DECUS
PROGRAM LIBRARY

DECUS NO.	8-257
TITLE	UCONN-EAP, EDITOR-ASSEMBLER
AUTHOR	Gerald E. Zajac Submitted by Howard A. Sholl
COMPANY	University of Connecticut Digital Systems Laboratory Storrs, Connecticut
DATE	Submitted August 7, 1969
SOURCE LANGUAGE	MACRO-8

UCONN-EAP, EDITOR-ASSEMBLER

DECUS Program Library Write-up DECUS No. 8-257

GEZ

Abstract

UCONN-EAP is a combined symbolic editor and assembler written for a DEC PDP-5 or PDP-8 computer with a 4K memory. This program is compatible with the manufacturer supplied software in that it retains all the essential commands of the Symbloic Tape Editor and all the features of the PAL III assembler. In addition the assembler will also handle literals.

The symbolic program to be assembled must be stored in memory, and is thus limited to about 3070 characters which corresponds to an assembled program of one to two pages in length depending upon how heavily the symbolic program is commented.

Since this program eliminates many steps involved in correcting assembly errors such as reloading separate editor and assembler programs, it should be extremely useful to someone learning to program a PDP-5 or PDP-8.

TABLE OF CONTENTS

PAGE

Introduction -- 1

Editor -- 3
    Control Characters -- 3
    Commands -- 4
    Summary of Editor Commands ---------------------------------- 8

Assembler --- 9
    Characters -- 12
    Definition of Terms --- 13
        Integers -- 13
        Symbols --- 13
        Symbol Definition --------------------------------------- 13
        Expression -- 14
        Current Address Indicator ------------------------------- 16
        Literals -- 16
        Comments -- 18
        Set Current Location Counter ---------------------------- 18
        Terminate Current Pass ---------------------------------- 18
    Psuedo-Instructions --- 19
    Error Messages -- 21
    Operating Instructions -------------------------------------- 22
        Switch Options -- 22
    UCONN-EAP Symbol Table -------------------------------------- 23

Introduction

UCONN-EAP is a Symbol Editor-Assembler program that will edit and assemble a symbolic program without having to load a separate program to perform each function. It also eliminates the need for punching a new symbolic tape, in order to reassemble a program, everytime a change is made in the symbolic program. In addition, the assembled program may be loaded and run without disturbing UCONN-EAP, if it is restricted to a particular area in storage. Then if the programmer is not satisfied with his program he can return to the editor-assembler, make the appropriate symbolic changes, and reassemble his program; consequently, UCONN-EAP will save a programmer many steps and much time in obtaining a satisfactory program. However, to achieve the convenience and efficiency of UCONN-EAP, the symbolic program to be operated on must be stored in memory; consequently, it is limited to 3072 text characters. This corresponds to a program of approximately one to two pages in length.

Although the length of the program to be operated on is limited, this editor assembler should nevertheless be extremely useful and efficient for novice programmers who will usually be writing short programs and making frequent errors. Since the rules for using UCONN-EAP are very similar to those of the existing manufacturer-supplied software, a minimum of relearning is required to use MACRO-8 or the Symbolic Tape Editor.

All the essential commands of the Symbolic Tape Editor are retained in UCONN-EAP. The assembler contains all features of PAL III, and also includes one of the most useful MACRO-8 feature in that it will handle literals. By a switch option UCONN-EAP will either punch a binary tape with a check sum or produce an octal-symbol listing.

Complete error checking is also provided during assembly. When an error is detected, a message is printed which gives an error code and the decimal line number in the symbolic program where it occurred. After an error message is printed, the command mode of the editor is reentered to permit the programmer to make any corrections.

A switch option also permits testing a program for errors by suppressing punching of a binary tape. Thus a program can be tested for errors in a matter of a few seconds.

The Editor

The editor will perform all essential operations required for editing lines of text. The repertoire of commands permit listing, changing, deleting and inserting lines of text. Data can be entered from either the Teletype keyboard or a paper tape reader. Output of text can either be done by listing the text buffer contents or punching an ASCII tape of the contents. A command is also available to punch leader-trailer code for ASCII tapes.

The editor will store up to 3072 characters or 255 lines of text. If either of these values is exceeded, input stops, and an error message is printed. At this time the editor returns to the command mode.

