via the L modifier.

         If "(ER)" is specified, punching "ER and data" is added
         to the instruction via the L modifier.

         If the optional modifier "M" is specified (PRTM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
      Machine Instruction: 01 0S AA BB 1L NI

      Examples: PRINT1 PRT  ANSWR1
                PRINT2 PRT  ANSWR1 THRU ANSWR2 GOTO PRINT3
                PRINT3 PRTM ANSWR1 THRU ANSWR2 /*MANUAL OPERATION
                PRINT4 PRT  (ER) TOTALS   /*PRINT AND PUNCH
                PRINT5 PRTM (ER) TOTALS   /*PRINT AND PUNCH MANUALLY




PTB - Print and Tab
      Format:
         ------------------------------------------------------------
         [L-name] PTB  [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]

         [L-name] PTBM [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]
         ------------------------------------------------------------
```

```
      Operation:
         Print the contents of name1 thru name2 by displaying
         the contents on the screen and writing the line(s) in
         SYSPRINT, using signed values for numbers (positive or
         negative), and tab to a #1 block.
         This instruction is the equivalent of PRN with tabbing 1
         tab position.

         Negative numbers are displayed with "CR" (credit) following
         the number. For example, "123,456.78CR".

         If "(EW)" is specified, punching "EW and data" is added
         to the instruction via the L modifier.

         If "(ER)" is specified, punching "ER and data" is added
         to the instruction via the L modifier.

         If negative numbers are punched, they are punched in
         "absolute value" format - not as signed numbers. Negative
         numbers cannot be formatted on tape or cards except as 10's
         complement numbers.

         If the optional modifier "M" is specified (PTBM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
      Machine Instruction: 01 2S AA BB 0L NI

      Examples: PTAB1  PTB  ANSWR1
                PTAB2  PTB  ANSWR1 THRU ANSWR2 GOTO PTAB3
                PTAB3  PTBM ANSWR1 THRU ANSWR2 /*MANUAL OPERATION
                PTAB4  PTB  (ER) TOTALS   /*PRINT AND PUNCH
                PTAB5  PTBM (ER) TOTALS   /*PRINT AND PUNCH MANUALLY



PTN - Print and Return Carriage
      Format:
         ------------------------------------------------------------
         [L-name] PTN  [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]

         [L-name] PTNM [(EW)/(ER)] name1 [THRU name2]
                       [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Print the contents of name1 thru name2 by displaying
         the contents on the screen and writing the line(s) in
         SYSPRINT, using signed values for numbers (positive or
         negative), and return the carriage to a #1 insert.
         This instruction is the equivalent of PRN with a return
         of the carriage.

         Negative numbers are displayed with "CR" (credit) following
         the number. For example, "123,456.78CR".

         If "(EW)" is specified, punching "EW and data" is added
         to the instruction via the L modifier.

         If "(ER)" is specified, punching "ER and data" is added
         to the instruction via the L modifier.

         If negative numbers are punched, they are punched in
         "absolute value" format, ie, in 10's complement format.

         If the optional modifier "M" is specified (PTNM), the
         action is manual and requires the Enter key to be
         depressed before the action is taken.
      Level Equality: N/A
```

```
      Machine Instruction: 01 3S AA BB 0L NI

      Examples: PRTN1  PTN  ANSWR1
                PRTN2  PTN  ANSWR1 THRU ANSWR2 GOTO PRTN3
                PRTN3  PTNM ANSWR1 THRU ANSWR2 /*MANUAL OPERATION
                PRTN4  PTN  (ER) TOTALS   /*PRINT AND PUNCH
                PRTN5  PTNM (ER) TOTALS   /*PRINT AND PUNCH MANUALLY



RCA - Read Card Alpha
      Format:
         ------------------------------------------------------------
         [L-name] RCA [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Read the next card image from the card reader file (CR)
         and type it by displaying the card on the monitor and
         writing it to SYSTYPE, then release the card.
         On the NCR390, the computer's typewriter was used to do
         the typing.
         No information is read into the 390's memory cells. The 390
         was used as a "pass thru" device to do the typing.
         Since the entire card is read, no RLS instruction is
         needed to release the card.
      Level Equality: N/A
      Machine Instruction: 04 1S -- -- -- NI

      Examples: RCA1   RCA
                RCA2   RCA GOTO READ2     /*TYPE 80-COLUMN CARD



RCN - Read Card Numerics
      Format:
         ------------------------------------------------------------
         [L-name] RCN [INTO] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Read the next card image from the card reader file (CR)
         and load up to 6 card cells (12-digit groups) into the
         cells named. After the numeric data has been read into
         the 390's cells, a RLS (Release Punched Card) must be
         issued before the next card is read.
         Card images consist of 12 numeric digits followed by 1
         space, from column 1 thru column 78.
            Example of Card Portion, Starting at Column 1:
            111111111101 111111111102 111111111103......06
         To fill 18 cells in the 390's memory, for example, would
         require 3 cards, with each card containing 6 card cells,
         and 3 RCN instructions. Before issuing the 2nd and 3rd
         reads, a RLS instruction would need to be issued prior
         to each read.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 04 0S AA BB -- NI

      Examples: RCN1   RCN WRK1  /* READ INTO 1 CELL
                       RLS       /* RELEASE CARD

                RCN2   RCN WRK1 THRU WRK2   /* READ INTO 2 CELLS
                       RLS                  /* RELEASE CARD

                RCN3   RCN INTO WRK1 THRU WRK6  /* READ INTO 6 CELLS
                       RLS                      /* RELEASE CARD

                RCN4   RCN INTO FLD1 THRU FLD6 GOTO RLS4
                RLS4   RLS GOTO PRTCARD
```

```
RDC - Read Console
      Format:
         ------------------------------------------------------------
         [L-name] RDC  [(EW)/(ER)] [INTO] name1 [THRU name2]
                       [GOTO G-name]

         [L-name] RDCA [(EW)/(ER)] [INTO] name1 [THRU name2]
                       [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Accept operator input from the keyboard into name1 thru
         name2. The keyed data must be 1 to 12 numeric digits.
         If the Enter key is depressed without entering any data,
         zeros are assumed to be the operator entry.
         SIM390 will acknowledge the entry after the Enter key has
         been depressed by printing the cell number where the data
         has been entered, followed by the data.

