Advanced Software Development Tools

For the HP-41

Version 2.0

PROGRAMMER’S MANUAL
Advanced Software Development Tools

For the HP-41

Version 2.0

PROGRAMMER’S MANUAL

Copyright (c) 1989 Warren Furlow
SOFTWARE LICENSE AGREEMENT

WARNING: ACCEPTANCE OF THIS SOFTWARE CONSTITUTES A FORMAL AGREEMENT SUBJECT TO THE FOLLOWING TERMS:

INDIVIDUAL LICENSING
This software and documentation shall in no way be modified, or distributed except where designated "Public Domain" and except that copies may be made for backup purposes. At no time should it be possible for this software to be executed by more than one user. Any software contained within that is designated "Public Domain" may be freely distributed and copied except that if it is modified, it should be clearly noted so.

SITE LICENSING
Inquire to address below.

LIABILITIES
The purchaser assumes complete liability for any damages arising from the use or misuse of this product.

WARRANTY
This software will be replaced free of charge if the medium is found to be defective when returned within a period of 60 days from purchase with proof of purchase. No warranty is made for the fitness of the software itself.

SEND REGISTRATION CARDS AND QUESTIONS TO:
Warren Furlow
5595 Coronation Court
Atlanta, Georgia 30338
(404) 396-1862
ACKNOWLEDGEMENTS

The ASDT project started about a year and a half ago with an early Pascal HP-41 MCODE cross assembler. After switching to 'C', I proceeded to write the other utilities. I also discovered that writing documentation can be harder than writing programs. I would like to thank those who have provided me with assistance, advice and sources of information over the course of the development of ASDT: Brian Walsh, Jeff Brown, Eleanor Furlow, Mike Katz and Ross Cooling. I would also like to thank in advance anyone that has any suggestions or comments that can improve this software.

REFERENCES

The following sources and organizations are very useful for the HP-41 enthusiast:

HP VASM Listings
These are the annotated listings of the operating system of the HP-41 and are available from HPX, send SASE.

Zenrom Programmer's Manual
This book is sold with or without the ZENROM module by EduCalc and is a very good reference to HP-41 MCODE.

HP-41 MCODE for Beginners
This book by Ken Emery is a good tutorial for HP-41 MCODE and is available from EduCalc or Synthetix publishers (who also publish a number of other useful books on the HP-41)

Synthetix
P.O. Box 1080
Berkeley, CA 94701-1080.

HPX
HPX is a user's group publishing a periodical journal providing the HP hand-held user with a forum for many areas of common interest, analysis of new products and interesting programs. Yearly dues are $25 in the continental U.S. and $35 elsewhere.

HPX
Brian Walsh
P.O. Box 4160,
Des Plaines Ill, 60016
(312) 884-0099

EduCalc
EduCalc carries the most complete line of HP hardware, software, accessories and books at competitive prices.

EduCalc Mail Store
27953 Cabot Road
Laguna Niguel, CA 92677
CONTENTS

1. Introduction .............................................................................................. 1
2. Features ..................................................................................................... 2
3. Definitions .................................................................................................. 3

4. Utilities
   A41 Assembler .......................................................................................... 4
   L41 Linker .................................................................................................. 8
   D41 Disassembler ...................................................................................... 10
   M41 Emulator ............................................................................................ 12
   T41 Translator ............................................................................................ 17
   41COM Communications Utility .............................................................. 18

5. Linking and Disassembly ........................................................................... 19

6. File Formats
   Source File Format .................................................................................... 22
   Backup File Format .................................................................................... 24
   ROM File Format ....................................................................................... 25
   Object File Format ..................................................................................... 26
   Link File Format ....................................................................................... 27
   Configuration File Format ......................................................................... 28
   Load File Format ....................................................................................... 29
   Label File Format ...................................................................................... 30
   Register File Format .................................................................................. 31

7. Reference Instructions
   Short Jumps .............................................................................................. 32
   Long Jumps ............................................................................................... 33
   Other Reference "Instructions" ................................................................... 34

Appendix
   A. Common ROM IDs ................................................................................. 35
   B. HP-41 ROM Memory Map ..................................................................... 36
   C. Format For ROM with FAT ................................................................. 37
   D. Character Translation Table ............................................................... 38
   E. Keycode Table ....................................................................................... 39
   F. RAM Configuration Data ...................................................................... 40
   G. Instruction Set Cross Reference Table ................................................. 41
SECTION 1. INTRODUCTION

The Advanced Software Development Tools (ASDT) package contains all the software needed to develop custom HP-41 ROM images on MS-DOS machines. This ability makes the development of MCODE programs faster and easier than ever for the noncommercial programmer and provides professional quality tools for commercial development. The following basic setup is recommended:

BASIC SETUP
HP-41CV/CX

IBM PC compatible computer; hard disk recommended

MLDL (Machine Language Development Lab) with at least 8K

HP82160 HP-IL module for HP-41

Either HP8297A HP-IL interface card (for IBM compatible) or HP-IL <=> RS232 converter
SECTION 2. FEATURES

ASSEMBLER
- Assembles MCODE instructions from any one of the three instruction sets: HP, Jencode or Jacobs/De Arras
- Contains built-in symbol table of the HP-41's mainframe labels which allows easy referencing to operating system entry points
- Allows generation of internal and external symbolic labels
- Assembles FAT pseudo-instructions for easy ROM image building
- Produces symbol cross reference table, if desired

LINKER
- Resolves external label references and creates a ROM image file suitable for loading into an EPROM or an MLDL
- Can link more than one ROM image at a time
- Produces symbol cross reference table, if desired

DISASSEMBLER
- Disassembles MCODE instructions into any one of the three instruction sets
- Contains built-in symbol table to the HP-41's mainframe labels
- Disassembles FAT pseudo-instructions for easy ROM image decoding
- The source file generated by the disassembler can be immediately re-assembled by the assembler

EMULATOR
- Reads ROM images and allows single stepping and debugging of the instructions which may be in any one of the three instruction sets
- Executes the HP-41 operating system (the HP-41 operating system ROMs are not included and must be loaded by the user with the communications utility)
- Provides full screen display of the HP-41's internal registers, display and RAM and allows the user to set and clear breakpoints, jump to any address and preload registers
- Incorporates mainframe labels into pages 0-2 and can read and incorporate labels from user-specified label files

INSTRUCTION SET TRANSLATOR
- Translates the instructions in source files from any of the three sets to any other set

COMMUNICATIONS
- Upload and download ROM image files from an HP-41 through HP-IL to a PC (requires basic setup)
SECTION 3. DEFINITIONS

Note: The mnemonics in this manual use the Zencode instruction set.

<address>
Any value in the range 0000-FFFF (hex)

<value>
Any value in the range 0000-FFFF (hex). This is the same as an <address> but is used in a more generalized sense.

$page>
Any value in the range 0 to F (hex). A page is also 4096 words.

<bank>
Any value in the range 1 to 2 (dec). The HP-41 supports bankswitching with special ROM devices.

<+disp>
Any value in the range +0 to +63 (dec).

<-disp>
Any value in the range -1 to -64 (dec).

<local label>
A label that is delimited with parenthesis. Example: (FOOBAR). Local labels are local to the file that they are defined in and are not "visible" outside that file. Any characters except spaces may be contained in the label definition but for convention labels should be limited to uppercase characters and the underscore. If a global label is the same as a local, there will be no conflict, but this should not be done to avoid confusion. Local machine code labels should not be confused with local User Code labels as they are completely different.

<global label>
A label that is delimited with brackets. Example: [FOOBAR]. Global labels are "visible" to all files that are linked together. Any characters except spaces may be contained in the label definition but for convention labels should be limited to uppercase characters and the underscore. Global machine code labels should not be confused with global User Code labels as they are completely different.

<label>
Either a <local label> or a <global label>. This is similar to a <symbol> but is used in the more restrictive sense that labels are not defined with the .EQU directive.

<symbol>
Same as a <label> but used in a more generalized sense to include all symbols including those defined with the .EQU directive.

<operand>
The argument that is given after the instruction. Example: XS is the operand for A=B XS.
SECTION 4. UTILITIES

A41 ASSEMBLER

SYNTAX
A41 [options] <file>

DESCRIPTION
A41 assembles HP-41 CODE mnemonics from one of three instruction sets and produces an object file that is linkable into a ROM image. A41 expects to find <file.src> that contains the code to assemble and produces <file.obj>. A41 also has the ability to reformat the source file and insert error messages and object data into it.

FILES
<file.src> source file (read from and written to)
<file.bak> backup file (written to)
<file.obj> object file (written to)
<file.lbl> label file (written to)

OPTIONS
/S If this option is not specified, the source file is read and not modified. Otherwise, the source file is read and formatted and various useful information such as error messages, object data and cross references are optionally inserted into it. A backup file is created that contains the original source code.
/R Does the same as /S and also produces a complete symbol cross reference table at the end of the source file.
/O This option causes the generation of the <address> and <data> fields into the source file when the /S or /R option is specified. These fields are for the user's benefit and are ignored by the assembler upon subsequent assemblies. This option does nothing if /S or /R is not specified.
/L Generates <file.lbl> that contains all local labels in the source file, but no global labels. See Section 6 for label file format.
/E Erases internal mainframe label table. The internal mainframe label table is a data table in the assembler that holds all of the HP-41's operating system entry points as specified by the VASM listings. This option supports custom HP-41 operating systems.

ASSEMBLER DIRECTIVES
.TITLE "title"
Gives a title to the object code. If this title is longer than 80 characters it will be truncated.

.HP
Specifies the HP instruction set and must be given before the code starts. One source file may contain several instruction set directives and they may be different.

.ZENCODE
Specifies the Zencode instruction set and must be given before the code starts.
.JDA
Specifies the Jacobs/De Arras set and must be given before the code starts.

.FILLTO <address>
Fills from current address to the specified address with zeros. If the .ORG directive is specified, this directive will fill from the current address to the specified address PLUS the origin address. The addresses are considered occupied and are not open for linking any other object code into.

.BSS <number>
Fills the next <number> words with zeros, where <number> is a positive decimal number. The addresses are considered occupied and are not open for linking any other object code into.

.NAME "name"
Macro for defining function names. This is used for construction of the FAT. It converts each character to its LCD equivalent and adds 80 hex to the last character. The order of the characters is automatically reversed as required by the HP-41.

.MESSL "string"
Similar to the NAME directive, the MESSL directive puts the string into the format required for output via the [MESSL] entry point. Each character is converted to its LCD equivalent and the 20 hex is added to the last character.

.ORG <address>
Specifies an absolute address in the range 0000-FFFF (hex) to originate the code at. All symbols defined in a file that contains this directive are absolute and cannot be relocated by the linker. This directive can be specified only once in each source file.

.EQU <global symbol> <value>
.EQU <local symbol> <value>
Equates a symbol with a value in the range 0000-FFFF (hex). This symbol functions just like any label, but is NOT relocatable since it represents an absolute constant.

