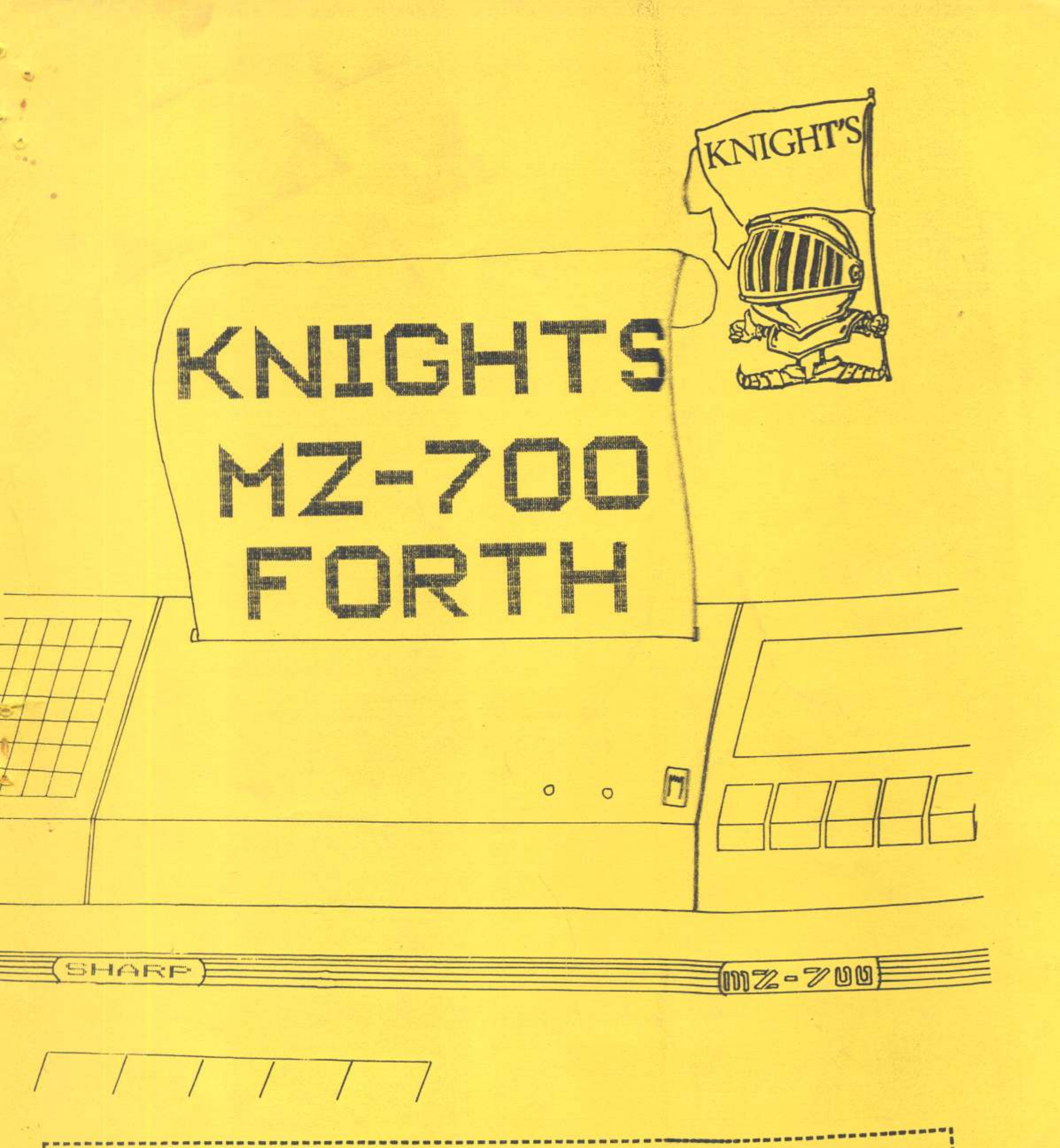


INCLUDES: FORTH LANGUAGE AND
10 APPLICATIONS
PROGRAMS

KNIGHTS COMPUTERS 108 ROSEMOUNT PLACE, ABERDEEN. TEL 0224 630526

KNIGHTS MZ-700 FORTH

LOADING THE LANGUAGE TAPE

Switch on your MZ-700 and insert the language tape in the cassette unit. Type L & press the CR key then depress the play button on the tape recorder: "LOADING MZ-700 Forth" appears.