         Negative numbers are displayed as-is. Ie, if "999999999999"
         is entered, SIM390 will acknowledge the entry by displaying
         "999999999999". Ie, the absolute value is displayed.

         If "(EW)" is specified, punching "EW and data" is added
         to the instruction via the L modifier.

         If "(ER)" is specified, punching "ER and data" is added
         to the instruction via the L modifier.

         If negative numbers are punched, they are punched in
         "absolute value" format, ie, in 10's complement format.

         If the optional modifier "A" is specified (RDCA), the
         action is automatic and does not require the Enter key
         to depressed before the action is taken.

         Warning:
         An automatic operation (RDCA) will clear the specified
         cell(s).
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 00 0S AA BB -L NI

      Examples: RDCON1 RDC  INTO ANSWR1
                RDCON2 RDC  INTO ANSWR1 THRU ANSWR2 GOTO RDCON3
                RDCON3 RDCA ANSWR1 THRU ANSWR2 /*CLEAR CELLS
                RDCON4 RDC  (EW) TOTALS   /*READ AND PUNCH
                RDCON5 RDCA (EW) TOTALS   /*CLEAR AND PUNCH AUTO
                RDCON6 RDC  ANSWR1 THRU ANSWR2
                RDCON7 RDC  (ER) INTO ANSWR1 THRU ANSWR2 GOTO RDC1



REJ - Read and Eject Ledger (not simulated)
      Format:
         ------------------------------------------------------------
         [L-name] REJ [INTO] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         This operation is treated as a HALTING NO-OP by the
         SIM390 program. SIM390 will display the current address
         and instruction, and stop, waiting for the Enter key to
         be depressed.
         Information on how to handle the REJ instruction is lacking.
      Compilation: The REJ instruction is checked for proper syntax
         and compiled according to the NCR390 specifications.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 06 2S AA BB -- NI

      Examples: REJ1   REJ ABC
```

```
               REJ2   REJ INTO ABC
               REJ3   REJ ABC THRU XYZ
               REJ4   REJ ABC THRU XYZ GOTO ADDUP
```

REP - Replicate Cells

Format:

```
   ------------------------------------------------------------
   [L-name] REP name1 END name2 [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

Replicates the data in name1 to every cell up to and including name2. The Block Copy (BLC) instruction is used to accomplish the replication.
The purpose of the Replicate instruction is to propagate 1 cell many times, rather than to copy a block of cells to another block of cells.
The number of cells to replicate cannot exceed 100.

Level Equality: None

Machine Instruction: 18 1R AA BB KK NI

Program Example:

```
               REP1   REP CON1 END LIMIT1 GOTO READTP
               CON1   DAT 887766554433   /*CONSTANT
                      WRK    /*TO BE FILLED WITH CON1
                      WRK    /*TO BE FILLED WITH CON1
                      WRK    /*TO BE FILLED WITH CON1
               LIMIT1 WRK    /*TO BE FILLED WITH CON1
               READTP EQU *
```

RLR - Read fron Ledger Reader (not simulated)

Format:

```
   ------------------------------------------------------------
   [L-name] RLR [INTO] name1 [THRU name2] [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

This operation is treated as a HALTING NO-OP by the SIM390 program. SIM390 will display the current address and instruction, and stop, waiting for the Enter key to be depressed.
Information on how to handle the RLR instruction is lacking.

Compilation: The RLR instruction is checked for proper syntax and compiled according to the NCR390 specifications.

Level Equality: Name1 and name2 (A-address and B-address)

Machine Instruction: 06 3S AA BB -- NI

Examples:

```
               RLR1   RLR ABC
               RLR2   RLR INTO ABC
               RLR3   RLR ABC THRU XYZ
               RLR4   RLR ABC THRU XYZ GOTO ADDUP
```

RLS - Release Punched Card

Format:

```
   ------------------------------------------------------------
   [L-name] RLS [GOTO G-name]
   ------------------------------------------------------------
```

Operation:

This operation pertains to the input file "CR" (Card Reader).
Logically release the card image, and point to the first card cell in preparation for the next RCN card read.
This operation is required only when working with the RCN instruction - not the RCA instruction.

Level Equality: None

```
      Machine Instruction: 04 2S AA BB -- NI

      Examples: RLS1   RLS
                RLS2   RLS GOTO GETCRD



RMG - Read Ledger with Magnetic Line Find (not simulated)
      Format:
         ------------------------------------------------------------
         [L-name] RMG [INTO] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         This operation is treated as a HALTING NO-OP by the
         SIM390 program. SIM390 will display the current address
         and instruction, and stop, waiting for the Enter key to
         be depressed.
         Information on how to handle the RMG instruction is lacking.
      Compilation: The RMG instruction is checked for proper syntax
         and compiled according to the NCR390 specifications.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 06 1S AA BB -- NI

      Examples: RLMAG1 RMG ABC
                RLMAG2 RMG INTO ABC
                RLMAG3 RMG ABC THRU XYZ
                RLMAG4 RMG ABC THRU XYZ GOTO ADDUP



RMK - Read Ledger with Mechanical Line Find (not simulated)
      Format:
         ------------------------------------------------------------
         [L-name] RMK [INTO] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         This operation is treated as a HALTING NO-OP by the
         SIM390 program. SIM390 will display the current address
         and instruction, and stop, waiting for the Enter key to
         be depressed.
         Information on how to handle the RMK instruction is lacking.
      Compilation: The RMK instruction is checked for proper syntax
         and compiled according to the NCR390 specifications.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 06 0S AA BB -- NI

