DEC's FOCAL 1969 Promotional Booklet

Part of the FOCAL Collection
Douglas W. Jones
University of Iowa Department of Computer Science

An index has been appended at the end of this document, along with notes on the transcription.

---

focal

a new conversational language
developed by Digital Equipment
Corporation
for its PDP-8 family of small computers

digital equipment corporation

Part 1

Computers have become very important in our lives. They watch over bank accounts, select the springs for our automobiles, and give us brighter color television. They control space ships, launch and control guided missiles and antimissile missiles, and aid in engineering design. Other computers are used to control machine tools and chemical processes, control inventories, and make out payrolls. Even our income tax returns are checked by computers. Further uses for the computer include analysis of biomedical data and the examination of business, social and historical information. We hear every day about some new feats performed by computers, and we tend to feel that the computer is an impersonal machine, destined to rule our lives. In reality, we control the computer. Anyone can learn to use the computer.

Anyone can learn to write computer programs in a few hours.

There are several computer facts about which everyone should be aware:

- THE COMPUTER IS STUPID. It cannot think. It can perform only a limited number of basic functions, and it must be told to do something before it can even begin to act.
- THE COMPUTER IS FAST. It performs any of its limited functions in millionths of seconds.
- THE COMPUTER IS ACCURATE. There is no need to perform calculations on the computer if you want answers that are only nearly correct. Accuracy is the computer's strong suit.
- THE COMPUTER IS A PERFECTIONIST. It handles its limited functions accurately and precisely every time it is told to act. The computer does things perfectly because it cannot do them incorrectly. Programmed correctly, the computer performs perfectly.

Using a computer is very much like using a telephone. By itself, the telephone is stupid. It cannot select numbers until we instruct it. The computer also must wait for instructions before it can begin to work. And it takes us little more time to learn to operate a computer than it took us to learn to use a dial telephone.

With FOCAL, Digital's easy-to-learn conversational programming language, all that is needed is a little time and a little effort. Within a few minutes, you can program the computer to solve common problems.

1

Now let's program the computer to work for us. We will begin with something that everyone understands -- calculating simple interest.

The interest formula that we are programming is

Interest =Principal * Rate * Time.

(In FOCAL, the asterisk * is used as a symbol for multiplication.) We will transform this formula into a set of instructions (program) which we will enter into the computer on the Teletype.

Let's pause right here and reflect on how we as humans would make the calculation:

> First, we would decide how much money we wanted to borrow.
>
> Next we would decide for how many years we wanted to borrow the money.
>
> We would then determine what the current interest rate is.
>
> Finally, we put pencil to paper and make a calculation.
>
> We would then probably add the interest charge to the amount we wanted to borrow, compare the cost of borrowing the money to the money we wanted to borrow, and then decide if we were going to borrow.

The computer arrives at its solution in exactly the same way.

One way of writing the program for calculating interest would be:

```
 01.10 SET PRINCIPAL=100
 01.20 SET TERM=5
 01.30 SET RATE=.05
 01.40 SET INTEREST=PRINCIPAL*RATE*TERM
 01.50 TYPE "INTEREST",INTEREST,!
 01.60 SET VALUE=PRINCIPAL+INTEREST
 01.70 TYPE "TOTAL VALUE",VALUE,!
*
```

```
 GO
INTEREST=    25.0000
TOTAL VALUE=   125.0000
*
```

In this computer program, we have given the computer a number of specific tasks to perform and specified the order or sequence in which we want them performed. When we number a line (possible choices are 1.01; 1.02; ....31.99), it becomes a part of a stored program. Therefore, whe3n you want to store a program within the computer's memory, give each line a number. Otherwise, the computer operates as a calculator and executes the statement immediately.