The editor operates in two modes - command mode and text mode. When UCONN-EAP is started, it is initially in the command mode; and as the name implies, commands are accepted from the Teletype keyboard under this mode.

A command consists of a single alphabetic character representing an operation or a single alphabetic character preceded by one or two decimal arguments specifying lines to be operated on. After typing a command, a carriage return will cause the editor to leave the command mode and enter the text mode to execute the command. If a non-existant command code or an argument specifying a line not in the buffer is given, the request will be echoed by a question mark, and the editor will wait for a new command.

Under most commands, the editor will return automatically to the command mode when the operation is complete; however, under commands that require input of text, a form feed character is necessary to terminate the input.

CONTROL CHARACTERS

Carriage Return (↓)

The carriage return is used to terminate commands given by the operator and to terminate lines of text. When text is entered from the keyboard, it terminates the current line and advances the teleprinter one line.

<u>Form Feed</u>

When this character is detected while in the text mode, the bell will be rung; and the editor will return to the command mode. To enter this character the Control key must be held down while the Form key is pressed.

<u>TAB (→|)</u>

This character is used for formatting. Tabs are set 8 characters apart; thus, whenever the Control and Tab key are pressed, spaces are stored in the buffer until the next tab position is reached.

Although use of the tab in formatting produces a neat and easily read symbolic program listing, its use should be kept to a minimum on lengthy programs since the spaces inserted consume storage area.

<u>Back Arrow (←)</u>

The arrow causes the contents of the current line to be cancelled. The text following this character is then entered on the same line until a carriage return is given. This character is effective both when entering text from the keyboard and when reading tapes.

COMMANDS

<u>APPEND</u>

A↵

This command will cause new lines of text to be appended to the end of the text buffer from the Teletype keyboard until a form feed character causes a return to command mode. No numerical arguments are permitted before the A.

This command permits preparation of symbolic program tapes on line since it is the only command that will open an empty buffer for input of text from the keyboard.

<u>CHANGE</u>

nC↵

n,mC↵

This command must have one or two numerical arguments. nC will cause line n to be replaced by the lines entered from the keyboard, and n,mC replaces lines n through m inclusively. Any number of lines may be inserted from the keyboard until a form feed is given to return to the command mode.

DELETE

nD ↲

n,mD ↲

The delete command will delete the line specified by n or lines n through m inclusively.

Since lines that are deleted or replaced are ignored rather than erased and used for storage of new text, it is possible to overflow the buffer if extensive changes are made. When this occurs the following error message is printed:

```
        BF LINE XXX
```

BF is the error code, and XXX is the number of lines in the buffer at the time of the overflow.

FORM FEED

F ↲

Before executing this command, the program will halt to permit the Teletype punch to be turned on. Pressing the Continue switch will cause six leader characters (200) to be punched then a form feed character followed by about six inches of leader code.

This command is useful for segmenting long tapes. That is, if this tape is read at a later time for editing or assembly, reading will stop whenever a form feed character is encountered; thus, the entire tape need not be entered in the text buffer. However, the form feed is not necessary at the end of a tape.

INSERT

nI ↲

Lines entered from the keyboard will be inserted before the line specified by the single argument n. If two numerical arguments are given, the second will be ignored. Any number of lines can be inserted from the keyboard until a form feed character is given to terminate the entry.

KILL

K↲

The entire contents of the buffer will be deleted. No numerical arguments are permitted.

LIST

L↲

nL↲

n,mL↲

The lines specified will be printed on the teleprinter. If no arguments are given with the L, the entire buffer will be listed. nL causes line n to be printed; and n,mL causes lines n through m inclusively to be printed. The editor returns to the command mode after completing the listing.

PUNCH

P↲

nP↲

n,mP↲

Before executing this command, the program will halt to allow the Teletype punch to be turned on. Then pressing the Continue switch will cause the lines specified to be punched. No argument with the command causes the entire buffer to be punched. A single line will be punched if only one argument is given; if two arguments are given, lines n through m inclusively will be punched.

READ

R↲

Paper tape will be read on the high speed paper tape reader until a form feed character is detected. This text will be appended onto the present contents of the buffer. Rubouts, line feeds, leader-trailer code (200), and blank tape is ignored.

Although UCONN-EAP was written to read paper tapes on a high speed reader, a low speed reader can be used by changing the contents of addresses 3152 and 3154 to 6031 and 6036 respectively.