      Examples: RLMEK1 RMK ABC
                RLMEK2 RMK INTO ABC
                RLMEK3 RMK ABC THRU XYZ
                RLMEK4 RMK ABC THRU XYZ GOTO ADDUP



RPS - Read Paper Tape from the Strip Reader
      Format:
         ------------------------------------------------------------
         [L-name] RPS [INTO] name1 [THRU name2] LOOPED/AT-END name3
                      [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Read tape record images from the Strip Reader file (SR)
         into name1 thru name2.
         Either LOOPED or "AT-END name3" must be specified.
         If LOOPED is specified and EOTs are encountered while
         reading, SIM390 will go to the address of this RPT
         instruction and continue reading (thus bypassing the EOTs).
         If "AT-END name3" is specified and EOTs are encountered
         while reading, SIM390 will exit to name3 (the AI address, or
```

Alternate Instruction address).

Under SIM390, EOT symbols consist of the letters "EOT " in positions 1 thru 4 of the tape image record (the leftmost positions).

The GOTO address (NI address) will be taken under 2 conditions:
1. All designated cells have been filled by tape record images.
2. An "ER" (end-of-record) symbol is read. Under SIM390, this is the symbol "ER " in positions 1 thru 3 of the tape image record (the leftmost positions).

Each file record is treated as a 390 word by SIM390, ie, as a 12-digit word. "EW" symbols are not needed to delimit words under SIM390.

The input records can be formatted in "data format" or "program format". The expected format defaults to "PROGFMT" (Program Format). See the SIM390 documentation for tape record formats.

Record Example - Data Format: 000033794215 xxx
Record Example - Program Format: 001 05 10 01 99 23 02 xxx
Note: The 3-digit address used by Program Format is ignored by SIM390.

Under SIM390, several other statement types can be included in the tape input file such as comment lines, data format commands, an AUTOLOAD statement, etc. See the SIM390 documentation for allowable commands in the tape file.

If a reader has been started for the SR file (Strip Reader file), it remains active until an AUTOLOAD statement is read. The text of the AUTOLOAD statement is the machine code for an RPT or RPS instruction. Subsequent RPT/RPS PL390 instructions will also operate normally, since the reader (if started) will have been deactivated by SIM390.

Level Equality: Name1 and name2 (A-address and B-address)
Machine Instruction: 05 3S AA BB AI NI

Examples:
```
RPS1   RPS INTO WRK1
RPS2   RPS INTO WRK1 THRU WRK2 GOTO RPS3
RPS3   RPS WRK1 THRU WRK2 LOOPED
RPS4   RPS WRK1 THRU WRK2 AT-END CALC1
RPS5   RPS WRK1 THRU WRK2 AT-END CALC1 GOTO ERFND
```

RPT - Read Paper Tape from the Reel Reader

Format:

```
[L-name] RPT [INTO] name1 [THRU name2] LOOPED/AT-END name3
             [GOTO G-name]
```

Operation:

Read tape record images from the Reel Reader file (RR) into name1 thru name2.
Either LOOPED or "AT-END name3" must be specified.
If LOOPED is specified and EOTs are encountered while reading, SIM390 will go to the address of this RPT instruction and continue reading (thus bypassing the EOTs).
If "AT-END name3" is specified and EOTs are encountered while reading, SIM390 will exit to name3 (the AI address, or Alternate Instruction address).

Under SIM390, EOT symbols consist of the letters "EOT " in positions 1 thru 4 of the tape image record (the leftmost

```
         positions).

         The GOTO address (NI address) will be taken under 2
         conditions:
         1. All designated cells have been filled by tape record
            images.
         2. An "ER" (end-of-record) symbol is read. Under SIM390, this
            is the symbol "ER " in positions 1 thru 3 of the tape
            image record (the leftmost positions).

         Each file record is treated as a 390 word by SIM390, ie, as
         a 12-digit word. "EW" symbols are not needed to delimit words
         under SIM390.

         Record Example - Data Format:    000033794215   xxx
         Record Example - Program Format: 001  05 10 01 99 23 02   xxx
         Note: The 3-digit address used by Program Format is ignored by
         SIM390.

         Under SIM390, several other statement types can be included
         in the tape input file such as comment lines, data format
         commands, an AUTOLOAD statement, etc. See the SIM390
         documentation for allowable commands in the tape file.

         If a reader has been started for the RR file (Reel Reader
         file), it remains active until an AUTOLOAD statement is
         read. The text of the AUTOLOAD statement is the machine
         code for an RPT or RPS instruction. Subsequent RPT/RPS
         PL390 instructions will also operate normally, since the
         reader (if started) will have been deactivated by SIM390.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 05 1S AA BB AI NI

      Examples: RPT1   RPT INTO WRK1
                RPT2   RPT INTO WRK1 THRU WRK2 GOTO RPT3
                RPT3   RPT WRK1 THRU WRK2 LOOPED
                RPT4   RPT WRK1 THRU WRK2 AT-END CALC1
                RPT5   RPT WRK1 THRU WRK2 AT-END CALC1 GOTO ERFND



RWD - Rewind Paper Tape on the Reel Reader
      Format:
         ------------------------------------------------------------
         [L-name] RWD [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Simulate rewinding the tape on the Reel Reader by pointing
         to the first record on the RR file.

         Note: This is similar to the way the 390 handled the
         operation, but not identical. On the 390, the tape was
         rewound to the first EOT symbol encountered which may
         or may not have been located at the beginning of the tape.