2

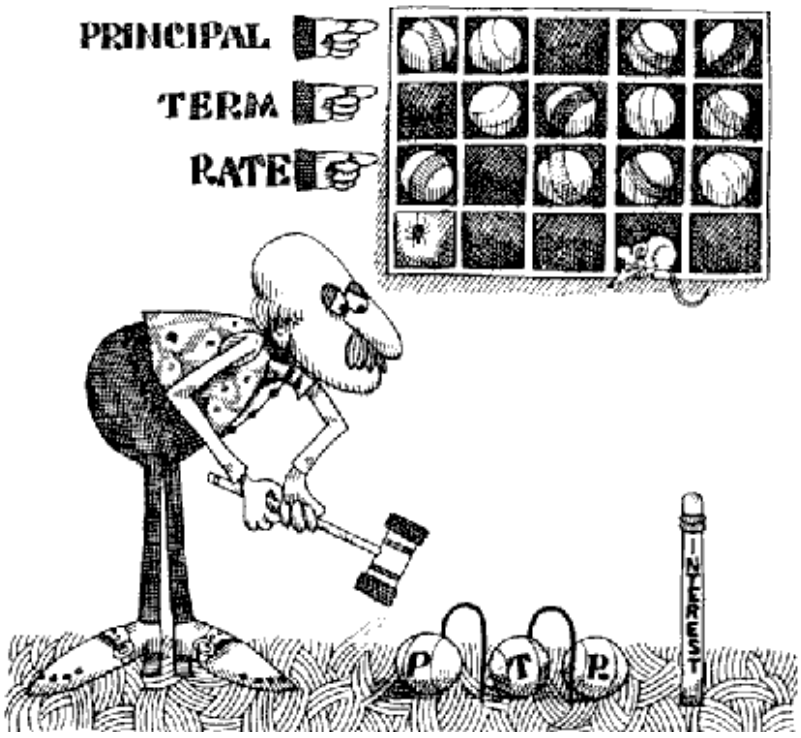
Line 1.1 is the command to set the value of the Principal to equal $100.

Line 1.2 commands the computer to set the term of the loan to equal 5 years.

Line 1.3 commands the computer to set the rate of interest to 5% (.05).

Line 1.4 is the command to calculate the interest and store it in memory. We as humans calculate interest by multiplying the Principal times the rate of interest times the term of the loan. The computer makes the calculation in *exactly* the same way.

Line 1.5 is the command to print the computed value of interest. In Line 1.5, we have given the computer a series of *three* tasks to perform. First -- type the word INTEREST (the characters enclosed within quotation marks are copied). Then type the computed interest value. Finally, return the type-



3

---

writer carriage and move the paper up one space (! is the computer command to return the typewriter carriage to the left margin and move up one space).

Line 1.6 commands the computer to set the VALUE equal to PRINCIPAL plus the computed value of INTEREST and store the VALUE in memory.

Line 1.7 commands the computer to type the words TOTAL VALUE, then type the calculated value (Principal + Interest), do a carriage return and move the paper up one space.

The computer is most useful when we have a great number of repetitious tasks to do -- such as calculating interest charges a great number of times a day. A bank teller gets very, very tired and bored answering the same question over and over again day after day. People constantly ask the teller, "How much will it cost me to borrow X dollars for Y years at 5%?"

Let's program the computer to answer these questions so the teller can tend to more important tasks.

A program to do this looks as follows:

```
 01.01 ASK "HOW MUCH MONEY DO YOU WANT TO BORROW ?",PRINCIPAL
 01.20 ASK "FOR HOW MANY YEARS ?",TERM
 01.30 TYPE "ENTER INTEREST RATE IN DECIMAL NOTATION.  THAT IS,",!
 01.40 TYPE "6.5 FOR 6 1/2 %,8.75 for 8 3/4 %,ETC..",!!
 01.50 ASK "WHAT INTEREST RATE ?",RATE
 01.60 SET INTEREST=PRINCIPAL*(RATE/100)*TERM
 01.70 TYPE "THE INTEREST ON",PRINCIPAL," DOLLARS BORROWED"
 01.80 TYPE " FOR",TERM," YEARS",!,"IS",INTEREST, " DOLLARS.",!!
 01.90 TYPE "TOTAL VALUE OF PRINCIPAL + INTEREST IS",PRINCIPAL+INTER,!
*
 02.10 TYPE "IT IS UNDERSTOOD,OF COURSE,THAT THIS IS SIMPLE INTEREST."
 02.20 TYPE !!!!!
 02.30 GOTO 1.1
```