<u>TRAILER</u>

T ↲

Before executing, the program will halt to permit the Teletype punch to be turned on. Pressing the Continue switch, will cause about six inches of leader-trailer to be punched.

<u>SYMBOL PRINT</u>

S ↲

This command will punch about six inches of leader code, and then punch and list the user defined symbols that were entered in the assembler symbol table on the last assembly operation. The symbols are printed in the order that they were defined - the symbol is printed first followed by its octal value. After the symbols are listed, about six inches of trailer is punched.

<u>ASSEMBLE</u>

X ↲

Control is transferred to the assembler and the symbolic program in the buffer is assembled.

Summary of Editor Commands

Command	Function
A↵	Append lines to end of buffer from the keyboard.
nC↵	Change line n.
n,mC↵	Change lines n through m.
nD↵	Delete line n.
n,mD↵	Delete lines n through m.
F↵	Punch form feed character and trailer code.
nI↵	Insert lines before line n.
K↵	Kill the entire buffer contents.
L↵	List entire buffer.
nL↵	List line n.
n,mL↵	List lines n through m.
P↵	Punch entire buffer.
nP↵	Punch line n.
n,mP↵	Punch lines n through m.
R↵	Read tape and append to end of buffer.
S↵	List symbols from last assembly.
T↵	Punch six inches of trailer code.
X↵	Assemble the symbolic program contained in the text buffer.

ASSEMBLER

This user's guide for the UCONN-EAP assembler assumes a knowledge of the basic rules for writing PDP-5/8 symbolic programs; however, if it is desired to learn or review this material, it is recommended that Sections 1 through 4 of MACRO-8 Programming Manual (Dec-08-CMAA-D) be read before reading this guide. This material will explain some of the terms used in this manual and also gives the instruction set of the PDP-8 which is the same as that of the PDP-5.

This assembler duplicates the PAL III assembler completely; and in addition it will also handle literals - a useful feature of MACRO-8; however, this is the only MACRO-8 feature contained in this assembler.

The floating point instructions are contained in the symbol table of this assembler and to use these the user is referred to the PDP-8 Floating-Point System Programming Manual (8-5-S).

When UCONN-EAP is started, it is in the command mode of the editor; thus to assemble a program, an X followed by a carriage return must be typed on the teletype keyboard. After assembly is complete, UCONN-EAP returns to the command mode. However, before the command to assemble is given, the switch register must be set for the desired output from the assembler. Depending upon the switch settings, the assembler will produce a binary object program tape, an octal-symbolic program listing, or all output except error messages can be suppressed. The final option saves much time in testing a program for assembly errors, and whenever an error is detected, UCONN-EAP returns to the command mode of the editor to allow corrections to be made. These options are discussed in detail under the section on Operating Instructions.

After assembly is completed a list of the user defined symbols and their respective octal values can be obtained by typing an S and a carriage return. If the teletype punch is turned on at this time, about six inches of leader code will be punched, then an ASCII listing of the symbols followed by another six inches of leader-trailer code. The symbols are listed in the order that they were defined. This listing which gives the location of the symbols is useful for debugging a program if the programmer does not wish to take the time to make an octal symbolic listing of his program; moreover, a tape containing the symbols and their values is essential if the DEC Dynamic Debugging Technique (DDT) is to be used for debugging a program.

An additional feature of this editor-assembler program is that an area is reserved for loading a programmer's object program. Page 1 (200-377), which is used as a scratch pad and temporary storage area when the assembler is operating and locations 1 through 60 on page 0 with the exception of location 10, are available for loading of an object program when UCONN-EAP is not running. The memory map in Figure 1 indicates which locations are available. Once the object program is loaded, it can be run to test its operation. Then if the programmer is not satisfied, he can restart UCONN-EAP, correct the symbolic program, and reassemble the program providing that his program did not alter any locations in UCONN-EAP.

The programmer may also load his object program in any unused portion of the text buffer which uses locations 4600 through 7600. The last used address of the text buffer can be determined by examining the pointer in location 10.