         A second rewind operation on the Reel Reader with a rewound
         tape causes the rewind operation to be ignored unless the
         GOTO address points to this line. In the latter case, the
         instruction is the equivalent of the "Unload" (ULD)
         instruction. The ULD instruction will cause SIM390 to
         simulate the rewinding and unloading of the Reel Reader tape,
         followed by an end-of-job condition.
      Level Equality: N/A
      Machine Instruction: 05 4R -- -- -- NI

      Examples: RWD1   RWD    /*REWIND THE TAPE ON THE RR FILE
                RWD2   RWD GOTO SETUP  /*REWIND, GO SETUP PROGRAM
                RWD3   RWD GOTO RWD3   /*UNLOAD THE RR TAPE
```

```
SHF - Shift and Copy

      Format1 (shift left):
         ------------------------------------------------------------
         [L-name] SHF name1 LEFT nn PLACE(S) [INTO name2]
                     [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Shifts name1 left nn places, and optionally copies it
         into name2.
      Level Equality: None
      Machine Instruction: 13 0R AA BB GG NI

      Examples:
         SHIFT1 SHF FACTOR LEFT  1 PLACE
         SHIFT2 SHF FACTOR LEFT 10 PLACES INTO WRKZ



      Format2 (shift right):
         ------------------------------------------------------------
         [L-name] SHF name1 RIGHT nn PLACE(S)
                     [ABSOLUTE/ABS/RSIGN] [INTO name2]
                      ------------
                     [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Shifts name1 right nn places, and optionally copies it
         into name2.
         By default, the number is treated as an absolute (unsigned)
         number. ABSOLUTE or ABS may be specified, even though
         it is the default.
         "RSIGN" means "retain sign", and may be specified if the
         shifted number should retain its original sign.
      Level Equality: None
      Machine Instruction: 13 FR AA BB GG NI

      Examples:
         SHIFT1 SHF FACTOR RIGHT  1 PLACE
         SHIFT2 SHF FACTOR RIGHT 10 PLACES RSIGN INTO WRKZ



SPS - Space Platen
      Format:
         ------------------------------------------------------------
         [L-name] SPS [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Space the platen 1 line by performing an automatic RDC
         operation, reading zeros into cell 0.

         ***WARNING***
         This instruction clears cell zero.
         The compiler will issue a warning message when this
         instruction is used, as a reminder.

      Level Equality: None.
      Machine Instruction: 00 0S 00 00 04 NI

      Examples: SPACE1 SPS
                SPACE2 SPS GOTO PRINT



SUB - Subtract
      Format:
```

```
         ------------------------------------------------------------
         [L-name] SUB name1 FROM name2 [GIVING name3] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Subtracts name1 from name2, and places the result in
         name2, or in name3 if the GIVING option is used.
         The numbers are considered to be signed positive or negative
         integers.
      Level Equality:
         Without GIVING: Name1 and name2.
         With    GIVING: Name1 and name3.
      Machine Instruction: 14 0R AA BB CC NI

      Examples: SUBONE SUB CON1 FROM WRK1
                SUBX   SUB CON1 FROM CON2 GIVING CON3



SUM - Summarize
      Format:
         ------------------------------------------------------------
         [L-name] SUM name1 THRU name2 GIVING name3 [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Adds name1 thru name2 and places the result in name3.
         Numbers are treated as signed numbers - not absolute values.
      Level Equality: Name1 and name2 (A and B addresses)
      Machine Instruction: 08 0S AA BB CC NI

      Examples: SUM1   SUM WRK1 THRU WRK3 GIVING TOTAL



ULD - Unload Paper Tape from the Reel Reader
      Format:
         ------------------------------------------------------------
         [L-name] ULD
         ------------------------------------------------------------
      Operation:
         Simulate rewinding the tape on the Reel Reader, and unloading
         it from the tape drive. After the unload operation is
         complete, the RR file (Reel Reader) is closed, and end-of-job
         operations are performed by SIM390.
         The D-address (Next Instruction) is the same as the
         address of the current instruction.
         Operands are not allowed, but embedded comments (/*COMMENT..)
         are permitted.
         Note: On the NCR390, an Unload instruction (Rewind with the
               NI pointing to itself) resulted in the unloading of
               the Reel Reader tape (as with SIM390), but the
               computer remained hung in a loop until the Reset key
               was depressed or the machine was powered off. SIM390
               prevents this loop by forcing an EOJ condition, and
               returning to an IDLE state, available for more work.
      Level Equality: None
      Machine Instruction: 05 4R -- -- -- NI

      Examples: ULD1   ULD
                ULD2   ULD   /* THIS WILL UNLOAD THE RR TAPE AND
                *               GO TO EOJ.



WRK - Work Cell
      Format:
         ------------------------------------------------------------
         [L-name] WRK
         ------------------------------------------------------------
      Operation:
```

```
         Generates all zeros in this cell.
         Trailing comments and embedded comments are permitted.
      Level Equality: N/A
      Op-Code: None.

      Examples: WORK1  WRK      WORK AREA 1
                WORK2  WRK      /*WORK AREA 2
                WORK3  WRK



WTL - Write Ledger with Line Find (not simulated)
      Format:
         ------------------------------------------------------------
         [L-name] WTL [FROM] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Name1 must be a higher address than name2.
         This operation is treated as a HALTING NO-OP by the
         SIM390 program. SIM390 will display the current address
         and instruction, and stop, waiting for the Enter key to
         be depressed.
      Compilation: The WTL instruction is checked for proper syntax
         and compiled according to the NCR390 specifications.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 07 1S AA BB -- NI

      Examples: WTL1   WTL FROM ABC THRU XYZ
                WTL2   WTL FROM ABC THRU XYZ GOTO ADDUP



WTN - Write Ledger with No Line Find (not simulated)
      Format:
         ------------------------------------------------------------
         [L-name] WTN [FROM] name1 [THRU name2] [GOTO G-name]
         ------------------------------------------------------------
      Operation:
         Name1 must be a higher address than name2.
         This operation is treated as a HALTING NO-OP by the
         SIM390 program. SIM390 will display the current address
         and instruction, and stop, waiting for the Enter key to
         be depressed.
      Compilation: The WTN instruction is checked for proper syntax
         and compiled according to the NCR390 specifications.
      Level Equality: Name1 and name2 (A-address and B-address)
      Machine Instruction: 07 0S AA BB -- NI