#$000-#3FF
This is not a directive but allows any literal to be entered directly into ROM. This is similar to the CON pseudo-instruction except that only literal values in the range 000-3FF (hex) can be entered and not symbols.

ERROR MESSAGES

*** FATAL ERROR (A01): Code runs past FFF (hex)
The assembler cannot assemble more than FFF (hex) words into one object module since that is the maximum length of a ROM image.

*** FATAL ERROR (A02): Source File Is Empty! Check backup file
The assembler found the source file to be empty which could have been caused by interruption of the assembler when it was running previously. The backup file will contain the original source file.

*** FATAL ERROR (A03): Failure to rename <file x> to <file y>
This error will result if for some reason the operating system prevents the assembler from renaming <file x> to <file y>. Check the file access on the files.
**FATAL ERROR (A04):** Out of memory!
This error occurs when the system's dynamic memory has been all used up. If this occurs, shorten source file.

**ERROR (A05):** Jump address not in same 4K page
The address specified for the quad relative jump is not in the same page as the instruction itself. The instruction is assembled anyway.

**ERROR (A06):** Illegal label definition: <label>
The label is greater than 13 characters or is not delimited by brackets or parenthesis.

**ERROR (A07):** Operand not recognized: <operand>
The operand is not valid for the instruction given. The assembler will still generate code.

**ERROR (A08):** Illegal .EQU definition
The symbol specified is not a legal symbol since it is greater than 13 characters or is not delimited by brackets or parenthesis or the value is not in the range 0000-FFFF (hex).

**ERROR (A09):** FAT names cannot be greater than 11 Characters
The HP-41 does not support FAT names greater than 11 characters long. No data will be generated.

**ERROR (A10):** Illegal address: <address>
The specified address is not in the range 0000-FFFF (hex).

**ERROR (A11):** Illegal number: <number>
The specified number is not a valid positive number.

**ERROR (A12):** .ORG must be specified before code starts
The .ORG directive was specified after the code started and was ignored.

**ERROR (A13):** Illegal .ORG definition
The address specified for the .ORG directive is not in the range 0000-FFFF (hex).

**ERROR (A14):** .ORG specified more than once
The .ORG directive cannot be specified more than once in each source file.

**ERROR (A15):** Unknown directive: <directive>
The directive is not valid.

**ERROR (A16):** Reference to external local not permitted: <local symbol>
A reference to a local symbol (delimited by parenthesis) was made and that symbol is not in the current source file.

**ERROR (A17):** Illegal instruction: "INSTRUCTION"
The instruction given is not in the current instruction set.

**ERROR (A18):** Missing Operand
An operand is required.
*** ERROR (A19): Illegal characters in NAME string
There were characters found in the FAT function name that are not allowed by the HP-41. The assembler supports all possible characters for NAME strings.

*** ERROR (A20): Illegal characters in MESSL string
There were characters found in the MESSL string that are not mapped to the HP-41. There are characters that can be displayed on the HP-41 that the assembler does not support. These are any characters with punctuation bits set or the extra characters that only the halfnut LCD can display. The MESSL directive cannot be used to encode these; they must be entered manually as literal data using the # token.

*** ERROR (A21): Illegal displacement: <displacement> (dec)
A displacement was specified that was out of the range -64 to +63 (dec).

*** ERROR (A22): Literal Address in relocatable object module
This occurs when a source file does not have the .ORG directive in it and short jump instructions have operands that are literal address. (Such as JNC).

*** WARNING (A50): Operand greater than FFF (hex)
The LC3 macro instruction expected an operand in the range 000-FFF (hex). The operand was truncated to 12 bits.

*** WARNING (A51): Operand greater than 3FF (hex)
The CON pseudo-instruction expected an operand in the range 000-3FF (hex). The operand was truncated to 10 bits.

*** WARNING (A52): Duplicate symbol: <symbol> Address not used <address>
A symbol was defined more than once in the same source file. Only the first occurrence is used, and the others are ignored. The address of all later occurrences are listed in the warning message.
L41 LINKER

SYNTAX
L41 [options] <file>

DESCRIPTION
The linker links the object files together to create one or more ROM files that contain one ROM image each. The commands that direct the linker are contained in <file.lnk>. See Section 6 for more information on the link file. L41 can link SDS format .410 files. (Any use of SDS object files requires the linking of the SDS file MFENTRY.410 since the SDS assembler does not resolve mainframe entry points)

FILES
<file.lnk> link file (read from)
<?????obj> object file(s) (read from)
<file.cfg> config file (written to)
<file.lbl> label file (written to)
Where ???? is a file name specified in the link file
<file.rom> ROM file (written to)
<file#.rom> ROM file(s) (written to)
Where # is a decimal number from 0 to the maximum number of ROM images that are linked M INUS one.

OPTIONS
/L Creates label file containing all global labels in all object modules.
/R Creates symbol cross reference table for the global labels and places it in the config file.
/A Assembles object modules that are out of date. This option causes A41 to be called if source file is newer than object file. Any letters that follow the /A are passed to A41 as options. Example: /ARO calls A41 with the /R and /O options. A41 is also called if /AL is specified and the source file is newer than the LABEL file.

ERROR MESSAGES
*** FATAL ERROR (L02): Object code runs past end of page
Attempted link for code that runs past the end of the current ROM image.

*** FATAL ERROR (L03): <object file> is corrupt
The object file is corrupt and cannot be read by the linker because it contains unexpected data.

*** FATAL ERROR (L04): Out of memory!
This error occurs when the system's dynamic memory has been all used up.

*** FATAL ERROR (L05): Cannot create the same page twice
The link file has more then one page command with the same parameters.

*** FATAL ERROR (L06): Illegal Parameter in $PAGE command
The parameters specified for the page command are not valid.

*** FATAL ERROR (L07): $PAGE Command not given before first object file name
A page command must be specified before the first object file name is given.
*** ERROR (L08): Jump address not in same 4K page
The address specified for the quad relative jump was not in the same page as the instruction itself.

*** ERROR (L09): Illegal displacement: <displacement> (dec)
The short jump instruction is referenced to a symbol that is out of its range of -64 to +63 (dec).

*** ERROR (L10): Symbol not defined: <symbol>
A reference to a non-existent symbol was made. The address used to link is 0000.

*** ERROR (L11): <object file> written to space occupied by <object file>
If the current object file maps to the space that another object file has already been linked to this error results and the new object file will be loaded over the old object file.

*** ERROR (L12): <number> Error(s) in assembly of <source file>
The assembler returned errors from the assembly of a source file. This message appears just before the linker terminates so if there are errors in the assembly they will be more obvious.

*** WARNING (L50): Reference from bank <bank x> to bank <bank y> at address <address>
This warning results when a reference from one bank to another is made. The reference is resolved as if the banks were the same which means unpredictable results are possible at run time.

*** WARNING (L51): Object file <object> originates at <origin page> but has been forced into page <current page>
This results when the address specified for the ORG directive when the file is assembled is not in the same page as the current page. The linker forces the object file into the current page and continues linking.

*** WARNING (L52): Duplicate symbol: <symbol> Address not used <address> from object file <object file>
A symbol was defined more than once. Only the first occurrence is used, and the others are ignored. The address and object file of all later occurrences is listed in the warning message.

*** WARNING (L53): $LOC value <value> has been forced into page <page>
The page specified for the $LOC command is not the same as the current page. The linker forces the <value> into the current page.
D41 DISASSEMBLER

SYNTAX
D41 [options] <file>

DESCRIPTION
The D41 disassembler is capable of disassembling ROM image files into one of the three instruction sets. It expects to find <file.rom> and it produces <file.src>. If the ROM file contains User Code, the User Code instructions are represented as literal data in the range #000 to #3FF.

FILES
<file.rom> ROM file (read from)
<file.lbl> label file (read from)
<file.src> source file (written to)

OPTIONS
/H Disassembles into the HP set (default). This option may not be given in conjunction with /Z or /J.
/Z Disassembles into the Zencode set. This option may not be given in conjunction with /H or /J.
/J Disassembles into Jacobs/De Arras set. This option may not be given in conjunction with /H or /Z.
/Pn Maps the ROM into page n, where n is 0 to F (hex). This is used for producing a listing for a ROM that is hard-configured such as one of the operating system pages. See Section 5 for more information on the use of this option. This option does not actually change any code. If this option is not specified, the default is page 8.
/F Causes the ROM image to be disassembled with a FAT. If this is not specified, the ROM is assumed to have no FAT.
/E Erases the internal mainframe label table so that the disassembly does not place the HP-41 operating system entry labels into the source file. This option is only useful if the ROM image maps into pages 0-2 but is not part of the HP-41's operating system. This option supports nonstandard operating systems.
/L:<label file>
Specifies a label file containing labels which are incorporated into the source listing just as the mainframe labels are. See Section 6 for label file format. This option may be used multiple times to specify all label files desired.

DEFAULTS
HP instruction set
Page 8
Does not disassemble with a FAT
Mainframe labels active
No external label files

ERROR MESSAGES
*** FATAL ERROR (D04): Out of memory!
This error occurs when the system's dynamic memory has been all used up.
*** ERROR (D05): Illegal label definition: <label> in: <label file>
The label is greater than 13 characters or is not delimited by brackets or parenthesis.

*** ERROR (D06): Illegal character found in NAME string
The NAME string contains a character that is not allowed by the HP-41. The disassembler supports all possible characters for NAME strings. A tilde (~) character is displayed instead.

*** ERROR (D07): Illegal character found in MESSL string
The MESSL string contains a character that is not supported by the disassembler. The disassembler will produce an error for all characters that have punctuation bits set or the extra characters that only the halfnut LCD can display. A tilde character is displayed instead.

*** ERROR (D08): Illegal $LOC command in label file: <label file>
The address specified for the LOC command is not in the range 0000-FFFF (hex).

*** ERROR (D09): FAT entry not recognized
The data in the FAT entry differs from the standard HP-41 FAT pseudo-instructions.

*** ERROR (D10): FAT not followed by two NOP instructions
The HP-41 requires that the FAT be followed by two NOP instructions.

*** ERROR (D11): Data at ROM address 1 (number of FAT entries) is incorrect
The data in address 1 of the ROM differs from the actual number of FAT entries disassembled.
M41 EMULATOR

SYNTAX
M41 [options] [ <file> ]

DESCRIPTION
M41 is a powerful and useful tool for testing and developing custom MCODE programs. It allows single stepping of MCODE programs and also supports a continuous run mode that executes the HP-41 operating system ROMs when they have been loaded with 41COM. (The ROMs are not provided with ASDT since they are owned by HP.)

OPERATION
After loading the operating system ROMs, the 'U' command may be executed and the emulator will mimic an HP-41. There are several things to know when the emulator is in this mode. If a ?KEY instruction is encountered, the emulator will check the PC's keyboard to see if a key is being pressed. If one is not, it will go on without interrupting the execution. If a key is pressed, it will be translated and loaded into the KEY register. Also, the registers and instructions will not be updated on the screen but the display will be. If the 'R' command is executed, the emulator will also run at full speed, but will stop for trap conditions and breakpoints.