Line 2.3 begins the program again. This program is an "endless loop" because we haven't told the computer when to quit and it would run forever (or until someone unplugged it!) asking the same questions over and over.

```
 GO
HOW MUCH MONEY DO YOU WANT TO BORROW ?:100
FOR HOW MANY YEARS ?:5
ENTER INTEREST RATE IN DECIMAL NOTATION.  THAT IS,
6.5 FOR 6 1/2 %,8.75 FOR 8 3/4 %,ETC..

WHAT INTEREST RATE ?:5.0
THE INTEREST ON=  100.0000 DOLLARS BORROWED FOR=    5.0000 YEARS
IS=   25.0000 DOLLARS.

TOTAL VALUE OF PRINCIPAL + INTEREST IS=  125.0000
IT IS UNDERSTOOD,OF COURSE,THAT THIS IS SIMPLE INTEREST.
```

4

```
HOW MUCH MONEY DO YOU WANT TO BORROW ?:550
FOR HOW MANY YEARS ?:10
ENTER INTEREST RATE IN DECIMAL NOTATION.  THAT IS,
6.5 FOR 6 1/2 %,8.75 FOR 8 3/4 %,ETC..

WHAT INTEREST RATE ?:7.75
THE INTEREST ON=  550.0000 DOLLARS BORROWED FOR=   10.0000 YEARS
```

```
IS=  426.2500 DOLLARS.

TOTAL VALUE OF PRINCIPAL + INTEREST IS=  976.2500
IT IS UNDERSTOOD,OF COURSE,THAT THIS IS SIMPLE INTEREST.
```

- The ASK statement allows a human to input data into the comptuer. Information in quotation marks is not operated on by the computer but is "echoed" by the computer to the human as a message (Lines 1.1; 1.2;1.5)
- The exclamation mark (!) is interpreted by the computer as "return the typewriter to the left margin of the paper and then move the paper up one space." The exclamation mark is very useful in formatting output from the computer. (Lines 1.3; 1.4; 1.8; 1.9; and 2.2)
- The TYPE statement causes the computer to output information. (Lines 1.3; 1.4;1.7;1.8;1.9; 2.1; and 2.2)
- The SET statement commands the computer to make a computation and retain the computed value as the symbol to the left of the equals sign. (Line 1.6)
- The GOTO statement directs the computer to go to a specific place and begin executing statements at that point. (Line 2.3)

Line 1.1 asks the question "How much money do you want to borrow?" The computer then stores the number the user types in, labelling that number PRINCIPAL.

Line 1.2 asks the question, "For how many years?" The value of the number given here is stored as the TERM.

Line 1.3 and 1.4 type a message to the user, telling him how to enter the INTEREST RATE.

Line 1.5 asks, "What interest rate?" The value given here is stored as the RATE.

The computer has all the facts it needs to calculate the INTEREST, based on the information that has been put in by a human.

Line 1.6 makes the calculation to determine the INTEREST. Note: we have divided the rate in decimal notation by 100 to give us the interest rate. (Remember, 6% is really .06 or 6/100). If we had not divided by 100, we

would have charged the customer 100 times too much money.

Lines 1.7 and 1.8 tell the computer to type out a message that we have included in the program. All words and characters inside the quotation marks are printed out as a message. When we want to print out a calculated value, we specify what value we want and then continue with the message. Notice in line 1.7 that we have printed the stored value of PRINCIPAL and in Line 1.8 we have printed the stored values of TERM and INTEREST.

5

Line 1.9 tells the computer to type out a message telling the value of principal plus interest and then type out the computed value of principal plus interest.

Line 2.1 is a message to the customer, telling him what method we used to compute the interest rate.