0000	PDP-5 Program counter (not available for any program)
0001 0007	Not used by UCONN-EAP Available for user programs
0010	UCONN-EAP pointer, not available for user programs
0011 0060	Temporary storage for UCONN-EAP Available for user programs when UCONN-EAP is not running
0061 0177	UCONN-EAP Not available for user programs
0200 0377	Scratch pad area for UCONN-EAP Available for user programs when UCONN-EAP is not running
0400 ≈ 4577	UCONN-EAP Not available for user programs
4600 ≈ 7577	Text buffer of UCONN-EAP Location 0010 will give last used location, and programs may be loaded above the last used location without destroying contents of buffer
7600 7777	Binary Loader program Not available for user programs

Figure #1

CHARACTERS

In writing symbolic programs the ASCII characters permitted are:

Letters:

ABCD......XYZ

Digits:

1 2 3 4 5 6 7 8 9 0

Non-printing Punctuation Characters:

Space

Carriage return

Although tabs are used for formatting, the assembler never sees them because the editor inserts the appropriate number of spaces when the symbolic program is stored in the text buffer.

The following characters are used to specify operations to be performed upon symbols or numbers.

Character	Use
Space	Combine symbols or numbers
+ plus	Combine symbols or numbers (addition)
- minus	Combine symbols or numbers (subtraction)
⤸ carriage return	Terminate line
, comma	Assign symbolic address to preceding symbol
= equal	Define parameter
* asterisk	Set current location counter
. period	Has value of current program counter
; semicolon	Terminate coding line
$ dollar sign	End of symbol program
/ slash	Indicates start of comment
() parentheses	Define literal on current page

All characters that are ignored by PAL III or MACRO-8 are ignored by the EAP editor when the symbolic program is entered into the buffer from paper tape. These characters are:

Form-feed
Blank Tape
Rubouts
Code 200
Line Feed

DEFINITION OF TERMS

INTEGERS

This assembler will accept only octal integers. The digits 8 or 9 appearing in any number will give an illegal character error. The integers may be composed of one to four octal digits. If more than four digits occur, the fifth will be interpreted as an illegal character.

SYMBOLS

A symbol is a string of alphameric characters obeying the following rules:

1. The first character must be alphabetic.
2. The remaining characters are either alphabetic (A-Z) or numeric (0-9).
3. Only the first six characters are meaningful. Any additional characters are ignored.
4. A symbol is terminated by any non-alphanumeric character.

SYMBOL DEFINITION

There are two methods of assigning a value to a symbol.

Parameter assignment:

A symbol is assigned the value of an algebraic expression by means of an equal sign. For example

```
B=CAT+DOG
```

B will be assigned the value of CAT + DOG; however, the value of CAT and DOG must have been previously defined. Also no space is allowed between B and the equal sign. The program counter is not incremented when a symbol is defined with an equal sign.

TAG:

If a symbol is the first on the line and it is followed by a comma, it will be assigned the value of the current location counter. For example:

```
   *200
   CLA
RAT,TADX
   .....
```

RAT will be assigned a value of 201.

If at any time an attempt is made to redefine a previously defined symbol with either the comma or the equal sign, an illegal redefinition error will occur and assembly will be terminated. Control then returns to the editor

<u>EXPRESSION</u>

Symbols and numbers can be combined by the following operators:

+ plus	2's complement addition
- minus	2's complement subtraction
space	the space may signify an inclusive OR or it may act as a delimiter.

If the operation is to be addition or subtraction, no spaces are allowed between the operator and either the preceding or following symbol or number; and if the OR operation is to be performed, the space must be the only operator between the symbols or numbers. The operations are performed as they are encountered in moving from left to right across the expression.

For example:

```
A=D+E F-G
```

First the 2's complement addition of D and E is performed. Next this sum is OR'ed with F, and then G is subtracted from the result.

However, the space acts as an address delimiter if it immediately follows the mnemonic of a memory reference instruction when this mnemonic if the first symbol.

For example:

```
    *3725
A,  0
    ...
    TAD A
```

A is a user defined symbol with a value of 3725, and the expression TAD A is evaluated as follows:

The 5 high order bits of A are taken as shown:

```
A  011   11   1   010   101
   011   11
```

These five bits are tested for all zeros, for a page 0 reference, and if they are not, they are then compared against the page bits of the current location counter to see if it is a current page reference. If it is a current page address, the current page bit (bit 5) is set.

```
TAD  001   010   000   000
```

Next the seven low order bits of A are OR'ed with the operation code to obtain:

```
TAD  A 001   011   010   101
```

or more concisely as an octal number:

```
1325
```

If it had happened that A was not on the zero page or on the current page where TAD A occured, an illegal reference error message would have been given.