The screen is divided into three areas; the instructions, the registers and the display. The pointer at the right hand side of the instructions indicates which instruction will be executed next. After every instruction is executed, it is rewritten to the display. This sometimes causes the instruction to change. An example of this is when a REG=C 3 instruction changes to a WRAB6L instruction. This occurs because no peripheral was selected when the entire screen was disassembled, but a PERSLCT FD was executed after this and that changed the active peripheral.

The display and register parts of the screen are mostly self-explanatory with the following notes. BK stands for BANK and is either 1 or 2. KEY is the KEY register which contains the keycode, while ?KEY is the keydown register and is either 1 or 0. PERRH is the selected peripheral. HEX is 1 if the CPU is in hex mode and 0 if in decimal mode. There are three registers that have different names depending on the active instruction set. SB is also ST, XSB is XST and F is T (This is shown in Appendix G). The block of RAM registers at the bottom left of the screen is the active chip and RAM is the RAM address selected. If the chip is non-existent, dashes are displayed. If an Extended Functions ROM is loaded into page 3 the emulator automatically makes all extended memory available. CODE is the code of the instruction just executed.

FILES
<????.rom> ROM file(s) (read from)
Where ????? is a file name specified in the load file
<file.lod> load file (read from)
If <file> is not given on the command line, DEFAULT.LOD is used instead
<file.lbl> label file (read from)
<file.reg> register file (read from and written to)

OPTIONS
/H Displays instructions using the HP set (default). This option may not be given in conjunction with /Z or /J
/Z Displays instructions using the Zenode set. This option may not be given in conjunction with /H or /J.
/J Displays instructions using Jacobs/De Arras set. This option may not be given in conjunction with /H or /Z.

/E Erases the internal mainframe label table so that the disassembly does not place the HP-41 operating system entry labels into the source file. This option is only useful for ROM images that map into pages 0-2 but are not part of the HP-41’s operating system. This option supports nonstandard operating systems.

/EGA43 /EGA /CGA /MONO

If one of the above video options is given, the emulator will be forced into the specified mode. Otherwise, the emulator will set itself to the highest mode available.

COMMANDS

A Display alpha register.

BC <address>
Clear the breakpoint at <address> if there is one.

BC
Clear the breakpoint at the current address if there is one.

BC <label>
Clear the breakpoint at <label> if there is one.

BS <address>
Set breakpoint at <address>.

BS
Set breakpoint at the current address.

BS <label>
Set a breakpoint at <label> if it exists.

BL List all breakpoints.

C <chip>
Display the specified RAM chip where <chip> is 000 to 3FF (hex) with exceptions as noted in Appendix F.

D DOS shell (Type EXIT to return).

E <code>
Executes a SINGLE byte instruction (Do not execute a multi-byte instruction).

FS <reg file>
Saves all registers to a register file.

FL <reg file>
Loads all register from a register file.
G <address>
Goto the address specified.

G <label>
Goto the label specified, if it exists.

H Change to the HP instruction set.

J Change to the Jacobs/De Arras instruction set.

K <key>
Enter a key to the KEY register and set the keydown register. This maps the PC keyboard to the HP-41 keyboard, just like for the 'U' run mode.

L <register> <value>
Load <register> with <value>, where <register> is the name of the CPU register for the current instruction set as described below:

<table>
<thead>
<tr>
<th>HP</th>
<th>Zenode</th>
<th>Jacobs/De Arras</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>14 hex digits</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>14 hex digits</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>14 hex digits</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td>M</td>
<td>14 hex digits</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>14 hex digits</td>
</tr>
</tbody>
</table>

LR <RAM register> <value>
Load <RAM register> with <value> where a RAM register may be any existing register from 000 to 3FF (hex) and the value is 14 hex digits long.

M Display this menu of commands.

QY Exit from the emulator.

R Run at full speed until a breakpoint, keyboard or POWOFF trap occurs.

U Execute the operating system ROMs (that you loaded) and check the keyboard when a ?KEY instruction is detected (see KEYBOARD MAPPING below). The ESC key will cause a break out of run mode, but it may take several seconds for it to be processed.

Z Change to the Zenode instruction set.

SPACE BAR
Single step the emulator.

KEYBOARD MAPPING
To simulate the HP-41 in USERCODE mode (execute the 'U' command) the PC's keyboard has been mapped to the following specifications. This is not a key for keycode mapping in all cases. For instance, the letters A-Z may only be entered if the emulator sees that ALPHA mode is set. Also, certain keys on the HP-41 keyboard are shifted, but may be entered directly on the PC keyboard WITHOUT first entering a SHIFT on the emulator because the emulator will set shift mode first. An example of this is the % function. The shift key on the PC keyboard does not map to the shift key on the HP-41.
<table>
<thead>
<tr>
<th>PC Key</th>
<th>HP-41 meaning</th>
<th>MCODE keycode value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>ON</td>
<td>18 (hex)</td>
</tr>
<tr>
<td>F2</td>
<td>USER</td>
<td>C6</td>
</tr>
<tr>
<td>F3</td>
<td>PRGM</td>
<td>C5</td>
</tr>
<tr>
<td>F4</td>
<td>ALPHA</td>
<td>C5</td>
</tr>
<tr>
<td>F5</td>
<td>SHIFT</td>
<td>C4</td>
</tr>
<tr>
<td>F6</td>
<td>SST</td>
<td>C2</td>
</tr>
<tr>
<td>F7</td>
<td>&lt;- (BACKARROW)</td>
<td>C3</td>
</tr>
<tr>
<td>F8</td>
<td>R/S</td>
<td>87</td>
</tr>
<tr>
<td>F9</td>
<td>FUNCTION (described below)</td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>KEYCODE (described below)</td>
<td></td>
</tr>
<tr>
<td>SHIFT F1</td>
<td>XEQ</td>
<td>32</td>
</tr>
<tr>
<td>SHIFT F2</td>
<td>ENTER^</td>
<td>13</td>
</tr>
<tr>
<td>SHIFT F3</td>
<td>CHS</td>
<td>73</td>
</tr>
<tr>
<td>SHIFT F4</td>
<td>EEX</td>
<td>83</td>
</tr>
</tbody>
</table>

If the emulator is in ALPHAmode:

A to Z
A to Z

a b c d e a b c d e 10, 30, 70, 80, C0
s Sigma 11
n Not equal 71
x Append 32
g Angle symbol 73
\( < > > \) \( < > > \) 31, 81, C1, 83
- + * / - + * / 14, 15, 16, 17
Space Space, 37, 77, 77

If the emulator is not in ALPHAmode:

. . 77
0 to 9 0 to 9

FUNCTIONS

All HP-41 keys that are marked with something other than a single digit symbol may be entered with the FUNCTION key (F9). The emulator will prompt for the name of the command to enter. The commands are not case sensitive. The table below lists valid commands for ALPHAmode:

<table>
<thead>
<tr>
<th>ON USER PRGM ALPHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND ASTO ARCL BST SST AVIEW R/S</td>
</tr>
</tbody>
</table>

The following are valid commands when not in ALPHAmode:

<table>
<thead>
<tr>
<th>ON USER PRGM ALPHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>s- s+ 1/X Y^X SORT X^2 LOG 10^X LN E^X</td>
</tr>
<tr>
<td>CLS X&lt;Y EDN ½ SIN ASIN COS ACOS TAN ATAN</td>
</tr>
<tr>
<td>SHIFT XEQ ASM STOLBL RCL CTO SST BST</td>
</tr>
<tr>
<td>ENTER CAT CATALOG CHS ISS EEX RTW CLX</td>
</tr>
<tr>
<td>X=Y? SF CF FS? X&lt;Y? BEEP P-R -P</td>
</tr>
<tr>
<td>X&gt;Y? FIX SCI ENG X=0? PI LASTX VIEW R/S</td>
</tr>
</tbody>
</table>
KEYCODES

It is also possible to enter hex keycodes directly by using the F10 key. This will enter any hex value from 00 to FF into the KEY register. If in 'U' mode, F10 toggles keycode entry. See Appendix E for the keycode table.

LIMITATIONS AND WARNINGS

When the Zenrom is loaded, it operates exactly as it should under 'U' mode, but there is one known condition when it will lock up. If the emulator is in ALPHAMode and a SHIFT is entered and then any key other than ALPHA is entered, the Zenrom will get stuck in a loop.

Some of the TEF instructions have undefined behaviors if the emulator is in decimal mode and there is a value greater than 9 in the register being added or subtracted.

ERROR MESSAGES

*** FATAL ERROR (M04): Out of memory! - Too many ROM files
This error occurs when the system's dynamic memory has been all used up.

*** FATAL ERROR (M05): Cannot load the same page twice
The load file has more than one page command with the same parameters.

*** FATAL ERROR (M06): Illegal Parameter in $PAGE command
The parameters specified for the page command are not valid.

*** FATAL ERROR (M07): ROM file name not specified in $PAGE command
The $PAGE command must have a ROM file name specified after it to load.

*** FATAL ERROR (M08): No ROM images loaded
There were no ROM images loaded.

*** FATAL ERROR (M09): Label file is empty: <label file>
The label file specified in the load file was empty.

*** ERROR (M20): Illegal $LOC command in: <label file>
The address specified for the LOC command is not in the range 0000-FFFF (hex).

*** ERROR (M21): Illegal label definition: <label> in: <label file>
The label is greater than 13 characters or is not delimited by brackets or parenthesis.

*** ERROR (M30): Unknown result
A peripheral I/O instruction was executed that has an unknown result for the current peripheral selected.

*** ERROR (M31): Unknown Peripheral
The peripheral select code is undocumented.

*** ERROR (M32): Timer not implemented
The timer I/O instructions are not supported.

*** ERROR (M33): Card reader not implemented
The card reader I/O instructions are not supported.

*** ERROR (M34): Printer not implemented
T41 INSTRUCTION SET TRANSLATOR

SYNTAX
T41 <file>

DESCRIPTION
The instruction set translator translates the mnemonics in a source file from one instruction set to any of the three. (It is possible to translate from Zencode to Zencode, for instance.) The translator reads the source file until it finds an instruction set directive which specifies the current instruction set. Then, it prompts for the new set with self-explanatory messages. The old source file becomes the backup file. Any errors will be flagged and stored in the new source file. If there is more than one instruction set directive in the file, T41 will prompt for a new instruction set each time it finds one.

FILES
<file.src> source file (read from and written to)
<file.bak> backup file (written to)

OPTIONS
None

ERROR MESSAGES
*** FATAL ERROR (T02): Source File Is Empty! Check backup file
The translator found the source file to be empty. The backup file should contain the original source file.

*** FATAL ERROR (T03): Failure to rename <filex> to <filey>
This error will result if for some reason the system prevents the translator from renaming <filex> to <filey>. Check the file access on the files.

*** FATAL ERROR (T04): Out of memory!
This error occurs when the system's dynamic memory has been all used up.

*** ERROR (T05): Instruction not given
An instruction was expected on this line and none was found.

*** ERROR (T06): Illegal label definition: <label>
The label is greater than 13 characters or is not delimited by brackets or parenthesis.