Line 2.2 instructs the computer to move the paper in the typewriter up five spaces. (The exclamation mark is a symbol telling the computer to return the Teletype carriage and move the paper up one space.)

Line 2.3 restarts the sequence of events that we have programmed into the computer. This program is a form of endless loop because we haven't told the computer when to stop.

This is only one example of how the computer could be used to perform a repetitious task.

But you say, "Boy, that sure is an impersonal way to run a bank. I'd never trade there because they're too impersonal". Remember this is only an example to explain how the computer works or can be made to work. I'll agree, I'd trade with a bank which had a nice looking girl or boy to answer my questions; unless, of course, I had to stand in line to get my question answered! Then I'd be very anxious to ask a computer so I could get my answer and be on my way.

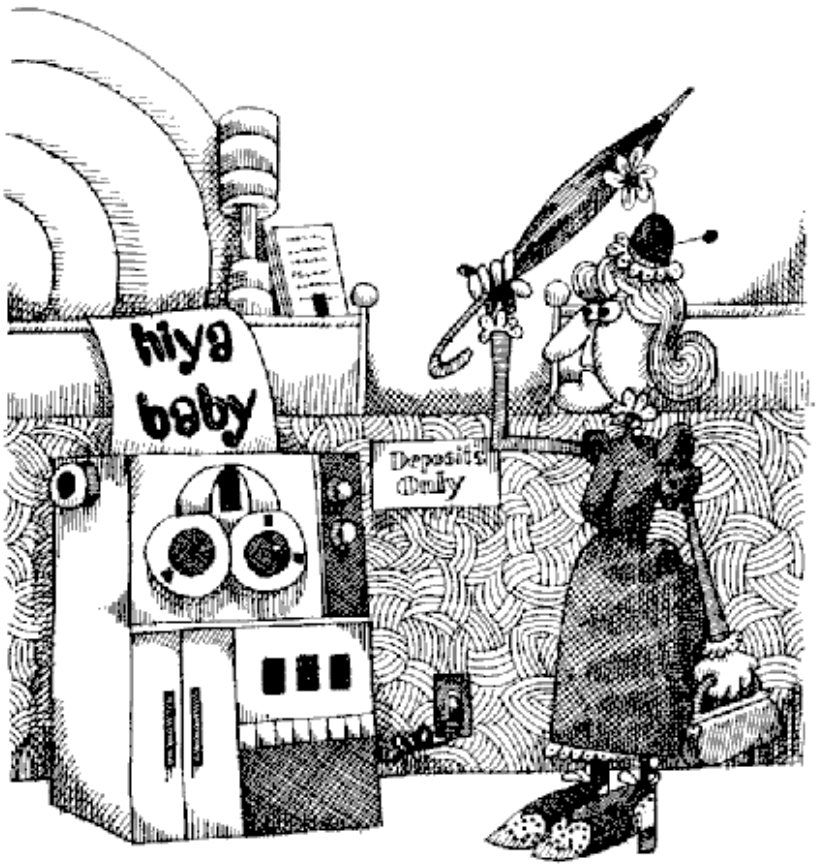
Let's write a computer program which is quite personal in nature and see how the computer *could* be made to be personal.

This time let's run the program before we explain it:

```
HI THERE, GOOD LOOKING.  HOW MUCH MONEY DO YOU WANT
TO BORROW ?:300
THANK YOU DEAR.  HOW LONG DO YOU WANT TO BORROW THE
MONEY FOR ?:2

SWEETS, THE GOING RATE OF INTEREST IS 8.5%.  IT WILL
COST YOU=   51.0000 DOLLARS TO BORROW=  300.0000 DOLLARS
FOR =    2.0000 YEARS.  YOU DO UNDERSTAND, OF COURSE,
THAT THIS IS SIMPLE INTEREST.

STEP RIGHT UP TO OUR TELLER AND HE WILL BE GLAD TO
HELP YOU.

NICE TALKING WITH YOU.  DO STOP IN AGAIN.  BYE,BYE, NOW.
```

Now I ask, just how much more personal can you get? One problem though; the computer doesn't know who it is working for -- male or female; young or old; pleasant or grouchy; single or married. How would you like a computer flirting with your grandmother? Mabe it really is best to keep the computer impersonal!!