      Examples: WTN1   WTN FROM ABC THRU XYZ
                WTN2   WTN FROM ABC THRU XYZ GOTO ADDUP



--- - "Renames" Statement (3 dashes in Symbolic position, col 8)
      Format:
         ------------------------------------------------------------
         L-name --- name1
         ------------------------------------------------------------
      Operation:
         Renames the statement at the preceding address.
         Ie, allows the program to refer to the original name
         and the alternate name as if they were the same.
         L-name is required.
         Multiple Renames statements can be used for the same
         address.
      Level Equality: N/A
      Machine Instruction: None

      Program Example:
```

```
         ADDR
          001   WORK1  WRK      /*THE WORK AREA
          001   WORK2  ---      /*WORK2 RENAMES WORK1
          001   WORK3  ---      /*WORK3 RENAMES WORK1
          002   EQUX   EQU *    /*NEXT ADDRESS
          002   ADDUP  ADD WORK1 AND WORK2 GIVING WORK3

                (All names in ADDUP refer to the same location)



UNUSED - Unused cell
      Format (column 1):
         ------------------------------------------------------------
         UNUSED
         ------------------------------------------------------------
      Operation:
         Generates 1 cell of zeros.
         Any Symbolic is ignored.
      Level Equality: N/A
      Machine Instruction: None

      Program Example:
                WORK1  WRK         /*THE WORK AREA
                UNUSED  /*UNUSED FOR NOW - MAY USE LATER
                WORK3  WRK         /*WORK AREA 3

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

---------- Language Quick Reference ----------

```
 Some NCR390 Symbolic operation codes have been replaced by
 different ones in PL390. The unused NCR390 codes are shown
 in parentheses and their PL390 replacement codes are listed.
 For example, BLA has been replaced by the ADD operation code.
 No functionality has been removed.

 Example:
    ADD  - Add          (valid)
   (BLA) - Block Add    (invalid - replaced by ADD)

 ------------------------------------------------------------
 [blank line] - Blank lines are permitted and are treated as
                comment lines.
 *    - Comment statement (* in column 1)
 /*   - Embedded comment in PL390 statement
 ACL  - Accept Ledger
 ADD  - Add  (single or block, for ADD/BLA)
 ADR  - Address
(BLA)- Block Add   (see ADD)
(BLC)- Block Copy  (see CPY)
(CAC)- Carriage Control  (see CAR and FRM)
 CAN  - Cancel Job (SIM390 Extended Instruction)
 CAR  - Carriage Control
(CFE)- Compare For Equality  (See IFF)
(CFM)- Compare For Magnitude (See IFF)
(CFZ)- Compare For Zero      (See IFF)
 CLR  - Clear Cells
 CPY  - Copy  (single or block, for CPY/BLC)
 DAT  - Data
 DIV  - Divide
 EOJ  - End of Job (SIM390 Extended Instruction)
 EQU  - Equate
 FRM  - Advance Continuous Forms
 HLT  - Halt CPU Processing
 IFF  - Compare For Zero (CFZ)
        Compare For Equality (CFE)
        Compare For Magnitude (CFM)
 INS  - Insert digits
 LOK  - Halt, go to this address if STEP
 MDD  - Multiply Dollar Decimal
 MUL  - Multiply and Shift
 MUS  - Multiply and Shift  (see also MUL which is a synonymn)
 NOP  - No Operation
 ORG  - Set Location Counter to the specified address
 PER  - Punch ER Symbol
 PRN  - Print Number as Signed (as positive or negative)
 PRT  - Print Number as Absolute Value  (see also PRN, PTB, PTN, PER)
 PTB  - "PRN" and Tab to a #1 Block
 PTN  - "PRN" and Return Carriage to #1 Insert
 RCA  - Read Card Alpha, type via typewriter
 RCN  - Read Card Numerics (into memory)
 RDC  - Read Console
(RDL)- Read Ledger  (see ACL, REJ, RLR, RMG, RMK)
 REJ  - Read and Eject Ledger
 REP  - Replicate cells
 RLR  - Read fron Ledger Reader
 RLS  - Release Punched Card
 RMG  - Read Ledger with Magnetic Line Find
 RMK  - Read Ledger with Mechanical Line Find
(RPC)- Read Punched Card  (see RCA, RCN, RLS)
```

```
 RPT - Read Paper Tape from the Reel Reader  (see also RPS)
 RWD - Rewind Paper Tape on the Reel Reader
(RWP)- Rewind Paper Tape on the Reel Reader  (see RWD)
 SHF - Shift and Copy
 SPS - Space Platen
 SUB - Subtract
 SUM - Summarize
 ULD - Unload Paper Tape from the Reel Reader
 WRK - Work Cell
 WTL - Write Ledger with Line Find  (see also WTN)
 WTN - Write Ledger with No Line Find
 --- - "Renames" Statement (3 dashes in Symbolic position, col 8)
 UNUSED - (starting in column 1) Unused cell


 PROGNAME - Names the program (up to 8 characters)
 PUNCH    - Punch data into the Object Deck


------------ Symbolics with Optional Modifiers-------

 Optional modifiers are placed in column 11, immediately
 after the Symbolic.

       Mod
       ---
        M = Manual operation
        A = Automatic operation (for RDC only)

 CAR  - Carriage Control, Automatic
 CARM - Carriage Control, Manual

 FRM  - Forms Advance, Automatic
 FRMM - Forms Advance, Manual

 PER  - Punch "ER", Automatic
 PERM - Punch "ER", Manual

 PRN,  PRT,  PTB,  PTN  - Print, Automatic
 PRNM, PRTM, PTBM, PTNM - Print, Manual

 RDC  - Read Console, Manual
 RDCA - Read Console, Automatic


-------------------------- Summary -----------------------------

 Common line formats:

  CLF1 --- No operands, no "GOTO"
  ------------------------------
  WRK  LOK  ULD


  CLF2 --- No operands, but includes a "GOTO"
  -----------------------------------------
  HLT  RWD  SPS  RCA  RLS  FRM  PER  ACL  NOP


  CLF3 --- Level equality required for operand names 1 and 2
  ----------------------------------------------------------
  RDC  PRT  PTB  PTN  PRN
  RCN  RMG  RMK  RLR  REJ  WTL  WTN
  RPT  RPS  SUM  CLR


 Special-use names
 -----------------
```

zeros to be generated. It is equivalent to the Symbolic "WRK".