*** ERROR (T07): Unknown directive: <directive>
The specified directive is not valid.

*** ERROR (T08): Illegal instruction: "<instruction>"
The instruction given is not in the current instruction set.
41COM COMMUNICATIONS UTILITY

SYNTAX
41COM <file>

DESCRIPTION
This utility transmits and receives 4K ROM image files to and from the HP-41 over the HP-IL loop. It requires two programs on the HP-41 called "ROMIN" and "ROMOUT". For information on loading these programs, see the file "BOOT.SRC". On-line help is available by typing H after running 41COM. This utility requires an HP82973A HP-IL interface card for the PC and an HP82160 HP-IL module for the HP-41.

It is also possible to use an HP-IL module in the HP-41 with an HP-IL <=> RS232 convertor and use the RS232 serial port of the PC to send and receive ROM files. The 41COM utility is not needed in this case since the PC can simply send and receive the ROM file over its serial port using any serial upload/download program that supports 8-bit ASCII. The software on the HP-41 end is the same either way.

FILES
<file.rom> ROM file (read from or written to)

OPTIONS
None

ERROR MESSAGES
*** ERROR: EOT received before all data sent
The HP-41 sent an EOT signal before it sent all 8192 bytes.

*** ERROR: Time out
The HP-41 prematurely terminated its transmission and the PC timed out or the HP-41 failed to respond in time to data sent by the PC. The timeout factor is approximately two seconds.
SECTION 5. LINKING AND DISASSEMBLY

BACKGROUND INFORMATION
Because the smallest unit of ROM memory in HP-41 is one page, the linker builds pages. Each HP-41 page is 4096 words long. Each word is 10 bits long. The HP-41 can address a total of 16 pages which are numbered page 0 through page F (hex). Some of these pages are already hard-wired to the HP-41 operating system (See Appendix B). Other pages are left open for the user's plug-in modules and these map to the four I/O ports. Each port has two 4K pages associated with it so a plug-in module can use either one or two pages. The actual hardware of the module determines which page(s) it uses. Most 4K modules use the lower page (although some use the upper) while all 8K modules use both. These configurations are known as port-configured since the actual pages that the module maps into are dependant on which port it is plugged into. Most of the commercially available ROM equipment is of this variety.

12K and 16K modules must be bank-switched as described later in this section.

It is possible to have a module that does not occupy the pages associated with the port it is plugged into. The ROM image(s) in this type of module are hard-configured to always map into the same page(s) regardless of which port it is plugged into. This is the case for several HP-41 peripherals or special modules such as the Mass Storage ROM. For instance, regardless of which port the Mass Storage ROM is plugged into, it will always occupy page 7. (The Mass Storage ROM is inside the HP-IL module.) Since most accessory hardware for the HP-41 on the market today does not support hard-configured ROM images, this is not an alternative to most MCODE programmers. One way to get around this is to pretend that a piece of hardware is hard-configured and always plug it into the same I/O port. This distinction is important since some jump instructions are not position independent.

The linker can link many possible configurations including port- and hard-configurations and bank switching.

LINKING PORT-CONFIGURED ROMS
Most types of user-created ROMs will be of this type. The best way to link these is to start with a SPAGE 8 1 command for the lower ROM. If there is an upper ROM, it would be linked with a command of SPAGE 9 1. If the upper page is bank switched, the hidden page would be linked with a command of SPAGE 9 2. The only reason the linker needs to know a page number is to resolve references that are relative to each other. It would work just as well to specify $PAGE 5 for the lower ROM and $PAGE 6 for the upper. An example of this type of link is the Advantage ROM. It is a 12K bankswitched port-configured module. It could be linked with $PAGE 8 1, $PAGE 9 1 and $PAGE 9 2. If the link file were named ADV.LNK, The ROMs would be named ADVO.ROM, ADV1.ROM and ADV2.ROM, respectively.

LINKING HARD-CONFIGURED ROMS
To link a ROM of this type, the $PAGE command would be specified with the page that the ROM is to be located at. If there is more than one ROM to be linked, simply specify exactly which page and bank each one is to be located at. The hardware will insure that each ROM is mapped to its proper location. For example, the operating system of the HP-41 is 12K of hard-configured non bank switched ROMs. It could be linked with $PAGE 0, $PAGE 1, $PAGE 2. If the link file were named NUT.LNK, the ROMs would be named NUTO.ROM, NUT1.ROM and NUT2.ROM, respectively.

DISASSEMBLY OF PORT-CONFIGURED ROMS
To disassemble a port-configured ROM, it is best to start at page 8 for the first ROM image and if there is a second, disassemble to page 9. For example, to disassemble an 8K port-configured ROM, do not specify the /Pn option (the default is 8) for the first page and specify /P9 for the second.
DISASSEMBLY OF HARD-CONFIGURED ROMS

To disassemble a hard-configured ROM, the page that the ROM was taken from must be specified with the /Pn option. For example, to disassemble the operating system of the HP-41, specify /P0 for the first page and /P1 and /P2 for the other two. If the operating system is disassembled into pages 0-2, the mainframe labels will appear properly at the locations they are defined at.
SECTION 6. FILE TYPES
SOURCE FILE FORMAT

FILE EXTENSION
.SRC

All source files must have a .SRC extension.

FILE TYPE
Normal MS-DOS text file

LINE FORMATS
 директив
Directive lines begin with a period and are followed by the directive which must be in all caps.

; Comments ...
Comment lines begin with a semicolon and may contain anything after the semicolon.

* Error
* Macro

Error and macro lines begin with an asterisk and are removed by the assembler when they are found in the source file. This type of line may contain anything after the asterisk. They are placed in the source file by the utilities to denote errors in code or to denote the code generated by macro expansions.

Instruction lines
Any line that is not one of the above is an instruction line and must follow the syntax diagram below:

Instruction lines have optional fields that may or may not affect assembly. The address and data fields are ignored by the assembler but are rewritten if the source file is rewritten. The user does not place the address and data fields in the source file. The address field is 4 hexadecimal digits long and the data field is 3, 6 or 9 hexadecimal digits long. The label field is next and is where a local or global label is defined. Labels without instructions on the same line will cause an error. Depending on which instruction is used, there may be from zero to three operand fields after the instruction but the assembler uses only the first and the rest are ignored and placed for the user’s benefit. Any characters after that must be proceeded by a semicolon and are treated as a comment.

USED BY
A41 The assembler reads the old source file and writes a listing to the new source file if the /S or /R option is given; otherwise it does not modify the source file.

D41 The disassembler writes its disassembly listing to a source file. If there is a source file with the same name as the one the disassembler is about to disassemble into, the original one is made into a backup file.

T41 The instruction set translator reads the old instructions from the source file and writes the translated instructions to a new source file of the same name.
FILE SAFETY
If the assembler or translator are interrupted, it is possible to lose the source file, but it is not possible to lose both the backup and source files.

COMPATIBILITY WITH SDS
There are significant differences in SDS .41A source files and ASDT .SRC files.

EXAMPLE
See any of the .SRC files on the ASDT disk.
BACKUP FILE FORMAT

FILE EXTENSION
.BAK

All backup files must have a .BAK extension.

FILE TYPE
Normal MS-DOS text file

PURPOSE
Backup files are maintained automatically by the utilities as a measure of file safety. It is not possible to interrupt one of the utilities and lose both the source and backup files.

LINE FORMATS
Same as for source files.

USED BY
A41  The assembler will make the old source file into the backup file if /S or /R is given. If a backup file already exists, it is overwritten. If /S is not given the backup file will not be modified.

D41  When the disassembler first executes, it checks for a source file with the same name as the one it is about to disassemble into. If it finds one that already exists, it makes it into a backup file. If a backup file by that same name also exists, it is overwritten. If it does not find a source file it does not do anything to the backup file.

T41  The old source file becomes the backup file every time that the translator is executed. Any preexisting backup file of the same name is overwritten.
ROM FILE FORMAT

FILE EXTENSION
  .ROM
  All ROM files must have a .ROM extension.

FILE TYPE
  Binary data file

DATA FORMAT
  HIGH 2, LOW 8
  Each 10-bit HP-41 word is stored in two 8-bit bytes in a ROM file. The high 2 bits of the 10-bit word are stored in the low 2 bits of the first byte and the low 8 bits are stored in the second byte. Since there are 4096 10-bit bytes in each ROM image, all ROM files must be exactly 8192 bytes long.

USED BY
  L41 The linker writes the final, linked code to one or more ROM files.
  D41 The disassembler reads the code from the ROM file and disassembles it.
  M41 The emulator can read and execute the ROM image files.
  41COM
  The HP-41 communications utility reads or writes ROM files over the HP-IL loop.

COMPATIBILITY WITH SDS
  ROM files are in the same format as .41R files.
OBJECT FILE FORMAT

FILE EXTENSION

.OBJ

All object files must have a .OBJ extension.

FILE TYPE

Binary data file

USED BY

A41 The assembler writes internal global label definitions (entry points), unresolved external references, relocation fixups, and the object code into each object file along with some other information used by the linker.

L41 The linker reads the object files, locates the code in the ROM image, resolves external references and fixes up all references.

COMPATIBILITY WITH SDS

ASDT .OBJ files may be renamed .410 and used with the SDS LINK41 utility. SDS .410 files may be renamed .OBJ and used with the ASDT L41 utility but the SDS file MFENTRY.410 must be renamed MFENTRY.OBJ and linked in also.
LINK FILE FORMAT

FILE EXTENSION
.LNK
All link files must have a .LNK extension.

FILE TYPE
Normal MS-DOS text file

PURPOSE
Link files are used to direct the linker in the creation of the ROM files.

LINE FORMATS
<object file name>
This is the file name of the object file to load given without the .OBJ extension. The object data is loaded at the current load address in the current ROM image. The load address is then modified so the next object file will load immediately following the previous.

$PAGE <page>
$PAGE <page> <bank>
$PAGE <page> <ROM name>
$PAGE <page> <bank> <ROM name>
Where <page> is 0 to F (hex) and <bank> is 1 or 2. If <bank> is not given, the default is 1. This command opens a new ROM image at the specified page and bank. All object files specified after this are linked into this ROM image until another $PAGE command is given or the link file ends. When there is only one ROM image it will be named the name of the link file (with a .ROM extension). If there is more than one ROM image, a number from 0 to the number of ROM images minus one is appended to this name. If <ROM name> is specified, that name will be used to name the ROM image instead.

$LOC <address>
Where <address> is a value 0000 to FFFF (hex). This command changes the load address so the object files following it are loaded consecutively starting at this address. If <address> is not in the same page as the page given with the current $PAGE command, a warning message will result and the linker will force <address> into the current page. If this command is not given, the initial load address is p000 where p is the current page.

$CH
Compute checksum and place in location FFF (hex) for the current ROM image. This command must be given for each page where a checksum is desired. If location FFF is occupied by any object code, an error message is generated and the checksum is written anyway.