6

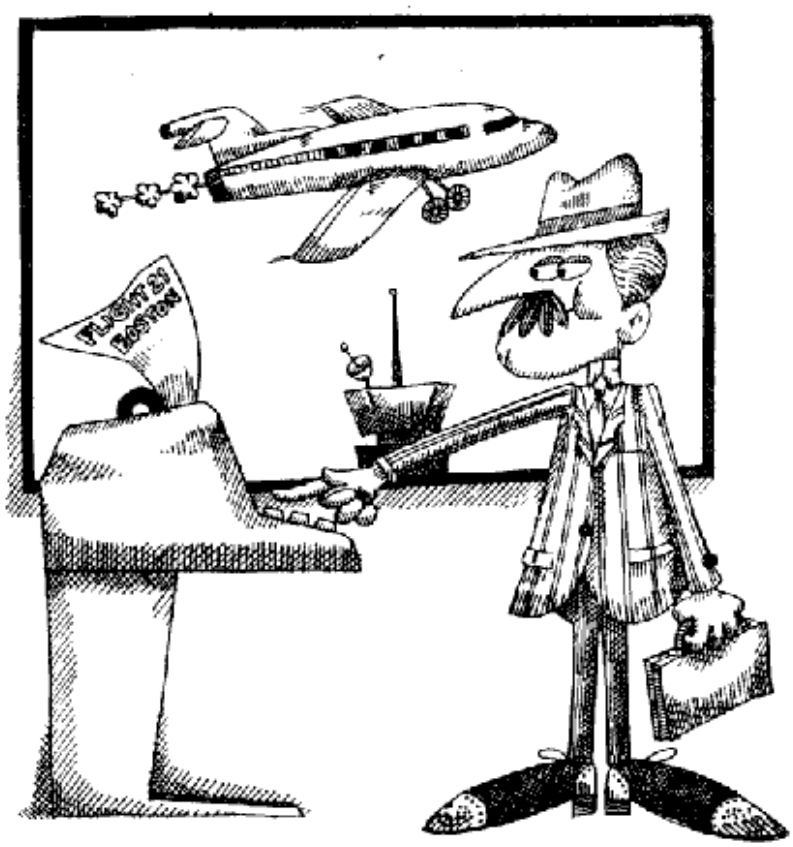


Here is how the program looks:

```
01.05 SET RATE=.085
01.10 TYPE !!!!!,"HI THERE, GOOD LOOKING.  HOW MUCH MONEY DO YOU WANT",!
01.20 TYPE "TO BORROW ?";ASK PRINCIPAL
01.30 TYPE "THANK YOU DEAR.  HOW LONG DO YOU WANT TO BORROW THE ",!
01.40 TYPE "MONEY FOR ?";ASK TERM
01.50 SET INTEREST=PRINCIPAL*RATE*TERM
01.60 TYPE ,!,"SWEETS, THE GOING RATE OF INTEREST IS 8.5%.  IT WILL",!
01.70 TYPE "COST YOU",INTEREST," DOLLARS TO BORROW",PRINCIPAL
01.80 TYPE " DOLLARS",!,"FOR ",TERM," YEARS.  YOU DO UNDERSTAND"
01.85 TYPE ", OF COURSE,",!,"THAT THIS IS SIMPLE INTEREST.",!!
01.90 TYPE "STEP RIGHT UP TO OUR TELLER AND HE WILL BE GLAD TO",!

02.10 TYPE "HELP YOU.",!!
02.20 TYPE "NICE TALKING WITH YOU.  DO STOP IN AGAIN.  BYE,BYE,NOW.",!
02.30 GOTO 1.1
*
```

7

So we see how the computer can be programmed to perform repetitive tasks. It can even repetitively answer questions it is programmed to answer. Many of us have an adversion to asking computer questions. We'd much rather talk to humans. But is this always true?