"*Knights MZ-700 Forth Mem=AFFF 6B0B" appears when the tape stops.

indicates that Knights Forth is ready to use in the direct mode.

TWO MODES

Forth has two distinct modes. Text editor mode is indicated by a * prompt. Keywords can be built up - either by loading saved keyword definitions from tape, or using the editor commands to write new Forth programs. Direct mode is indicated by a ## prompt, and is used to run programs and test keywords.

PROGRAMMING IN FORTH

The essence of Forth programming lies in defining new Forth words which use previously defined commands to perform the desired task. Effectively the programmer tailors the language to meet his own specifications. It is this aspect of the language that makes Forth such fun to use. Calculations are stack based and utilise Reverse Polish Notation (this will be explained later), making programming in Forth a challenging alternative to languages such as BASIC or PASCAL.

TYPES OF WORD IN MZ-700 FORTH

F0 Machine language word i.e. the relevant procedure is defined by a section of machine code. The SP register of the Z80 points to the data stack, control is passed to the interpreter again by means of the JP(IY) instruction.

F1 Word defined using previously defined words and constants. Most of the definitions you write will be of this type.

F2 Variables - when a variable name is used the storage address is pushed on to the top of the data stack. This then allows you to store numeric values at that address; the net result is similar to integer variables in BASIC. Forth also deals with strings, when a string is presented to the interpreter it is stored on the data stack as two integers, the first being the address in memory where the string has been stored, the second is the length of the string.

F5 Constant - similar in principle to an integer variable, however as the name implies the initial value on assignment remains unchanged.

F6 Forth word defined by the system.

TEXT EDITOR COMMANDS

The text editor uses the concept of a current line pointer.

COMMANDS

B : move line pointer to the top of the text buffer.

Tn : list n lines from current line without altering position of the current line pointer (If n is omitted then all the lines after the LP are listed). Lines listed in this manner can be screen edited. The space bar can be used to pause the listing.

Pn : place the LP on the nth line from the top of text (when n is omitted the LP is placed on the final line).

+n : move line pointer n lines forward

—n : move LP n lines backwards.

Dn : deletes n lines after LP (If n is omitted then all lines after the LP are deleted).

I : inserts text immediately before LP. To end insertion of text press CR on a blank line or else press shift and break at the same time.

Sstring : search text after LP for string - if found, display line and place LP on next line. If the string is not located the LP is moved to the final line.

F : compile source text from LP onwards. As each line is compiled it is listed to the screen.

Wstring : writes source text to tape with filename string.

L : load text from tape.

= : display size of source text.

! : exclamation mark returns Forth to RUN mode.

& : clear text buffer.

% : denotes a comment in Forth. Any line with this as the first character will be ignored by the compiler.

ERROR MESSAGES

XXX DATA? Word XXX has not been defined.

EMPTY Data stack is empty.

THEN? There is nowhere for control to be passed to after THEN.

DO—LOOP? Loop not closed

BEGIN—END? Loop not closed.

NOT FOUND Specified word to be forgotten has not been declared.

AVAIL? Memory error.