;comment
Any line that begins with a semicolon is ignored.

USED BY
L41 Link files are used to direct the linking of object files.

EXAMPLE
See PCCOM.LNK on the ASDT disk for an example of a link file.
CONFIGURATION FILE FORMAT

FILE EXTENSION
.CFG
  All config files must have a .CFG extension.

FILE TYPE
Normal MS-DOS text file

PURPOSE
  This type of file is only written by the linker and is used to show which ROM is mapped to which page as specified by the link file. The config file may optionally have the symbol cross reference table written to it.

LINE FORMATS
$PAGE <page> <bank> <rom file>
  Where <page> is 0 to F (hex) and <bank> is 1 or 2. This documents the mapping of <rom file> to the specified page and bank.

; anything
* anything
blank line
  These three line formats contain information that is for the user’s benefit only. The * lines contain the symbol cross reference table if there is one.

USED BY
L41  Config files are written by the linker. The linker will write the ROM image mapping and optionally a symbol cross reference table.

EXAMPLE
  After executing the demo on the ASDT disk, a configuration file with the name POCOM.CFG will be created.
LOAD FILE FORMAT

FILE EXTENSION
.LoD
All load files must have a .LoD extension.

FILE TYPE
Normal MS-DOS text file

PURPOSE
This file is very similar to the config file format. It is only used by the emulator to load the ROM images and label files.

LINE FORMATS
$PAGE <page> <bank> <rom file>
where <page> is 0 to F (hex) and <bank> is 1 or 2. The ROM file will be loaded into the specified page and bank.

; anything
* anything
blank line
These three line formats contain information that is for the user's benefit only and is not read by the emulator.

$LABELS <label file>
The labels in the <label file> will be read and incorporated into the disassembly listing of the emulator.

USED BY
M41 Load files are read by the emulator and specify the ROM images to read in. The file DEFAULT.LOD is special since the emulator will read it if it exists and a load file name is not specified on the command line.

EXAMPLE
A load file with the name DEFAULT.LOD is included on the ASDT disk although it specifies fictitious ROM images.
LABEL FILE FORMAT

FILE EXTENSION
.LBL

All label files must have a .LBL extension.

FILE TYPE
Normal MS-DOS text file

LINE FORMATS
<label> <address>
Where <label> is any global or local label and <address> is a value 0000 to FFFF (hex). Labels do not have to be in any particular order. If the label file has more than about fifty labels in it, they should NOT be placed in alphabetical or reverse alphabetical order.

$LOC <address>
Where <address> is a value 0000 to FFFF (hex). This address is simply added to each of the addresses of the labels affected. The labels affected are those that appear after each $LOC command and before the next $LOC command or the end of file. This command may be given more than once in the same file and if not given at all the default is 0000.

; comment
Any line that begins with a semicolon is ignored.

USED BY
A41 When the /L option is given, the assembler writes all local labels to a label file that has the same name as the source file. It does not write global labels.

L41 When the /L option is given, the linker writes all global labels read from all object files to one label file that has the same name as the link file. It does not write local labels since they do not appear in the object files.

D41 For each /L:<label file> option specified on the command line of the disassembler, all local and global labels are read from the specified file and incorporated into the source listing just as for the mainframe labels. The addresses specified by each label are used exactly as they appear unless they are offset by the $LOC command.

M41 The emulator reads the label files just like the disassembler and incorporates the labels into its listings.

EXAMPLE
[FOOBAR_A] 0023
(FOOBAR_D) 32C2
$LOC B000
[FOOBAR_B] 04A3
$LOC 0200
(FOOBAR_C) 032B
;this is a comment

[FOOBAR_A] is interpreted as 0023, (FOOBAR_D) is 32C2, [FOOBAR_B] is B4A3, (FOOBAR_C) is 052B.
REGISTER FILE FORMAT

FILE EXTENSION
.REG
All REG files must have a .REG extension.

FILE TYPE
Binary data file

USED BY
M41 The emulator can read and write the CPU and RAM registers into a register file.
SECTION 7. REFERENCE INSTRUCTION FORMATS

SHORT JUMPS

INSTRUCTIONS

JNC <operand> Jump on no carry, otherwise do nothing
JC <operand> Jump on carry, otherwise do nothing

DESCRIPTION
The short jump instruction jumps -64 to +63 words from where the instruction is. This type of instruction encodes only a relative displacement in its bits.

FORMATS

JNC <+disp>
JNC <-disp>

Where <+disp> is +0 to +63 and <-disp> is -1 to -64 (dec). The actual address that is jumped to is relative to where the short jump instruction is located after it is assembled and linked and the ROM is plugged into the HP-41. Regardless of where the instruction ends up, it always jumps the same number of words forward or backward.

JNC <address>

Where <address> is 0000 to FFFF (hex). The assembler resolves this type by taking the specified address and subtracting the jump instruction's ASSEMBLE TIME ADDRESS to get a positive or negative displacement. This should only be used if the object module is assembled with the .ORG directive so that the assemble time address will be the same as the run time address. This is a literal specification and the linker does not relocate anything. It is possible to specify an address that is too far from the jump, causing A41 to report an error.

JNC <label>

Where <label> is any local or global label. The address that will be jumped to is whatever the value of the label is, assuming it is within range. If the label is external, the linker must resolve the reference, otherwise it is completely resolved during assembly.
LONG JUMPS

INSTRUCTIONS

NCXQ <operand>  Goto on no carry, otherwise do nothing
CGO <operand>  Goto on carry, otherwise do nothing
NCXQ <operand>  Execute on no carry, otherwise do nothing
CXQ <operand>  Execute on carry, otherwise do nothing

DESCRIPTION

These instructions occupy two words each and can jump to anywhere in the HP-41 address space. The address that they jump to is encoded in the instruction’s bits and regardless of where the instruction is located, it always jumps to the same location. The goto types of long jumps simply cause an immediate jump depending on the carry flag. The execute types do the same but also push a return address onto the HP-41’s return stack. The value that is pushed is the address of the word that follows the long jump.

FORMATS

NCXQ <address>
Where <address> is 0000 to FFFF (hex). The assembler resolves this type of reference literally and the linker does not change it.

NCXQ <label>
Where <label> is any local or global label. If the label is internal, it is resolved but will be fixed up (resolved again) at link time. If the label is a mainframe label, the assembler resolves it immediately without any relocation possible. If it is not a mainframe label or an internal, then the label is external and the linker must resolve the reference after it relocates all of the object files. If the /E option is specified, the mainframe label table is erased and all references to mainframe labels are treated as references to externals.
OTHER REFERENCE "INSTRUCTIONS"

The rest of the jump and reference instructions use a general format described below:

INSTRUCTION <value>
Where "INSTRUCTION" is one of the other reference "instructions" described below and <value> is some number usually from 0000 to FFFF (hex). This value is used exactly as it appears and is not relocated.

INSTRUCTION <symbol>
Where <symbol> is any local or global symbol. The symbol is relocated by the linker if it is a relative label but is not relocated if it is an absolute symbol.

CON
Enter constant into ROM
DEF4K  (Not useful - kept for compatibility with SDS)
DEF8K  Define 4K MCODE FAT entry
DEF8K  Define 8K MCODE FAT entry
U4KDEF Define 4K User Code FAT entry
U8KDEF Define 8K User Code FAT entry
LC3   LC three times
NCGOREL Goto on no carry relative to current quad
NCXQREL Execute on no carry relative to current quad

For the descriptions below assume [FOOBAR] resolves to the value 06D2 (hex).

CON  This is not an instruction at all but is used to directly enter a 10 bit value into the ROM image. This value could represent data, or even an instruction. CON 123 would make the object data contain the value 123 while CON [FOOBAR] will be 6D2. If the high 6 bits are not all zero a warning message will be displayed and the assembler will only use the low 10 bits.

DEF4K, DEF8K, U4KDEF, U8KDEF
These FAT entry pseudo-instructions are used to define the FAT and should only appear in the first part of the ROM image.

LC3  This is a macro instruction that expands into three LC instructions containing the low three nybbles of the address or symbol. LC3 E2A4 expands to LC 2, LC A, LC 4 and LC3 [FOOBAR] expands to LC 6, LC D, LC 2. If the high 4 bits are not all zero a warning message will be displayed and the assembler will only use the low 12 bits.

NCGOREL, NCXQREL
The quad relative goto and execute pseudo-instructions are used to jump from one address to another within a port-configured ROM image. They call special mainframe routines that do the actual jump so they are slower and use 1 or 2 return stack levels. (See Section 5 for port-configured ROM images)
APPENDIX A

COMMON ROM ID’S

1 Math
2 Statistics
3 Surveying
4 Finance, ES-41
5 Standard
6 Circuit Analysis, ES-41
7 Structures
8 Stress Analysis
9 Home Management, CCD
10 Games, Auto/Dup, PPC ROM
11 Real Estate, Eramco, CCD, Paname
12 Machine Design
13 Thermal
14 Navigation
15 Petroleum, Mountain Computer
16 Petroleum
17 Plotter, NFCROM
18 Plotter
19 Securities, Structures, Clinical Lab, Aviation
20 PPC ROM
21 Data logger, Assembler 3
22 HP-IL Development, Advantage
23 Extended I/O
24 HP-IL Development, Advantage
25 Extended Functions
26 Time
27 Wand
28 Mass Storage
29 Printer
30 Card Reader
31 Data logger
## APPENDIX B

### HP-41 ROM MEMORY MAP

<table>
<thead>
<tr>
<th>Page</th>
<th>BANK 1</th>
<th>BANK 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Port 4 upper page</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Port 4 lower page</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>Port 3 upper page</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Port 3 lower page</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Port 2 upper page</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Port 2 lower page</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Port 1 upper page</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Port 1 lower page</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HP-IL Mass Storage ROM</td>
<td>--Not used</td>
</tr>
<tr>
<td>6</td>
<td>Printer ROM</td>
<td>--Not used</td>
</tr>
<tr>
<td>5</td>
<td>Timer ROM</td>
<td>X-Functions ROM (CX only)</td>
</tr>
<tr>
<td>4</td>
<td>Reserved by HP</td>
<td>--Not used</td>
</tr>
<tr>
<td>3</td>
<td>X-Functions ROM (CX only)</td>
<td>--Not used</td>
</tr>
<tr>
<td>2</td>
<td>Operating system ROM 2</td>
<td>--Not used</td>
</tr>
<tr>
<td>1</td>
<td>Operating system ROM 1</td>
<td>--Not used</td>
</tr>
<tr>
<td>0</td>
<td>Operating system ROM 0</td>
<td>--Not used</td>
</tr>
</tbody>
</table>

---

36
APPENDIX C

FORMAT FOR ROM WITH FAT

p000 ROM ID Number
p001 Number of Functions (n)
---------------------- FAT ----------------------
p002 Address of First Function
p003 " "
.
.
p(2n) Address of Last Function
p(2n+1) " "
p(2n+2) FAT Terminator (must be loaded with 000)
p(2n+3) " "
------------------ CODE ------------------
p(n+4)
.
.
pFF3
------------------ POLLING VECTORS ------------------
pFF4 Pause Loop
pFF5 Main Running Loop
pFF6 Deep Sleep Wake Up With No Key Down
pFF7 Power Off
pFF8 I/O Service
pFF9 Deep Sleep Wake Up
pFFA Cold Start
-------------------- ROM TERMINATOR --------------------
pFFB Revision Level Characters (optional)
pFFC " "
pFFD " "
pFFE " "
pFFF Checksum (optional)

Where p is the number of the page that the ROM maps to. This format is used for ROMs at pages 3 and 5 to F.
APPENDIX D

CHARACTER TRANSLATION TABLE

The HP-41 display characters with values from 00 to 1F (hex) translate directly to the IBM/PC characters with values from 40 to 5F (hex). These are the following PC characters:

@ABCDEFGHIJKLMNOPQRSTUVWXYZ\[\ ]~_  

The HP-41 display characters with values from 20 to 3F (hex) translate directly to the IBM/PC characters with values from 20 to 3F (hex) with the exception of the { }~; characters. These are the following IBM/PC characters:

SPACE !"#$%&'()*+,-./0123456789~<=>?  