Let's say your are in a busy airport and you just missed your connecting flight. The lines look a mile long and you know some airline has a flight leaving in ten minutes. You'd then be very willing to step up to an impersonal typewriter, answer its questions about where you wanted to go. You'd be very grateful when the computer told you it had four more seats left on a plane going where you wanted to go. It could even tell you how much the ticket costs so you could have the money ready when you boarded the plane. As

---

you boarded the plane, the stewardess would notify the computer to subtract one from the seats remaining so that the next person questioning the computer would have the very latest and accurate information.

We would then be very appreciative of the computer and could care less if it answered our questions impersonally. It did its job and helped us get home a few hours early.

A program to give us information on airline departures looks like this:

```
01.10 TYPE "I AM PROGRAMMED TO GIVE FLIGHT INFORMATION FOR THE ",!
01.20 TYPE "FOLLOWING CITIES:",!!
```

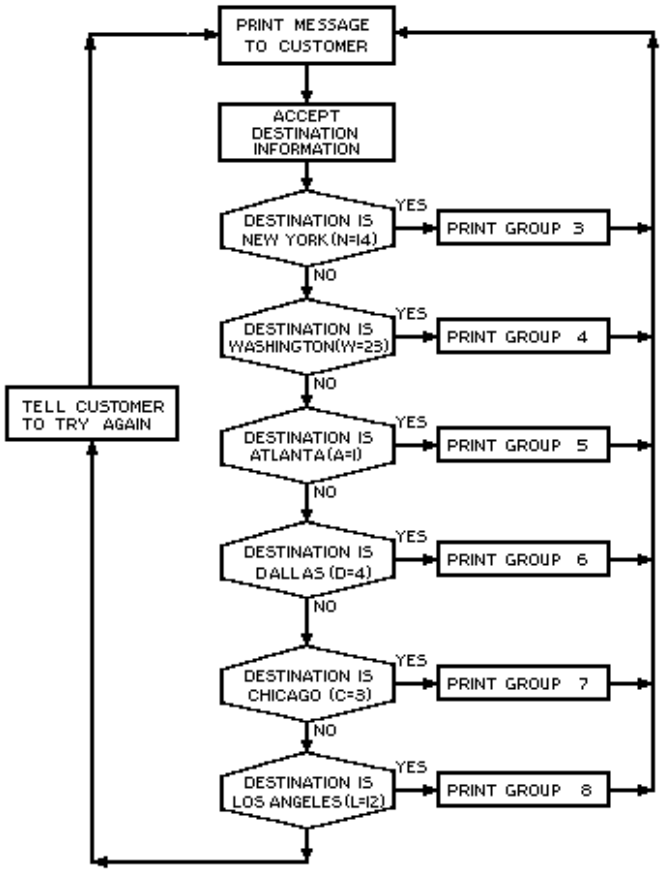
```
01.30 TYPE "NEW YORK",!,"WASHINGTON",!,"ATLANTA",!,"DALLAS",!
01.40 TYPE "CHICAGO",!,"LOS ANGELES",!!
01.50 TYPE "WHAT IS YOUR DESTINATION ?(TYPE FIRST LETTER OF CITY.  "
01.60 TYPE "THAT IS D FOR",!,"DALLAS, N FOR NEW YORK AND ETC..)",!
01.70 TYPE "PRESS THE SPACE BAR AFTER YOU IDENTIFY YOUR DESTINATION.",!!
01.80 ASK DESTINATION

02.30 IF (DES-14) 2.4,3.1,2.4
02.40 IF (DES-23) 2.5,4.05,2.5
02.50 IF (DES-1) 2.6,5.1,2.6
02.60 IF (DES-4) 2.7,6.1,2.7
02.70 IF (DES-3) 2.8,7.1,2.8
02.80 IF (DES-12) 2.9,8.1,2.9
02.90 TYPE !!,"TRY AGAIN.",!!!!!;GOTO 1.1

03.10 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
03.20 TYPE " 7:50 A.M.       25 B        NORTHEASTERN   121",!
03.22 TYPE " 9:15 A.M.       13 B        ALLEGHNY       111",!
03.24 TYPE "12:50 P.M.       19 B        EASTERN         91",!
03.26 TYPE " 4:50P.M.        17 A        AMERICAN        53",!
03.29 TYPE "11.50 P.M.       21 A        NORTHEASTERN   117",!
03.30 TYPE !!!!!;GOTO 1.1

04.05 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
04.10 TYPE " 9:30 A.M.        5 A        TWA             54",!
04.20 TYPE "12:30 P.M.       21 B        ALLEGHNY        23",!
04.21 TYPE " 3:35 P.M.       15 B        EASTERN         15",!
04.22 TYPE " 9:50 P.M.       12 A        EASTERN         91",!
04.23 TYPE !!!!!;GOTO 1.1

05.10 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
05.20 TYPE " 7:30 A.M.        3 B        AMERICAN        50",!
05.30 TYPE "11:55 A.M.       13 A        ALLEGHNY        17",!
05.40 TYPE " 5:40 P.M.       21 B        TWA            131",!
05.50 TYPE !!!!!;GOTO 1.1

06.10 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
06.20 TYPE " 7:30 A.M.       28 A        UNITED         101",!
06.30 TYPE "10:50 A.M.       15 B        AMERICAN        51",!
06.40 TYPE " 2:45 P.M.        5 A        TWA             93",!
06.50 TYPE " 6:50 P.M.       21 B        AMERICAN       121",!
06.60 TYPE "11:55 P.M.        3 A        UNITED          63",!
06.70 TYPE !!!!!;GOTO 1.1

07.10 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
07.20 TYPE " 7:55 A.M.       15 B        UNITED          53",!
07.30 TYPE "11:35 A.M.       25 A        DELTA           17",!
07.40 TYPE " 2:15 P.M.       13 B        TWA            161",!
07.50 TYPE " 5:30 P.M.        7 A        UNITED          53",!
```

```
07.60 TYPE " 7:30 P.M.       16 B        AMERICAN        99",!
07.70 TYPE "10:50 P.M.       25 A        DELTA           59",!
07.80 TYPE "11:50 P.M.        8 B        TWA            121",!
07.90 TYPE !!!!!;GOTO 1.1
```