ARITHMETIC OPERATORS

+	F0	2nd + TOP→TOP e.g. 3 2 + places 5 on TOP of the data stack.
—	F0	Subtraction e.g. 3 2 - places 1 on TOP of data stack.
*	F0	Multiplication e.g. 3 2 * places 6 on TOP of the data stack.
/M	F0	Integer division e.g. 3 2 / places 1 on the TOP of the data stack.
/	F0	Places quotient on TOP and remainder on 2nd e.g.5 2 /M TOP=2 2nd=1.
OR		F0 2nd OR TOP is placed on TOP of data stack. e.g. $231 $120F OR puts $123F on TOP of stack.
AND	F0	2nd AND TOP is placed on TOP of data stack. e.g.$231 $120F AND puts $201 on TOP of the stack.
XOR	F0	2nd XOR TOP is placed on TOP of data stack e.g. $231 $120F XOR puts $103E on TOP of the stack.
=	F0	If 2nd=TOP then TOP becomes -1 else TOP becomes 0 e.g. 3 3 = places -1 on TOP of the data stack.
<	F0	If 2nd<TOP then TOP becomes -1 else TOP becomes 0 e.g. 2 3 < TOP becomes -1.
>	F0	If 2nd > TOP then TOP becomes -1 else TOP becomes 0 e.g. 2 2 > TOP becomes 0.
MV+	F0	This command allows a block of memory to be moved from the address stored in 3rd top to the address stored in 2nd top. The size of the block to be moved is stored in TOP. The + sign denotes the fact that the block is to stored upwards from the new location. e.g. $1000 $D000 $100 MV+ has the effect of moving $100 bytes starting from $1000 and storing them from $D000 onwards.
MV	F0	Performs in a similar manner to MV+ except that the command operates downwards in memory. e.g. $1000 $D000 $100 MV- has the effect of taking $100 bytes from address $1000 downwards and moving them to address $D000 downwards.
@	F0	Takes the value of TOP as an address and places the contents of the locations pointed in TOP. e.g. $1000 @ puts the 16 bit number stored in locations $1000 and $1001 in TOP.
@B	F0	As for @ except that only 1 byte is transferred. The effect is similar to the PEEK command in BASIC.
!	F0	Takes a 16 bit number from 2nd and places in the address specified by TOP. e.g. $1234 $1000 ! location $1000 will contain $34 and $1001 will contain $12.

WORD	TYPE	USE AND EXPLANATION
[	F6	[W1 W2 .... WN] is used to define a new Forth language command using previously defined words W1 to WN. e.g. [NEG 0 SWAP -] defines a new Forth word NEG which when called negates the value on TOP of the data stack. NEG may then be used in other more important definitions. Words inside the definition must be separated by either a space or an end of line character. N.B. a right hand square bracket terminates the word definition.
M[	F6	Defines a word using machine language. e.g. M[BYE $C3 0 0]M defines a word BYE which when used returns the computer to monitor by executing an instruction similar to the USR(0) in BASIC.
VAR	F0	Allocates 2 bytes of store to be referenced by a variable name. The initial value of the variable is taken from the TOP of the data stack. e.g. -1 VAR X will declare a variable X with initial value -1.
,	F0	Extends the most recently defined variable or any other definition by 2 bytes and alters the TOP of the stack accordingly. e.g. -1 VAR X 0 , defines X and extends it by 2 bytes. As a result X is a 4 byte word and has content FFFF0000 hex.
CONST	F6	Defines a constant, the value of which is taken to be the top of the data stack. e.g. 1 CONST ONE declares a constant ONE of value 1. Therefore ONE ONE + . displays 2.
FORGET XXX	F6	removes word XXX from dictionary stack. Any words defined after the word to be forgotten are also removed.
.	F0	Removes the value from the top of the data stack and displays it on the screen in the current base.
#.	F0	Removes the number from the top of the data stack and displays it as an unsigned base 10 number (0-65536) e.g. -1 #. will display 65535.
$.	F0	Removes the value from the TOP of the data stack and displays as an unsigned base 16 number in the range (0-FFFF).
S.	F0	Prints a string to the screen. The TOP of the data stack must contain the address of the string, and 2nd TOP the length of the string.
LF	F6	Outputs a line feed to the screen.
?	F6	Prompts for integer input and adds it to the data stack. If the user input is not of the correct type it is ignored and the procedure will be repeated.
GETKY	F0	Scans keyboard & places the ASCII code of the key pressed on the data stack. Ø indicates no key pressed.