Non-procedural instructions

EQU ORG WRK DAT ADR --- (3 DASHES)

Compiler directives

PROGNAME
PUNCH

Extended Symbolic Operations

The following instructions exist on the NCR390 computer,
but the PL390 Symbolic is different from the NCR390 version.
(Only Symbolics which generate an instruction are listed).

ACL (RDL) - Accept Ledger (position ledger)
CAR (CAC) - Carriage Control
FRM (CAC) - Advance Continuous Forms
IFF (CFZ) - Compare for Zero
IFF (CFE) - Compare for Equality
IFF (CFM) - Compare for Magnitude
LOK (HLT) - Halt, goto this address if STEP
MUL (MUS) - Multiply and Shift (same as "MUS")
NOP (CFE) - No Operation
PER (PRT) - Punch ER Symbol
PRN (PRT) - Print Number as Signed (as positive or negative)
PTB (PRT) - "PRN" and Tab to a #1 Block
PTN (PRT) - "PRN" and Return Carriage to #1 Insert
RCA (RPC) - Read Card Alpha
RCN (RPC) - Read Card Numerics
REJ (RDL) - Read and Eject Ledger
REP (BLC) - Replicate
RLR (RDL) - Read fron Ledger Reader
RLS (RPC) - Release Punched Card
RMG (RDL) - Read Ledger with Magnetic Line Find
RMK (RDL) - Read Ledger with Mechanical Line Find
RPS (RPT) - Read Paper Tape from the Strip Reader
RWD (RWP) - Rewind Paper Tape on the Reel Reader
SPS (RDC) - Space Platen
ULD (RWP) - Unload Paper Tape from the Reel Reader
WTN (WTL) - Write Ledger with No Line Find

Extended Instructions

The following instructions do not exist on the NCR390 computer.
However, the PL390 language and the SIM390 simulator support
these extensions to the operation of the computer.

EOJ - End of Job (Op code 20)
CAN - Cancel Job (Op code 21)

(Non-Procedural Instructions)
WRK - A work cell
DAT - Data
ADR - Address
EQU - Equate
ORG - Set address counter
--- - "Renames" statement

Instructions Not Simulated

The following instructions are not simulated by SIM390, but
they will be compiled correctly by the PL390C compiler:

1. Read Magnetic Ledger:
   ACL - Accept Ledger (no-op on SIM390)
   REJ - Read and Eject Ledger
   RLR - Read from Ledger Reader
   RMG - Read Ledger with Magnetic Line Find
   RMK - Read Ledger with Mechanical Line Find

2. Write Magnetic Ledger:
   WTL - Write Ledger with Line Find (see also WTN)
   WTN - Write Ledger with No Line Find

Opcode Cross Reference

NCR390 To PL390

NCR390 Operation code to
PL390 Language Symbolics Cross Reference

NCR390 Opcode	PL390 Symbolics
00	RDC
01	PRT, PRN, PER, PTB, PTN
02	CAR, FRM
03	HLT, LOK
04	RCA, RCN, RLS
05-1	RPT
05-3	RPS
05-4	RWD, ULD
06	RMG, RMK, RLR, REJ, ACL
07	WTL, WTN
08	SUM
09	CLR
10	MDD
11	MUL/MUS, SQR
12	INS
13	SHF
14	SUB
15-0	IFF EQ/NE (CFE), NOP
15-1	IFF ZERO (CFZ)
16	IFF GE/LT (CFM)
17-0	ADD
17-1	CPY
18-0	ADD A THRU B (BLA)
18-1	CPY A THRU B (BLC), REP
19	DIV
--- Extensions ---	
20	EOJ
21	CAN

Hardware Specifications and Prices - 1960

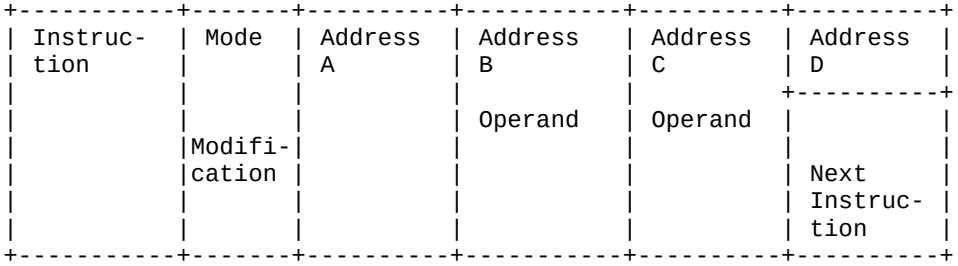
BRL 1961, NATIONAL 390, start page 0714

NATIONAL 390
National Cash Register Company Model 390 Computer

MANUFACTURER
National Cash Register

APPLICATIONS
System is designed to handle all types of accounting records, reports,
and statistics,
paper tape sorting (Direct and Sequential), engineering calculations,
and linear programming problems (Limited to 10 x 15 matrix or less)

PROGRAMMING AND NUMERICAL SYSTEM
Internal number system Binary Coded Decimal Decimal
Digits/word 12
Decimal digits/instruction 12
Instructions per word 1
Instructions decoded 27
Arithmetic system Fixed point
Instruction type Four address
Number range From - 1 x 10-9 to + 9 x 10-9
Instruction word format



Two decimal digits each
Automatic built-in subroutines include block transfer,
and sum and add pairs of numbers.
Variable block instructions perform some functions similar to B-boxes.