The HP-41 display characters with values from 100 to 10F (hex) translate to a number of non-intuitive IBM/PC characters. These are the following PC characters:

xabcdefghijklmnopqrstuvwxyz  

The extra HP-41 display characters supported by the halfnut display (the one with rounded edges) are not supported by ASDT.

The following table further explains the mapping of certain characters:

<table>
<thead>
<tr>
<th>HP-41 Display Characters</th>
<th>PC Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left goose</td>
<td>{</td>
</tr>
<tr>
<td>Right goose</td>
<td>)</td>
</tr>
<tr>
<td>Boxed star</td>
<td>~</td>
</tr>
<tr>
<td>Comma character</td>
<td>;</td>
</tr>
<tr>
<td>Append</td>
<td>x</td>
</tr>
<tr>
<td>Overbar</td>
<td>o</td>
</tr>
<tr>
<td>Single quote</td>
<td>`</td>
</tr>
<tr>
<td>One leg hangman</td>
<td>t</td>
</tr>
<tr>
<td>Two leg hangman</td>
<td>u</td>
</tr>
<tr>
<td>One arm hangman</td>
<td>v</td>
</tr>
<tr>
<td>Full hangman</td>
<td>w</td>
</tr>
<tr>
<td>Micro</td>
<td>m</td>
</tr>
<tr>
<td>Not equal to</td>
<td>n</td>
</tr>
<tr>
<td>Sigma</td>
<td>s</td>
</tr>
<tr>
<td>Angle symbol</td>
<td>g</td>
</tr>
</tbody>
</table>
### APPENDIX E

#### KEYCODE TABLE

<table>
<thead>
<tr>
<th></th>
<th>C6</th>
<th>C5</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>30</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td>11</td>
<td>31</td>
<td>71</td>
<td>81</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>72</td>
<td>82</td>
</tr>
<tr>
<td>13</td>
<td>73</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>34</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>35</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>36</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>37</td>
<td>77</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX F

RAM CONFIGURATION DATA

M41 emulates extended memory to support the HP-41CX. If ROM is loaded into page 3 when M41 is run, it assumes all possible RAM registers are available.

The following registers are available when emulating any HP-41:

- 000-00F
- 000-0FF

The following registers are available when the emulator detects a ROM in page 3. This is equivalent to two Extended Memory modules installed in either an HP-41 with one Extended Function module or an HP-41CX.

- 000-00F
- 000-0FF
- 040-1FF
- 201-2EF
- 301-3EF
APPENDIX G

INSTRUCTION SET CROSS REFERENCE TABLE

INTRODUCTION
This document contains all of the instructions supported by ASDT and is intended as a reference guide to programming the HP-41.

Note: When the Jacobs/De Arras set failed to contain instructions (such as the display instructions) HP mnemonics were substituted. The mnemonics NCXQREL and NCGOREL were created for ASDT.

REFERENCES
HP-41 CODE for Beginners, Ken Emory, Synthetix, 1985
The ZENROM Programmer’s Manual, Zenrange Ltd., 1984
The HP-41 VASM listings, Hewlett Packard, 1985

CPU REGISTERS

<table>
<thead>
<tr>
<th>HP</th>
<th>ZENCODE JACOBS/</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>De ARRAS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| C | C | C | 56 | Primary accumulator |
| A | A | A | 56 | Secondary storage and accumulator |
| B | B | B | 56 | Secondary storage and accumulator |
| M | M | M | 56 | Alternate storage |
| P | P | P | 4 | Nybble Pointer |
| Q | Q | Q | 4 | Nybble Pointer |
| PT | PT | R | 4 | Either P or Q, whichever is the active pointer |
| SB | ST | ST | 8 | Lower CPU flags 7-0 |
| G | G | G | 8 | Alternate storage (often flags) |
| F | P | T | 8 | Beeper register |
| STK | STK | ADR | 16 | The first address on the return stack |
| Upper Flags | | 6 | Upper CPU flags 13-8 |
| Carry Flag | | 1 | Set by some instructions; cleared after all others |
| Keydown Flag | | 1 | Set when key register has data ready |
| Key Register | | 8 | Contains the keycode entered by pressing a key |
| Return Stack | | 16x4 | Contains 4 16-bit words that hold return addresses |
| Program Counter | | 16 | Points to next instruction to execute |

56 BIT REGISTER FORMAT
Nybble 13 is the most significant; 0 is the least
Notation: A[x:y] means all nybbles in A REG from x to y; A[x] is just nybble x
GENERAL PURPOSE INSTRUCTIONS

CODE  HP  ZENCODE  JACOBS/ OPERAND  De ARRAS

FLAG INSTRUCTIONS

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>004</td>
<td>ST=0 CF</td>
<td>clear flag</td>
</tr>
<tr>
<td>008</td>
<td>ST=1 SF</td>
<td>set flag</td>
</tr>
<tr>
<td>00C</td>
<td>ST=1 SF</td>
<td>set carry if flag set</td>
</tr>
<tr>
<td>3C4</td>
<td>CLRST</td>
<td>clear lower CPU flags (7-0)</td>
</tr>
<tr>
<td>358</td>
<td>ST=C ST=C</td>
<td>copy C[1:0] into ST</td>
</tr>
<tr>
<td>398</td>
<td>C=ST C=ST</td>
<td>copy ST into C[1:0]</td>
</tr>
<tr>
<td>3D8</td>
<td>CSTEX C&lt;ST</td>
<td>exchange C[1:0] and ST</td>
</tr>
</tbody>
</table>

LOAD CONSTANTS

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>LC LC LD&amp;R</td>
<td>load constant to C[PT], then decrement pointer</td>
</tr>
<tr>
<td>110</td>
<td>LDI LDI LDS#X</td>
<td>load next word in ROM into C[2:0]</td>
</tr>
<tr>
<td>000</td>
<td>CON CON CON</td>
<td>enter hex constant into ROM</td>
</tr>
<tr>
<td>000</td>
<td>PCNS PCNS PCNS</td>
<td>decimal constant same as CON</td>
</tr>
<tr>
<td>000</td>
<td>XROM XROM XROM</td>
<td>decimal constant same as CON</td>
</tr>
</tbody>
</table>

POINTER

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>014</td>
<td>?PT=? ?PT=?</td>
<td>set carry if active pointer equal</td>
</tr>
<tr>
<td>01C</td>
<td>PT= PT= R=</td>
<td>set active pointer</td>
</tr>
<tr>
<td>0A0</td>
<td>SELP PT=P SLCTP</td>
<td>make P the active pointer</td>
</tr>
<tr>
<td>0C0</td>
<td>SELQ PT=Q SLCTQ</td>
<td>make Q the active pointer</td>
</tr>
<tr>
<td>3D4</td>
<td>DBCPT -PT R=R-1</td>
<td>decrement the active pointer</td>
</tr>
<tr>
<td>3DC</td>
<td>INCP+PT R=R+1</td>
<td>increment the active pointer</td>
</tr>
</tbody>
</table>

RAM ACCESSING

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>270</td>
<td>DADD=C RAMSLCT RAMSLCT</td>
<td>select the RAM chip addressed in C[2:0]</td>
</tr>
<tr>
<td>2F0</td>
<td>DATA=C MDATA WRITEDATA</td>
<td>writes C[13:0] to selected RAM register</td>
</tr>
<tr>
<td>038</td>
<td>C=DATA RDATA READDATA</td>
<td>writes selected RAM register to C[13:0]</td>
</tr>
<tr>
<td>028</td>
<td>REGM=C REG=C WRIT 0 to P (hex)</td>
<td>write to RAM register in selected chip</td>
</tr>
<tr>
<td>038</td>
<td>C=REGM C=REG READ 0 to P (hex)</td>
<td>read from RAM register in selected chip</td>
</tr>
</tbody>
</table>

ROM ACCESSING

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>330</td>
<td>Csimd RROM PPD#CH#X</td>
<td>copies the ROM data addressed in C[6:3] into C[2:0]</td>
</tr>
<tr>
<td>400</td>
<td>WNLDL WNLIL WROM</td>
<td>write C[2:0] to address C[6:3]</td>
</tr>
<tr>
<td>100</td>
<td>ENROM1 ENBANK1 ENROM1</td>
<td>enable ROM bank 1 for current ROM device</td>
</tr>
<tr>
<td>180</td>
<td>ENROM2 ENBANK2 ENROM2</td>
<td>enable ROM bank 2 for current ROM device</td>
</tr>
</tbody>
</table>

KEYBOARD

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>220</td>
<td>C=KEYS C=KEY C=KEY</td>
<td>copy key register to C[4:1]</td>
</tr>
<tr>
<td>230</td>
<td>GOKS#S GTOKEY GTOKEY</td>
<td>copy the key register into the low byte of the PC</td>
</tr>
<tr>
<td>3C8</td>
<td>ASTKB CLXREY CLXREY</td>
<td>clear keydown flag if no key pressed and clear key register</td>
</tr>
<tr>
<td>3CC</td>
<td>CKXKB ?KEY ?KEY</td>
<td>set carry if keydown flag is set</td>
</tr>
</tbody>
</table>
## MODE SETTING

- **060,000 POWOFF POWOFF POWOFF**; halt the CPU
- **260 SETHEX SETHEX SETHEX**; set hexadecimal mode
- **2A0 SETDEC SETDEC SETDEC**; set decimal mode
- **2E0 DISPOFF DISPOFF DISPOFF**; turn display off
- **320 DISTOG DISTOG DISTOG**; toggle state of display