!B	F0	Takes 1 byte from 2nd and places in the address specified by TOP. Like a BASIC POKE

LOOP STRUCTURES

IF (ELSE) THEN	F6	IF (treatment when TOP<>0) ELSE (treatment when TOP=0) THEN e.g. 0 > IF "PLUS" ELSE "NOT PLUS" THEN S. when TOP is positive the word "PLUS" is displayed, otherwise "NOT PLUS" is displayed.
F.IF (ELSE) THEN	F6	IF (treatment when TOP=0) ELSE (treatment when TOP<>0) THEN e.g. F.IF "ZERO" S. THEN When TOP is zero, "ZERO" is displayed.
DO..LOOP	F6	Take the contents of TOP as the initial value, 2nd as the final value and repeatedly execute the section of code in between the words DO and LOOP, incrementing the counter on top of the return stack until such time as it is equal to the final value. This performs in a manner similar to the FOR-NEXT loop in BASIC.
BEGIN..END	F6	If TOP is 0 on reaching END jump back to BEGIN and repeat the section of code. e.g. BEGIN GETKY END will wait until the user presses a key.
I	F6	Used within a DO..LOOP section, this command copies the current value of the loop counter (which lies on top of the return stack) on to TOP. e.g. 10 0 DO I . LOOP if used within a Forth word will output 0123456789
J	F6	Also used within a Forth word, this command has the effect of copying the 2nd top of the return stack on to TOP. TOP will then contain the upper value of the loop section.

GRAPHICS COMMANDS

SETG	F0	sets the screen at position X=2nd Y=TOP e.g. 30 24 SETG is like SET 30,24 in BASIC.
RESG	F0	Resets the screen at X=2nd Y=TOP e.g. 30 24 RESG is like RESET 30,24 in BASIC.
LINE	F0	Draws a line between two points on the screen. e.g. 0 0 79 49 LINE draws a diagonal line between (0,0) and (79,49)
VRCL VRMV	F0 F0	These commands are designed to allow the animation of pictures. VRCL reserves an area of memory into which all subsequent SETG, RESG and LINE commands are directed. A picture can then be built up as though setting points on the screen. VRMV will then copy the entire picture on to the screen giving the impression of a quick change. After VRMV the graphics commands are restored.
LIMIT	F6	Changes bottom address of data and return stacks. 2nd gives the bottom address of the data stack, TOP gives the bottom address of the return stack. After LIMIT the data and return stacks become empty and the interpreter is initialised.

GRAPHICS COMMANDS

CORDR F0

Coordinate rotate - If:
3rd contains value X
2nd contains value Y
TOP contains value Z
then if (X,Y) are the coordinates of a point and Z is twice an angle theta, then the values returned after CORDR are called are :

TOP contains Y'
2nd contains X'

Where (X',Y') are the coordinates of that point rotated through theta degrees. The transformations themselves are:

X*COS(Z/2)-Y*SIN(Z/2)→2nd

X*SIN(Z/2)+Y*COS(z/2)→TOP

CORDV F0

Assuming X,Y,Z as before then if Z is zero and (X,Y) are the coordinates of a point then the values returned are the polar coordinates of the point expressed as (theta,R) where theta is the angle and R is the radius from the origin to that point.
The transformations themselves are:
X + Y → 2nd
Z+2*(TAN (Y/X))→TOP
N.B. X must not be negative
TOP contains 2 * theta.