ARITHMETIC UNIT
Operation Including Storage Access Time, In Microseconds:
   Add 11,000 (11 milliseconds)
   Mult 250,000 (250 milliseconds)
   Div 400,000 (400 milliseconds)
Above times are "worst case".
Because of the 4 address system, command times all include access
and storage.
Arithmetic unit is constructed of 48 cores, with transformers and diodes.
Arithmetic mode Serial Timing Synchronous Operation Sequential

STORAGE

No. of Words	No. of Digits	Media Access Core
200	2400	22 microsec/bit

Magnetic Ledger

200 digits	200	220 char/sec

Variable word length on magnetic cards

BRL 1961, NATIONAL 390, start page 0715

INPUT
Media Speed
Paper Tape (Photoelectric) 400 char/sec
Punched Card (IBM 024 or 026) 18 char/sec
Magnetic Ledger Card 220 char/sec
Speed of reading and writing depends on card length.
The average is 1.5 to 2.0 secs.
Console Keyboard (Standard)
The Magnetic Ledger Card is a standard ledger card with standard visible posted information on the front and strips of magnetic tape on the back capable of storing up to 200 digits of information pertaining to that account.

OUTPUT
Media Speed
Paper Tape 17 char/sec
Punched Card 18 char/sec
Magnetic Ledger Cards Same as input
Accounting Machine Printer 1200 char/min
The Accounting Machine type printer is completely programmable both horizontally and vertically.
It will accommodate continuous forms, journals, cut forms, and ledger cards all simultaneously, if desired and has all accounting machine checking, comparing, and accumulating features.

CIRCUIT ELEMENTS OF ENTIRE SYSTEM
Type Quantity
Diodes 4,000
Transistors 1,150
Magnetic Cores 9,792
14 vacuum glow triodes are used as indicators.

CHECKING FEATURES
Among the fixed checking features are a 5 bit parity check, reader and punch check, power supply tolerances auto check, a print-out check, and ledger card read-write failure indicators are used. Test points are available on all logical circuits.

POWER, SPACE, WEIGHT, AND SITE.
PREPARATION
KVA,
computer 4.8 KVA 1 phase 240v Area,
computer 78 sq ft Room size 10 ft x 15 ft Floor
loading 20 lbs/sq ft
40 lbs concen
max Weight, computer 1,000 lbs 1,500 lbs, total

PRODUCTION RECORD
Number produced to date 6
Number in current operation 6
Number on order 100+
Anticipated production rates 600 - 700 annually
Quantity production will commence in the first quarter of 1961.

COST, PRICE AND RENTAL RATES

	Basic System Price	Monthly Rental
390-3 Console and Central Processor	$56,300	$1,395
361-1 Paper Tape Reader	10,000	250
461-2 Tape Recorder	1,735	50
Additional Equipment		
381-1 Punch Card Reader Coupler	2,250	60
468-1 Punch Card Coupler	815	27
417 Paper Tape Rewinder-Splicer	1,215	30
361-2 Paper Tape Reader	?	?

Maintenance service is included in the rental price,
and is approximately 5 to 6% of purchase price annually.

PERSONNEL REQUIREMENTS

A typical installation will require a combination supervisor and programmer,
an operator, and possibly one clerk.
The number of input operators would depend on the volume and type of input media
and the method of creating it, e.g. by-product of necessary parent machine operation,
off-line separate operation, etc.

RELIABILITY, OPERATING EXPERIENCE, AND TIME AVAILABILITY

Acceptance test specifies 40 hours continuous operation without failure or error.
Tests are run under extreme marginal conditions.

ADDITIONAL FEATURES AND REMARKS

Outstanding features include magnetic ledger cards, accounting machine printer,
4 address system, internally stored program, decimal coding, and desk size.
The unique Magnetic Ledger Card which combines visible, auditable, historical information
posted on the front, with machine language encoded on the back.
Up to 200 characters of information pertaining to each account can be stored on
the back of each card.
The magnetic ledger philosophy provides unlimited external storage facility
and immediate random access to a complete, up-to-date historical record.

FUTURE PLANS

Future plans include alphanumerics, a document sorter (MICR) input,
optical document and journal readers, automatic ledger handling,
increased speed and capacity, and a high speed printer.

Historical Notes - Later Models

A later model of the NCR390 called the NCR395 was also
developed by National Cash Register. I don't have any
specs for that machine, but suspect it was also a "numeric"
machine, with alpha capability limited to typing information
manually or automatically from punched cards, but not storing
it in memory or on tape. The one photo I've obtained of the
395 shows that the console was similar to the 390's.
It was a bit slimmer and sleaker, and had different buttons,
but was similar in appearance.

Another successor was the NCR399 introduced in 1974. It included
2 cassette tape drives, and a programming language called NEAT/AM.
AM meant Accounting machine. It could take 30 minutes to compile
a program due to all the tape switching for the compiler, source
program, linker, and object code. The appearance of the console
still had some similarities to the NCR390, but the keyboard was
a single surface, with a 10-key pad replacing the many-key numeric
buttons of the 390. Magnetic ledgers were still used.

The NCR500 computer (1965) included internal alpha capability,
but I never used one and have few specs for it.
It used punched paper tape, punched cards, and mag cards,
and was programmed in machine language, and apparently,
assembler language. Memory consisted of 400 words, and used
a 4-address scheme for programming (like the NCR390).

Two other machines which were follow-ons to the 399 and 500 were
the 499 and the 590 (1965).