9

```
08.10 TYPE !!,"DEPARTURE        GATE        AIRLINE        FLIGHT #",!
08.40 TYPE " 9:15 A.M.        5 B        TWA            161",!
08.50 TYPE "11:30 A.M.       17 A        UNITED          53",!
08.60 TYPE " 2:50 P.M.        3 A        AMERICAN       111",!
08.60 TYPE " 5:30 P.M.       20 B        UNITED          61",!
08.40 TYPE " 9:40 P.M.        3 A        TWA            173",!
08.60 TYPE "11:55 P.M.       19 A        AMERICAN       121",!
08.90 TYPE !!!!!;GOTO 1.1
```

A flow chart of the previous program looks like this:



10

When a customer types in a letter to declare his destination, the computer converts the letter to its numerical equivalent (A = 1, B = 2,...Z = 26) and then

checks to see if the letter corresponds to one of the cities for which it has information. If the computer has stored information on Los Angeles, for example, it goes to line 8.1 and begins printing out the stored information. After it has finisted printing this information, the computer goes to line 1.1 and begins printing a message to the next customer.

Notice that group 2.0 is written so that the program "ripples" through group 2.0, looking for the letter the customer typed in. If it doesn't find the letter, it types "TRY AGAIN" and restarts at line 1.1.

As another example, let's say you drove into Boston or New York City dead tired; all the major hotels and motels are full. You could drive all over town looking for a place to stay and personally talk with all the desk attendants. After several hours of search, I'm sure you'd be very receptive to letting an impersonal computer find you a place to stay in a matter of seconds!