STACK OPERATORS

For all the following operations the stack is assumed to be as per diagram initially:

1	TOP
2	2nd
3	3rd
4	4th
5	5th

STK F6

Writes out the contents of the stack without alteration.
e.g. STK outputs 1 2 3 4 5

CL F0

Clears the stack

DUP F0

Duplicates TOP
e.g. DUP STK gives 1 1 2 3 4 5

DDUP F0

Duplicates TOP and 2nd
e.g DDUP STK gives 1 2 1 2 3 4 5

OVER F0

Duplicates 2nd
e.g OVER STK gives 2 1 2 3 4 5

2OVER F0

Duplicates 3rd
e.g. 2OVER STK gives 3 1 2 3 4 5

3OVER F0

Duplicates 4th
e.g.3OVER STK gives 4 1 2 3 4 5

SWAP F0

Swaps TOP & 2nd
e.g. SWAP STK gives 2 1 3 4 5

DROP F0

Removes TOP
e.g. DROP STK gives 2 3 4 5

ROT F0

Transfers bottom to TOP
e.g. ROT STK gives 5 1 2 3 4

WORD	TYPE	USE AND EXPLANATION
SAVE	F6	Saves the Forth dictionary stack to tape using named files. e.g. SAVE KNIGHT DEMO
LOAD	F6	Load the next object file from tape. The filename is not checked.
VERI	F6	Verifies sucessful saving of a tape program. If OK then no message is displayed, otherwise the VERIFY ERROR is displayed.
REG	F2	Points to an area of memory where parameters for passing values to the Z80 registers can be located. The arrangement is: REG F A C B E D L H
USR	F6	Jump to machine code subroutine at address pointed to by TOP. On calling, AF, BC, DE and HL of REG are transferred across to the Z80 registers, on returning, the contents of the registers are transferred to REG. e.g. REG 1 + CONST AREG defines the address of the A register argument location. $30 AREG !B USR puts $30 into the A register argument location and calls the subroutine starting at address $12.
APLY	F6	Treats the character in quotes as a binary operator on TOP and 2nd e.g. 3 4 "+" APLY is equivalent to 3 4 +.
BASE	F2	Address in memory where the current base for mathematical calculations is stored. Only the lower byte of TOP is significant. e.g. $10 BASE ! puts the micro into BASE 16.
SP	F5	Address in memory where bottom address of return stack is located.
DS	F5	Address where bottom of the dictionary stack is located.
TXT	F6	Return to editor mode.

REVERSE POLISH NOTATION

Forth uses RPN for mathematical calculations. RPN is easy to learn. All numbers are stored on the data stack, when a binary operator (such as + or *) is encountered, Forth takes both operands off the TOP of the data stack, performs the calculation and then pushes the result back on TOP. Once learned, it is very easy to use and dispenses with the need for the parentheses that conventional infix notation requires.

In addition to the MZ-700 Forth language itself, we supply 10 programs to demonstrate some of the features of Forth programming.
To load the programs follow these steps.

1) Type TXT and press CR to get into editor mode.
2) Type & and press the CR key to clear the edit buffer.
3) Type L and press CR to load the program into memory.
4) Once loading has finished, type F and then press CR to compile the program. The text will list to the screen as compilation occurs. Some of the programs add new commands, others demonstrate Forth capabilities. Once compiled press ! and CR to enter run mode. Programs will start as soon as you enter the appropriate key word and press CR.

PROGRAMS SUPPLIED WITH KNIGHTS FORTH

EXTRA COMMANDS

Before running the rest of the sample programs supplied with Knights Forth, this program should be loaded and compiled. This adds a whole host of extra commands to those already present on the Forth language tape. These are the extra commands:

?REG displays the contents of the Z-80 registers in both decimal and hexadecimal.

<= less than or equal to test.

DISCOMPILE shows the defined words in the dictionary stack together with their storage addresses and definitions. DISCOMPILE starts with the most recently defined word and works downwards.

PAINT fills the screen with foreground and background colours as selected by a single byte in location COLOR.