Thus, for the NCR390 series and its relatives, we have (I think):

NCR Computer Model	Year-Introduced	
390	**1960**	**(Mass production: May 1961)**
395	**19??**	
500	**1965**	
590	**1965**	
399	**1974**	**(2 later models in 1976)**
499	**1975**	**approx**

NCR Computer List

Comprehensive Computer Catalogue

Manufacturer	Machine Name	Origin	Date	Reference
NCR	102 aka CRC 102	USA	Jan-52	NCb p234
NCRss				
NCR	102A	USA	1953	BW 46
NCR	107 aka CRC 107	USA	Apr-53	NCa p190
NCRss				
NCR	120D	USA	1955	BW 89
NCRss				
NCR	303	USA	Jan-56	NCb p235
NCRss				
NCR	304	USA	Nov-59	IPM p334
NCRss				
NCR	310	USA	Apr-61	PAL p334
NCR	390	USA	May-61	BW 221
NCRss				
NCR	315	USA	Jan-62	EoCS p544
NCR	315 CRAM	USA	Jan-62	BW 242
NCRss				
NCR	315-100	USA	Nov-64	BW 329
NCRss				
NCR	315/RMC-501	USA	Jul-65	BW 347
NCR	500	USA	Sep-65	PAL p337
NCR	590	USA	Sep-65	BW 353
NCRss				
NCR	315/RMC-502	USA	May-67	PAL p338
NCRss				
NCR	Century 100	USA	Sep-68	BW 452
NCRss				
NCR	Century 200	USA	Jun-69	BW 494
NCRss				
NCR	Century 101	USA	1972	EoCS p1129
NCRss				
NCR	Century 251	USA	1973	EoCS p1129
NCRss				
NCR	8550	USA	1976	EoCS p1129
NCR	Criterion 8550	USA	1976	IC Aug76
NCR	Criterion 8570	USA	1976	IC Aug76
NCR Comten	3650	USA	1976	CR90
NCRss				
NCR	7200 Model VI	USA	1977	IC Jan77
NCR	8250	USA	1977	IC Apr77
NCR	8560	USA	1977	EoCS p1129
NCR	N-8450	USA	1977	IC Jul77
NCR	V-8450	USa	1977	IC Jul77
NCR Comten	3690	USA	1977	CR90
NCRss				
NCR	V-8585-II	USA	1982	CR90
NCR	V-8595-II	USA	1982	CR90
NCRss				
NCR	Decision Mate V	USA	1983	BG84 p429
NCR	V-8635	USA	1983	CR90
NCR	V-8645	USA	1983	CR90
NCR	V-8655	USA	1983	CR90
NCR	V-8665	USA	1983	CR90
NCR	V-8675	USA	1983	CR90
NCR	V-8685	USA	1983	CR90
NCR	9300	USA	Mar-83	ABC Oct87
NCRs Comtens				
NCR Comten	5620	USA	1985	CR90
NCR	9400	USA	Mar-85	ABC Oct87
NCR	9500	USA	Oct-85	ABC Oct87
NCRss				

NCR	7000CP	USA	1986	CR90
NCR	9800	USA	1986	CR90
NCR Comten	3695	USA	1986	CR90
NCR Comten	5660	USA	1986	CR90
NCR Comten	721-II	USA	1986	CR90
NCR	9300IP	USA	Jan-86	ABC Oct87
NCR	9400IP	USA	Jan-86	ABC Oct87
NCRss				
NCR	V-8835	USA	1987	CR90
NCR	V-8845	USA	1987	CR90
NCR	V-8855	USA	1987	CR90
NCR	V-8865	USA	1987	CR90
NCR	V-8875	USA	1987	CR90
NCR	V-8885	USA	1987	CR90
NCR	V-8895	USA	1987	CR90
NCR	Tower 32/800	USA	Feb-87	ABC Oct87
NCRss				
NCR	10000/35	USA	1988	CR90
NCR	10000/55	USA	1988	CR90
NCR	10000/65	USA	1988	CR90
NCRss				
NCR	I 8130	USA	Oct-97	01 p52
NCR	I 8150	USA	Oct-97	01 p52
NCRss				
NCR	102D	USA		ARL
NCR	299	USA		CBI 55
NCR	370	USA		TCT
NCR	395	USA		GBD p507
NCR	399	USA		01 p52
NCR	499	USA		CBI 55
NCR	8000	USA		CBI 55
NCR	8100	USA		CBI 55
NCR	8200	USA		CBI 55
NCR	8300	USA		CBI 55
NCR	8400	USA		CBI 55
NCR	8500	USA		CBI 55
NCR	8600	USA		CBI 55
NCR	Century 300	USA		
NCR	Century 50	USA		DM Jun72
NCR	Century 8200	USA		IC Jan76
NCR	I-9000	USA		CBI 55
NCR	MiniTOWER	USA		ABC Oct87
NCR	TOWER 32/400	USA		ABC Oct87
NCR	TOWER XP	USA		ABC Oct87
NCR	TOWER/600	USA		ABC Oct87
NCR	Tower 32/S	USA		BvH
NCR	V-8500	USA		CBI 55
NCR	V-8600	USA		CBI 55
NCR	V-8800	USA		CBI 55
NCR	Worldmark 5100M	USA		

--- Acknowledgements ---

The designer and builder of the NCR390 computer and its
associated documentation is the National Cash Register Company,
Dayton, Ohio.

Our NCR Customer Engineer on Guam, Ichiro Harada of Tokyo,
taught me much about the NCR390 and its programming.

Glen Thompson let me borrow his NCR Gold Manual to study,
which was crucial to understanding the computer.

Thanks also to Vern Cozad and Dave Opal at FE Warren AFB,
Wyoming, who assisted me when I began using the NCR390.

Information from various websites has also been included in
this document.

Dave Morton created the following programs and documents:
1. SIM390 - the NCR390 simulator.
2. PL390 - the NCR390 symbolic programming language.
3. PL390C - the compiler for PL390 programs.
4. Tictactoe for the NCR390 in NCR390 machine language.
5. Tictactoe for the NCR390 in the PL390 language.
6. All test programs (PLX..., etc).
7. All documentation for this software and the PL390 language.

David E. Morton
09-02-2002