### M, N, G, F Registers

<table>
<thead>
<tr>
<th>Code</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>070</td>
<td>( M = C )</td>
</tr>
<tr>
<td>080</td>
<td>( C = N )</td>
</tr>
<tr>
<td>0F0</td>
<td>( \text{CNEX} )</td>
</tr>
<tr>
<td>158</td>
<td>( M = C )</td>
</tr>
<tr>
<td>198</td>
<td>( C = M )</td>
</tr>
<tr>
<td>1D8</td>
<td>( \text{CNEX} )</td>
</tr>
<tr>
<td>058</td>
<td>( G = C )</td>
</tr>
<tr>
<td>098</td>
<td>( \text{CEX} )</td>
</tr>
<tr>
<td>258</td>
<td>( \text{P} = \text{SB} )</td>
</tr>
<tr>
<td>298</td>
<td>( \text{SB} = \text{P} )</td>
</tr>
<tr>
<td>2D8</td>
<td>( \text{FEXSB} )</td>
</tr>
</tbody>
</table>

### Other

<table>
<thead>
<tr>
<th>Code</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>(?LID )</td>
</tr>
<tr>
<td>03C</td>
<td>( \text{RCR} )</td>
</tr>
<tr>
<td>370</td>
<td>( \text{C} = \text{CORA} )</td>
</tr>
<tr>
<td>3B0</td>
<td>( \text{C} = \text{CANDA} )</td>
</tr>
<tr>
<td>1A0</td>
<td>( \text{CLRABC} )</td>
</tr>
</tbody>
</table>

- \( \text{NOP} \); no operation
- \( \text{?BAT} \); set carry if battery is low
- \( \text{?LOWBAT} \); rotate C reg right by the nybble
- \( \text{C} = \text{CORA} \); \( C[13:0] = C \) bitwise or A
- \( \text{C} = \text{CANDA} \); \( C[13:0] = C \) bitwise and A
- \( \text{ABC} = 0 \); clear all nybbles of A,B,C registers
TIME ENABLE FIELD INSTRUCTIONS

PT PT R [PT] Nybble pointed to by active pointer
X X S[X] [2:0] Sign and exponent
WPT WPT R [PT:0] Nybbles pointed to by active pointer through nybble 0
W ALL ALL [13:0] Entire register
PQ PQ P-Q [Q:P] Nybbles from pointer Q to pointer P subject to:
if P<=Q then [Q:P]; if P>Q then [13:P]
XS XS XS [2] Exponent sign
M M M [12:3] Mantissa

s m m m m m m m m m m x x x

All of the instructions in the following group work on the above
Time Enable Fields (TEF). C[TEF] is the C register with whatever field
is selected from above. Ex: A[PT], A[XS] etc.

Any of the arithmetic TEF instructions set the carry flag if either an
overflow or underflow occurs.

TEF

|   |   |   |   |
---|---|---|---|
002 | A\=0 | A\=0 | A\=0 |
022 | B\=0 | B\=0 | B\=0 |
042 | C\=0 | C\=0 | C\=0 |
062 | ABEX A\>B A\>B |
082 | B\=A B\=A B\=A |
0A2 | ACEX A\>C A\>C |
0C2 | C\=B C\=B C\=B |
0E2 | BCEX B\>C B\>C |
102 | A\=C A\=C A\=C |
122 | A\=A\+B A\=A\+B A\=A\+B |
142 | A\=A\+C A\=A\+C A\=A\+C |
162 | A\=A\+1 A\=A\+1 A\=A\+1 |
182 | A\=A\-B A\=A\-B A\=A\-B |
1A2 | A\=A\-1 A\=A\-1 A\=A\-1 |
1C2 | A\=A\-C A\=A\-C A\=A\-C |
1E2 | ACEX A\>C A\>C |
202 | C\=A\+C C\=A\+C C\=A\+C |
222 | C\=A\+1 C\=A\+1 C\=A\+1 |
242 | C\=A\-C C\=A\-C C\=A\-C |
262 | C\=C\-1 C\=C\-1 C\=C\-1 |
282 | C\=C\-C C\=C\-C C\=C\-C |
2A2 | C\=C\-1 C\=C\-1 C\=C\-1 |
2C2 | ?\=B\# ?\=B\# ?\=B\# |
2E2 | ?\=C\# ?\=C\# ?\=C\# |
302 | ?\=A\<C ?\=A\<C ?\=A\<C |
<table>
<thead>
<tr>
<th>Address</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>322</td>
<td>A&lt;B</td>
<td>;set carry if A[TEF] is less than B[TEF]</td>
</tr>
<tr>
<td>342</td>
<td>A#0</td>
<td>;set carry if A[TEF] is not equal to 0</td>
</tr>
<tr>
<td>362</td>
<td>A#C</td>
<td>;set carry if A[TEF] is not equal to C[TEF]</td>
</tr>
<tr>
<td>382</td>
<td>ASR</td>
<td>;shift A right by one nybble (leftmost byte set to 0)</td>
</tr>
<tr>
<td>3A2</td>
<td>BSR</td>
<td>;shift B right by one nybble (leftmost byte set to 0)</td>
</tr>
<tr>
<td>3C2</td>
<td>CSR</td>
<td>;shift C right by one nybble (leftmost byte set to 0)</td>
</tr>
<tr>
<td>3E2</td>
<td>ASL</td>
<td>;shift A left by one nybble (rightmost byte set to 0)</td>
</tr>
</tbody>
</table>

**TEF TWO BYTE**

<table>
<thead>
<tr>
<th>Address</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>062,082</td>
<td>A=B</td>
<td>;copy B[TEF] into A[TEF]</td>
</tr>
<tr>
<td>0E2,0C2</td>
<td>B=C</td>
<td>;copy C[TEF] into B[TEF]</td>
</tr>
<tr>
<td>0A2,102</td>
<td>C=A</td>
<td>;copy A[TEF] into C[TEF]</td>
</tr>
</tbody>
</table>
JUMPING INSTRUCTIONS

There are duplicate mnemonics for three of the HP jump instructions. GONC is the same as GOTO, GSUBNC is the same as GOSUB, and GOLNC is the same as GOLONG. HP's assemblers will check the instruction proceeding a GOTO, GOSUB or GOLONG to be sure that it cannot set the carry. This insures that these mnemonics cause an unconditional jump and not a just a jump on no carry. ASDT does not do this and assembles the duplicates exactly the same.

SHORT JUMPS

<table>
<thead>
<tr>
<th>Code</th>
<th>Mnemonic</th>
<th>Flags</th>
<th>Offset Range</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>GOC</td>
<td>JC</td>
<td>-64 to 63 (dec)</td>
<td>;short relative jump on carry</td>
</tr>
<tr>
<td>003</td>
<td>GONC</td>
<td>JNC</td>
<td>-64 to 63 (dec)</td>
<td>;short relative jump on no carry</td>
</tr>
</tbody>
</table>

LONG JUMPS

<table>
<thead>
<tr>
<th>Code</th>
<th>Mnemonic</th>
<th>Flags</th>
<th>Offset Range</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>001,000</td>
<td>GSUBNC</td>
<td>NCXQ</td>
<td>ADDRESS</td>
<td>;execute on no carry</td>
</tr>
<tr>
<td>001,001</td>
<td>GSUBC</td>
<td>CXQ</td>
<td>ADDRESS</td>
<td>;execute on carry</td>
</tr>
<tr>
<td>001,002</td>
<td>GOLNC</td>
<td>NCQ</td>
<td>ADDRESS</td>
<td>;goto on no carry</td>
</tr>
<tr>
<td>001,003</td>
<td>GOLC</td>
<td>CGO</td>
<td>ADDRESS</td>
<td>;goto on carry</td>
</tr>
</tbody>
</table>

QUAD RELATIVE JUMPS

<table>
<thead>
<tr>
<th>Code</th>
<th>Mnemonic</th>
<th>Flags</th>
<th>Offset Range</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>349,08C,000</td>
<td>GSBSAN</td>
<td>NCXQREL</td>
<td>ADDRESS</td>
<td>;execute relative to current quad</td>
</tr>
<tr>
<td>341,08C,000</td>
<td>GOLSAN</td>
<td>NCGOREL</td>
<td>ADDRESS</td>
<td>;goto relative to current quad</td>
</tr>
</tbody>
</table>

The HP instructions GSBSAM and GOLSAM are three byte jumps just like GSB41C and GOL41C, but they are limited to jumping into the current 1K quad. ASDT does not implement GSBSAM and GOLSAM since GSB41C and GOL41C are assembled intelligently by ASDT to use the same-quad mainframe routines if they can or else use the quad-specific mainframe routines.

The following tables show which addresses are used to decide which quad relative routines to assemble to.

Quad Relative Branch Instruction Bytes

<table>
<thead>
<tr>
<th>Quad 0</th>
<th>Quad 1</th>
<th>Quad 2</th>
<th>Quad 3</th>
<th>Same Quad</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3FF</td>
<td>400-7FF</td>
<td>800-BFF</td>
<td>C00-FFF</td>
<td></td>
</tr>
<tr>
<td>NCXQREL</td>
<td>349 08C</td>
<td>36D 08C</td>
<td>391 08C</td>
<td>3B5 08C</td>
</tr>
<tr>
<td>NCGOREL</td>
<td>341 08C</td>
<td>365 08C</td>
<td>389 08C</td>
<td>3AD 08C</td>
</tr>
</tbody>
</table>

* These are followed by the third byte containing the low 10 bits of the address to jump to.

Actual Instruction for Quad Relative Branches

<table>
<thead>
<tr>
<th>Quad 0</th>
<th>Quad 1</th>
<th>Quad 2</th>
<th>Quad 3</th>
<th>Same Quad</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCVQ</td>
<td>[GOSUB0]</td>
<td>[GOSUBL]</td>
<td>[GOSUB2]</td>
<td>[GOSUB3]</td>
</tr>
<tr>
<td>(Address)</td>
<td>23D2</td>
<td>23DB</td>
<td>23E4</td>
<td>23ED</td>
</tr>
<tr>
<td>NCVQ</td>
<td>[GOL0]</td>
<td>[GOL1]</td>
<td>[GOL2]</td>
<td>[GOL3]</td>
</tr>
<tr>
<td>(Address)</td>
<td>23DO</td>
<td>23D9</td>
<td>23E2</td>
<td>23EB</td>
</tr>
</tbody>
</table>

Actual Instruction in HP mnemonics

<table>
<thead>
<tr>
<th>Quad 0</th>
<th>Quad 1</th>
<th>Quad 2</th>
<th>Quad 3</th>
<th>Same Quad</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOSUB</td>
<td>[GOSUB0]</td>
<td>[GOSUBL]</td>
<td>[GOSUB2]</td>
<td>[GOSUB3]</td>
</tr>
<tr>
<td>GOSUB</td>
<td>[GOL0]</td>
<td>[GOL1]</td>
<td>[GOL2]</td>
<td>[GOL3]</td>
</tr>
</tbody>
</table>