C. performs in a manner similar to S. except that the text is printed in the colours selected by COLOR.

HEX puts Forth into base 16.

DECI puts Forth into base 10.

CHM clears the screen.

ABS returns the absolute value of TOP.

TXT/P prints the contents of the text buffer on the printer.

S./P prints a string to the printer.

SP. prints a space to the printer.

P26 puts the plotter/printer in 26 characters per line mode.

P40 puts the plotter/printer in 40 characters per line mode.

P80 puts the plotter/printer in 80 characters per line mode.

PBS prints a back space on the plotter/printer.

PFF prints a form feed on the plotter/printer.

PHOME moves the paper 66 lines upwards on the plotter/printer.

CUBE DEMO is a high speed demonstration which shows a rotating cube. The keyword KNIGHT starts the program.

WHEEL demonstrates the speed and efficiency of Forth. The word WHEEL starts the demonstration.

FORTH BRIDGE adds these extra commands to Forth.

LIST which lists all the words currently in the Forth dictionary stack. This is similar to DISCOMPILE, however only the names and types of the definitions are output.

BYE returns the MZ-700 to monitor level.

PROGRAMS SUPPLIED WITH KNIGHTS FORTH

FORTH BRIDGE continued

BON turns on the keyboard bell.

BOFF turns off the keyboard bell.

MUSIC will interpret the string on top of the stack as music data. The music stops when a ■ character (graph + SHIFT + F) is reached.

RANN :- entering the command n RANN will generate a random integer between 0 and n-1.

FORTH SYMPHONY demonstrates how music can be played using Forth. Entering the keyword RUN lets you hear the notes.

COLOUR CHANGER shows a string and changes the colour 100 times. Written simply so that you can list it and see how the effects were obtained. The keyword is DEMO.

THE RELAXER fills the screen with random colours and smoothly changes them. The Forth word RELAXER starts the program.

BIORHYTHM is a Forth program to calculate your average biorhythm for an 80 day period. The keyword RUN starts the program.

BLOCK GRAPHICS adds 3 new commands to Forth.
The format is: P1 P2 P3 P4 WORD BLOCK where:

P1..P4 are the coordinates of the 2 points diagonally furthest apart on the block to be filled in,

WORD can be:

SET indicating that the block is to be filled in with set points

RES indicating that the rectangle is to be made up of reset points

REV indicating that set points within the rectangle are to be reset and vice versa.

e.g. 0 0 79 49 SET BLOCK will fill the entire screen with set points.

NUMBER ADDER adds 2 numbers together in Forth. List this program and learn how to use a stack based language.