If the instruction had been:

```
TAD I A
```

the "I" would mean that A is to be used as an indirect address. Thus, the indirect address bit (bit 3) would also have been set to give:

```
     TAD I A  001  111  010  101
```

or more concisely: 1725

The address field of a memory reference instruction may also be any valid expression. For example:

```
     TAD A+30
```

CURRENT ADDRESS INDICATOR

The period (.) has a value equal to the current location counter, and may be used in any expression. For example:

```
          *300
          TSF
          JMP .-1
```

JMP .-1 is equivalent to JMP 300 and will cause a jump back to the TSF instruction. Or:

```
          *300
          .+20
```

would produce in location 300, the number 320.

LITERALS

The use of the literal provides a means of using a constant without the programmer explicitly reserving a location for the constant. For example, adding five to the accumulator could be coded as:

```
          ...
          TAD B
          TAD BDX
          ...
          ...
     BDX, 5
```

But by using a literal, it would be coded as:

```
...
TAD B
TAD (5)
...
```

The left parenthesis tells the assembler that 5 is to be stored in the constants table at the end of the current page. The instruction in which the literal appears is then encoded with an address referring to the location where the constant is stored. The first literal encountered after the program counter is set, is assigned to the last location on the current page (page address 177). The next literal, if it is not equal to the first, is stored at page address 176, and so on. A literal is stored only when first encountered; any subsequent references are to this location.

Any valid expression may be enclosed within the parenthesis, and the right parenthesis may be omitted. The following examples are acceptable:

```
AND (177
TAD (CLA+100
```

The last example, CLA is a permanent symbol with a value of 7200; thus, the last example is equivalent to:

```
TAD (7300
```

The following example is illegal:

```
TAD (5)+(7)
```

Literals may not be nested. If a binary tape is being punched, the literals are punched at the end of every page in which they are used. If the program counter is reset to a previously used page, the assembler has no way of knowing this; thus, any literals generated this time will be stored over those previously generated when the object program is loaded into memory.

COMMENTS

Comments may be placed at any point in a program. They must start with a slash and end with a semicolon or carriage return. If a semicolon occurs within a comment, the characters following the semicolon will be interpreted as the start of another instruction on a new line; thus, any printing characters except the semicolon may appear within a comment.

SET CURRENT LOCATION COUNTER

The current location counter is reset whenever an asterisk (*) is encountered as the first non-space character on a line. The current location counter is set to the value of the expression following the asterisk.

For example:

```
*370
*D+20
```

are acceptable, however, D must have been previously defined.

The starting address of a program is initially set to 200 by the assembler; thus, 200 will be the origin if none is specified by the programmer.

TERMINATE CURRENT PASS

The dollar sign $ encountered as the first non-space character of a line signals the assembler to terminate the current assembly pass. If no $ is placed at the end of a program, the assembler will attempt to assemble until the last line of the text buffer is encountered, and if it does reach the end of the buffer, it will print an error message consisting of:

```
$?  LINE XXX
```

XXX is the last line of the buffer plus one.

PSUEDO-INSTRUCTIONS

The psuedo instructions are directions to the assembler to perform specified tasks. In UCONN-EAP, there are only two psuedo-instructions, EXPUNGE and FIXTAB. These are used for altering the permanent symbol table.

EXPUNGE will expunge the entire symbol table except for the psuedo-instructions.

FIXTAB fixes all the symbols that are currently in the symbol table.

For example:

```
        *200
        SMSF=6101
BEGIN,  CLA
        TAD A
        JMP X
A,      0240
        FIXTAB
X,      DCA B
        ...
```

In the above example, all the symbols defined before the FIXTAB will become part of the permanent symbol table. That is, SMSF, BEGIN, and A will be added to the permanent symbol table and will have the values 6101, 0200, and 0203 respectively.

```
        EXPUNGE
        TAD=1000
        JMP=5000
        CAT=6010
        FIXTAB
        $
```

In this example the EXPUNGE psuedo-instruction tells the assembler to delete the entire symbol table. Following this, the symbols TAD, JMP, and CAT are entered in the user symbol table, but upon encountering the FIXTAB these three symbols are placed in the permanent symbol table; consequently, only these three symbols are now in the permanent symbol table.