Note that the actual instruction is always NCXQ to the appropriate label and the mainframe routine at the label determines if a return will be pushed or not making either a NCXQREL or a NCGOREL. You can't use a NCOO or a CGO to jump to one of these routines because these instructions do not push the return address, which is needed to know which page the jump was in.
It is possible to make a three byte jump that jumps on carry using CXQ [POOBAR]. These are not really useful since the carry must always be set or the program will execute the third byte of the jump instruction after skipping the first two. Likewise, never set the carry before a NCGOREL or NCQREL. The address that is pushed on the return stack for a three byte jump is the address to the THIRD word, so if the jump crosses a quad boundary, it will jump into the quad that contains the third word.
FAT DEFINITION

000,100 DEFRAK DEFRAK DEFRAK ADDRESS ;Obsolete
000,000 DEFRAK DEFRAK DEFRAK ADDRESS ;define MCODE function in same 4K ROM
000,000 DEFRAK DEFRAK DEFRAK ADDRESS ;define MCODE function in next 8K ROM
200,000 U4KDEF U4KDEF U4KDEF ADDRESS ;define USER CODE function in same 4K ROM
200,000 U4KDEF U4KDEF U4KDEF ADDRESS ;define USER CODE function in next 8K ROM

RETURN STACK AND RETURNS

180 GOTOC GOTOC GOTOCR ;jump to the address in C[6:3]
170 STR=C STR=C STRADPR ;push C[6:3] onto the return stack
1B0 C=STK C=STK CPOADPR ;pop the return stack into C[6:3]; put 0 in last location of stack
020 SPDAOD CLRSTN XQ>GO ;pop first address off return stack
360 RINR C RINR ?RINR ;return if carry set
3A0 RTNNC RTNNC ?RTNNC ;return if carry not set
3E0 RTN RCRTN ?RCRTN ;unconditional return

PERIPHERAL INSTRUCTIONS

Peripheral addresses for PERSLCT
00 No peripheral enabled
FB Timer
FC Card Reader
FD Display
FE Wand
10 Special display for halfnut versions

PERIPHERAL ACCESSING

3F0 PFAD=C PERSLCT PERSLCT 0 to F (hex) ;select the peripheral addressed in C[1:0]
024 SELPF PERIceland SELPF 0 to P (hex) ;allow peripheral to take control

PERIPHERAL FLAGS

02C FLG=F ?PF ?PI= 0 to 13 (dec) ;set carry if peripheral flag set
02C ?F3=1 ?PF 3 ?PI= 3 ;set carry if peripheral flag 3 set
06C ?F4=1 ?PF 4 ?PI= 4 ;set carry if peripheral flag 4 set
0AC ?F5=1 ?EADAV ?PI= 5 ;set carry if peripheral flag 5 set
0EC ?F10=1 ?OAV ?PI= 10 ;set carry if HP-IL output register available
12C ?F8=1 ?PRAV ?PI= 8 ;set carry if HP-IL frame available
16C ?F6=1 ?IFCR ?PI= 6 ;set carry if HP-IL interface clear received
1AC ?F11=1 ?TFAIL ?PI= 11 ;set carry if timer clock access failure
22C ?P2=1 ?WDB ?PI= 2 ;set carry if wand has data in wand buffer
26C ?P9=1 ?PRHI ?PI= 9 ;set carry if HP-IL frame not received as sent
2AC ?P7=1 ?SRQ ?PI= 7 ;set carry if service request received
2BC ?P13=1 ?SERV ?PI= 13 ;set carry if service request
32C ?PI=1 ?CRDR ?PI= 1 ;set carry if card reader flag set
36C ?P12=1 ?ALM ?PI= 12 ;set carry if alarm due
3AC ?P0=1 ?PBSSY ?PI= 0 ;set carry if peripheral flag 0 set
DISPLAY INSTRUCTIONS

Zencode display instructions:
WR/RD - Write/Read
A/B/C - to/from display registers A/B/C
1/4/6/12 - 1/4/6/12 characters to/from
R/L - Right/Left of display

Writing instructions cause the new characters to be pushed on the specified side which pushes the characters on the other end off into oblivion.

Reading instructions cause the characters to be taken off the specified side and pushed on the other side.

DISPLAY READING

028  FLIDA  RDA12L  FLIDA
068  FLIDB  RDB12L  FLIDB
0A8  FLIDC  RDC12L  FLIDC
0E8  FLIDAB  RDA6BL  FLIDAB
118  READEN  READAN  READEN  ; copy annunciators into C[2:0]
158  FLSDC  RDC1L  FLSDC
198  FRSDA  RDA1R  FRSDA
238  FRSDB  RDB1R  FRSDB
278  FRSDC  RDC1R  FRSDC
2B8  FLSDB  RDA1L  FLSDB
338  FRSDAB  RDA1R  FRSDAB
378  FLSDB  RDA1L  FLSDB
3B8  RABCR  RABC1R  RABCR
3F8  RABCL  RABC1L  RABCL

DISPLAY WRITING

028  SRLDA  WRA12L  SRLDA
068  SRLDB  WRB12L  SRLDB
0A8  SRLDC  WRC12L  SRLDC
0E8  SRLDAB  WRAB6L  SRLDAB
118  SLLDAB  WRABC4L  SLLDAB
158  SLLABC  WRABC4R  SLLABC
198  SRLDA  WRA1L  SRLDA
238  SRLDB  WRB1L  SRLDB
278  SRLDC  WRC1L  SRLDC
2B8  SLSDA  WRA1R  SLSDA
338  SLSDB  WRB1R  SLSDB
378  SLSDB  WRB1L  SLSDB
3B8  SLSABC  WRABC1L  SLSABC
3F8  SLSABC  WRABC1R  SLSABC
2F0  WRITEN  WRITAN  WRITEN  ; copy bits from C[2:0] into annunciators
TIME MODULE
028  WRTIME  WTIME  WRTIME
068  WDTIME  WTIME  WRTIME
0A8  WRAIM  WAIM  WRAIM
0E8  WRSTS  WSTS  WRSTS
128  WESCR  WSCR  WESCR
168  WSINT  WINTST  WSINT
1E8  STPINT  STPINT  STPINT
228  DSWKUP  WKUPOFF  DSWKUP
268  ENWKUP  WKUPON  ENWKUP
2A8  DSAIM  AIMOFF  DSALM
2E8  ENALM  ALMON  ENALM
328  STOGC  STOGC  STOGC
368  STARTC  STARTC  STARTC
3A8  PT=B  TIMER=A  PT=B
3E8  PT=A  TIMER=B  PT-A

TIME MODULE
038  RDTIME  RTIME  RDTIME
078  RCTIME  RTIMEST  RCTIME
0B8  RDAIM  RAIM  RDAIM
0F8  RDAIM  RAIM  RDAIM
138  RDSTS  RSTS  RDSTS
178  RDINT  RINT  RDINT

CARD READER
028  ENWRIT  ENDWRIT  ENWRIT
068  STWRIT  STWRIT  STWRIT
0A8  ENREAD  ENREAD  ENREAD
0E8  STREAD  STREAD  STREAD
168  CRDWPF  CRDWPF  CRDWPF
1E8  CRDINF  CRDINF  CRDINF
228  CRDOHF  CRDOHF  CRDOHF
268  CRDOHF  CRDOHF  CRDOHF
328  TSTBUF  TSTBUF  TSTBUF
368  TRPCRD  SETCTR  TRPCRD
3E8  TClipG  TClipG  TClipG

INTELLIGENT PERIPHERAL
200  HPIL=C  HPIL=C  HPIL=C  0 to 7 ;copy C[1:0] to HP-IL register
VARIATIONS
062 BAEX B<0A B<0A TEF
0A2 CAEX C<0A C<0A TEF
0E2 CBEX C<0B C<0B TEF
202 C=C+A C=C+A C=A+C TEF
1D8 MCX N<0C N<0C
0F0 NCX N<0C N<0C

VARIATIONS AND DUPLICATES
0E2 C=0? ?C=0 ?C=0 TEF
22 C=0? ?B=0 ?B=0 TEF
30 A<0? ?A<0 ?A<0 TEF
34 A<0? ?A<0 ?A<0 TEF
36 A<0? ?A<0 ?A<0 TEF
37 C=C1A C=C0A C=C0A
3B0 C=C.A C=CANDA C=CANDA
3B8 FRSABC RDABC1 RFRSABC
0BC ORAV? ?ORAV ?PI= 10
12C FRAV? ?FRAV FRAV?
26C FRNS? ?FRNS ?PI= 9
2AC SRQR? ?SRQR SRQR?
36C ALC Across? ?ALM ?PI= 12
160 P=Q? ?P=Q ?P=Q
120 P=Q? ?P=Q ?R= 0 to 13 (dec)
001,000 GOSUB NCX? NCX? ADDRESS
001,002 GOLONG NCGO ?NCGO ADDRESS
003 GOTO JNC JNC -64 to +63 (dec)
024 HPL=CH PERTCT SELFPEF 0 to P (hex)

034 S0= 0 CF 0 CLR0 0
034 S1= 0 CF 1 CLR0 1
024 S2= 0 CF 2 CLR0 2
004 S3= 0 CF 3 CLR0 3
044 S4= 0 CF 4 CLR0 4
084 S5= 0 CF 5 CLR0 5
144 S6= 0 CF 6 CLR0 6
244 S7= 0 CF 7 CLR0 7
304 S8= 0 CF 8 CLR0 8
244 S9= 0 CF 9 CLR0 9
0C4 S10= 0 CF 10 CLR0 10
184 S11= 0 CF 11 CLR0 11
344 S12= 0 CF 12 CLR0 12
2C4 S13= 0 CF 13 CLR0 13

388 S0= 1 SF 0 SET0 0
08 S1= 1 SF 1 SET0 1
08 S2= 1 SF 2 SET0 2
08 S3= 1 SF 3 SET0 3
048 S4= 1 SF 4 SET0 4
088 \ S5= 1 \ SF 5 \ SETF 5
148 \ S6= 1 \ SF 6 \ SETF 6
288 \ S7= 1 \ SF 7 \ SETF 7
108 \ S8= 1 \ SF 8 \ SETF 8
248 \ S9= 1 \ SF 9 \ SETF 9
0CB \ S10= 1 \ SF 10 \ SETF 10
188 \ S11= 1 \ SF 11 \ SETF 11
348 \ S12= 1 \ SF 12 \ SETF 12
2CB \ S13= 1 \ SF 13 \ SETF 13
38C \ T50=1 \ TP 0 \ TSET 0
30C \ T51=1 \ TP 1 \ TSET 1
20C \ T52=1 \ TP 2 \ TSET 2
00C \ T53=1 \ TP 3 \ TSET 3
04C \ T54=1 \ TP 4 \ TSET 4
08C \ T55=1 \ TP 5 \ TSET 5
14C \ T56=1 \ TP 6 \ TSET 6
28C \ T57=1 \ TP 7 \ TSET 7
10C \ T58=1 \ TP 8 \ TSET 8
24C \ T59=1 \ TP 9 \ TSET 9
0CC \ T510=1 \ TP 10 \ TSET 10
18C \ T511=1 \ TP 11 \ TSET 11
34C \ T512=1 \ TP 12 \ TSET 12
20C \ T513=1 \ TP 13 \ TSET 13