STARTING FORTH PROGRAMMING

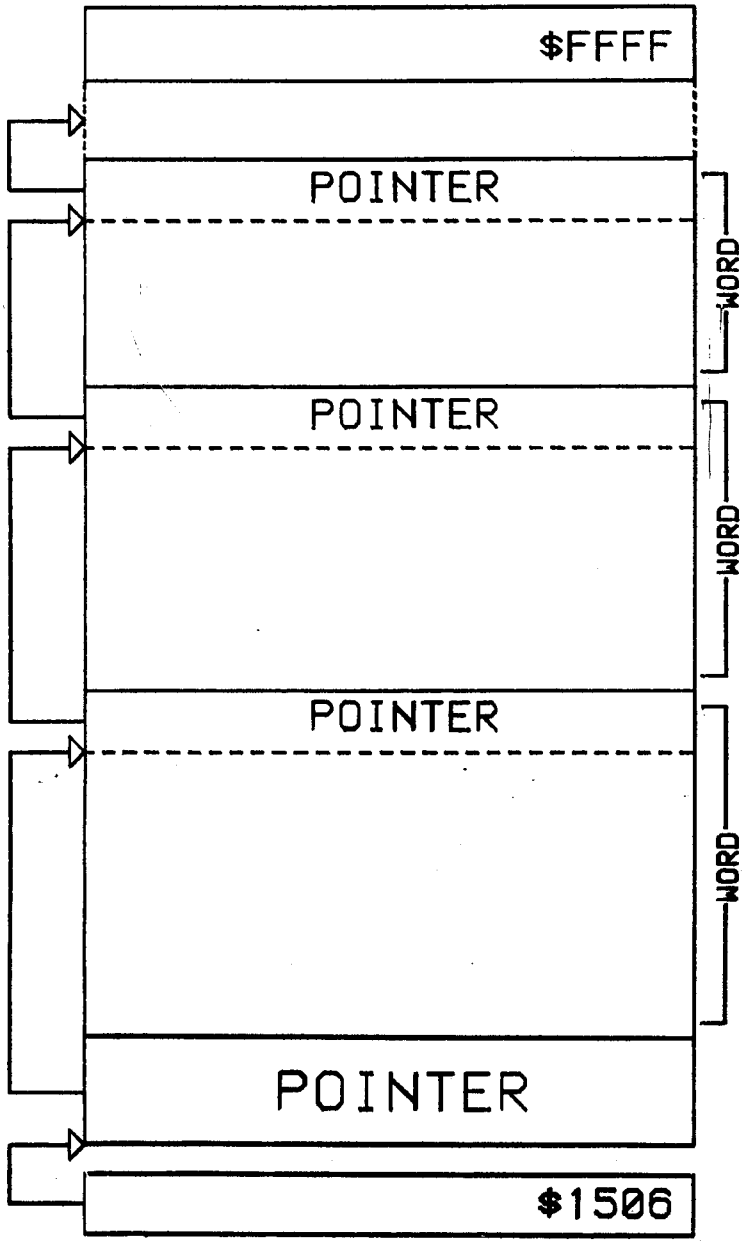
As you will appreciate from these instructions Forth is a very complex language and beginners should start by examing the listings of the sample programs on this tape.

Start programming by making a small change to our sample - try putting your name on the screen instead of Knights in the Cube Demo. Once you have done this successfully proceed to changing the keyword. Then experiment with further small changes, make the cube rotate slowly, change the graphics and colours.

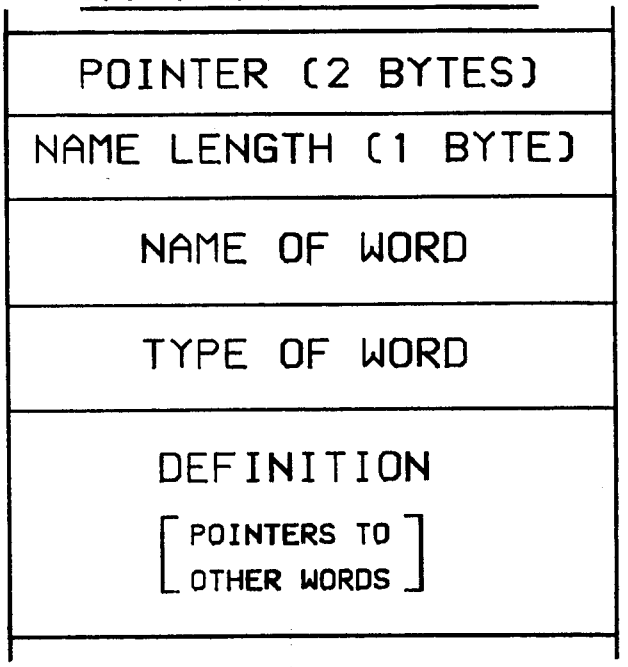
STRUCTURE OF KNIGHTS MZ-700 FORTH LANGUAGE

The Knights Forth Language structure diagrams below were printed and programmed on an MZ-700 and a printer/plotter.

STRUCTURE OF A FORTH DICTIONARY



STRUCTURE OF A FORTH WORD



STRUCTURE OF A FORTH DATA STACK