Once these psuedo instructions are used to alter the symbol table, the program will have to be reloaded to restore the original symbol table.

To obtain a new program tape after the symbol table is altered, the text buffer should be killed (see kill command under Editor) to reset the pointers. A binary punch routine can then be used to punch a new binary tape of the program by punching all locations between 10_8 and the last location occupied by the new symbol table. This last address can easily be determined by examining the pointer in location 0140.

ERROR MESSAGES

The general form of the error message is:

```
CC XXXXXX LINE YYY
```

CC is a two character code specifying the type of error. If XXXXXX is printed, it is the character or symbol in question, and YYY is the decimal number of the line in the symbolic program where the error occurred.

Whenever an error occurs UCONN-EAP returns to the command mode of the editor so that corrections can be made.

ERROR CODES

BF	Either the text buffer or the line location table is full; that is, an attempt was made to store more than 3072 characters or more than 255 lines. The line number given is the last line stored. This is the only non-assembly error.
IC	Illegal Character. A non-alphameric character appeared in a symbol, some character other than an octal digit occurred in an octal number, or more than 4 digits appeared in an octal number.
IR	Illegal Reference. An off-page reference was made to some page other than page 0.
PE	Page Exceeded. The number of locations used on a page exceeded those available.
RD	Illegal Redefinition. An attempt was made to redefine a symbol by use of a comma or an equal sign.
ST	Symbol Table-Full. 124_{10} permanent and user-defined symbols can be stored.
US	Undefined Symbol. A symbol was used that had not been previously defined.
$?	No dollar sign occurred before the end of the text buffer was reached.

OPERATING INSTRUCTIONS

To assemble a program with UCONN-EAP:

1. Load UCONN-EAP with the Binary Loader.
2. Set the Switch Register to 0400.
3. Depress Load Address.
4. Depress Start.

Now that UCONN-EAP is running the symbolic program can be typed in from the 33-ASR keyboard or read in from paper tape. (See section on the Editor for the appropriate commands.)

When the symbolic program is stored in the text buffer, the switch register can be set as explained below to obtain a binary object program tape, an octal-symbolic listing, or all output except error messages will be suppressed during assembly. All output is on the 33-ASR Printer/Punch. Once the switch register is set for the desired option, UCONN-EAP can be told to assemble the symbolic program stored in the text buffer by typing an "X" and a carriage return.

After assembly is completed and UCONN-EAP has returned to the command mode, the user defined symbols can be listed by typing an "S" and a carriage return.

SWITCH OPTIONS

Switch Up	
None	Punch a binary object tape
0	Print an octal symbolic listing
1	Suppress punching binary object tape
0 and 1	Octal symbolic listing will be printed but leader and trailer will be suppressed
10	Retain user symbols from last assembly operation

If switch 10 is down (off) user symbols are cleared from the symbol table each time assembly mode is entered.

UCONN-EAP SYMBOL TABLE

Pseudo Instructions

EXPUNGE

FIXTAB

I

Z

Memory Reference Instructions

AND	0000
TAD	1000
ISZ	2000
DCA	3000
JMS	4000
JMP	5000

Microinstructions

NOP	7000
CLA	7200
CLL	7100
CMA	7040
CML	7020
RAR	7010
RTR	7012
RAL	7004
RTL	7006
IAC	7001
SMA	7500
SZA	7440
SPA	7510
SNA	7450

SNL	7420
SZL	7430
SKP	7410
OSR	7404
HLT	7402

Combined Microinstruction

CIA	7041
LAS	7604
STA	7240
STL	7120
GLK	7204

Program Interrupt

ION	6001
IOF	6002

Analog to Digital Converter

ADC	6004

High Speed Perforated Tape Reader

RSF	6011
RRB	6012
RFC	6014

Teletype Keyboard Reader

KSF	6031
KCC	6032
KRS	6034
KRB	6036

Teletype Teleprinter/Punch

TSF	6041
TCF	6042
TPC	6044
TLS	6046

Floating Point Interpretive Commands

FEXT	0000
FADD	1000
FSUB	2000
FMPY	3000
FDIV	4000
FGET	5000
FPUT	6000
FNOR	7000

Oscilloscope Display, Type 34B

DCX	6051
DXL	6053
DCY	6061
DYL	6063
DIX	6054
DIY	6064
DXS	6057
DYS	6067