Some people become very aggravated when a computerized bookkeeping system malfunctions and bills them incorrectly. They would be much more aggravated, however, if they were billed several months late, or had to stand in line hours to find out what they owed the gas company, the telephone company, the oil company, the city tax collector, etc. In a word, computers are only as accurate and impersonal as they are programmed to be.

Now that we know how the computer can be programmed to do different things, let's look again at our program for calculating interest.

Let's say we want to see what happens to the interest on the loan as the rate of interest changes by .5% from 4% to 10%.

This would take us many minutes to calculate, so instead, let's write a program and let the computer do the number crunching.

```
01.10 ASK "HOW MUCH MONEY DO YOU WANT TO BORROW ?",PRINCIPAL
01.20 ASK "FOR HOW MANY YEARS ?",TERM
01.30 FOR RATE=4.0,.5,10;DO 2.0
01.40 QUIT

02.10 SET INTEREST=PRINCIPAL*(RATE/100)*TERM
02.20 TYPE "RATE",RATE,"   ","INTEREST",INTEREST,!
*
 GO
HOW MUCH MONEY DO YOU WANT TO BORROW ?:100
FOR HOW MANY YEARS ?:5
RATE=    4.0000   INTEREST=   20.0000
```

```
RATE=    4.5000   INTEREST=   22.5000
RATE=    5.0000   INTEREST=   25.0000
RATE=    5.5000   INTEREST=   27.5000
RATE=    6.0000   INTEREST=   30.0000
RATE=    6.5000   INTEREST=   32.5000
RATE=    7.0000   INTEREST=   35.0000
RATE=    7.5000   INTEREST=   37.5000
RATE=    8.0000   INTEREST=   40.0000
RATE=    8.5000   INTEREST=   42.5000
RATE=    9.0000   INTEREST=   45.0000
RATE=    9.5000   INTEREST=   47.5000
RATE=   10.0000   INTEREST=   50.0000
*
```

11

The computer is at its best when it has a great number of repetitive tasks to perform. In this example, the repetitive statement is the FOR command.

The FOR command has the following format:

FOR Variable = INITIAL VALUE, CHANGE BY, FINAL VALUE; DO SOMETHING

Lines 1.1 & 1.2 ask questions to the user

Line 1.3 Commands the computer to start at an initial interest rate of 4% and do everyting beginning with a 2 before changing the interest rate by 1/2 percent. Control will continue to be transferred between line 1.3 and the group 2.0 until the interest rate reaches 10%.

Line 1.4 commands the computer to halt.

Up to this point we have given you an insight into how the computer does some of the "magic" things that it does. Hopefully at this point you realize that the computer is only as "smart" as the person giving the instructions (programming).

The next section of this booklet is for the person who knows just a little bit of mathematics. If you don't already know a little math but want to continue learning more about computer programming, stick with us and we'll teach you more. Chances are we'll also teach you a little mathematics in the process!

12

Part 2

The computer is at its best when it works with a formula that requires a great many repetitious calculations. The following programs illustrate FOCAL's general capabilities.

Stated in English, the problem we will solve is:

> Generate a table of values of X, X^2 and $\sqrt{X}$ for all values of X from 0 to 100 in increments of 10.
>
> Graphically, we wish to accomplish the following:

X = 0	X^2 = 0	$\sqrt{X}$ = 0
.	.	.
.	.	.
.	.	.
X = 100	X^2 = 10,000	$\sqrt{X}$ = 10

> To generate this table using FOCAL,

```
01.10 FOR X=0,10,100;TYPE "X",X,"   ",X^2",X^2,"   ","SQRT",FSQT(X);
*
```

Don't get excited. A detailed explanation follows.

FOR is a FOCAL command. It means:

> FOR(VARIABLE)=INITIAL VALUE, INCREMENTED BY, FINAL VALUE; DO SOMETHING
>
> The TYPE statement is used to make computations, to format computer output, and to output information from the computer.
>
> In the example above, the type command tells the computer to:

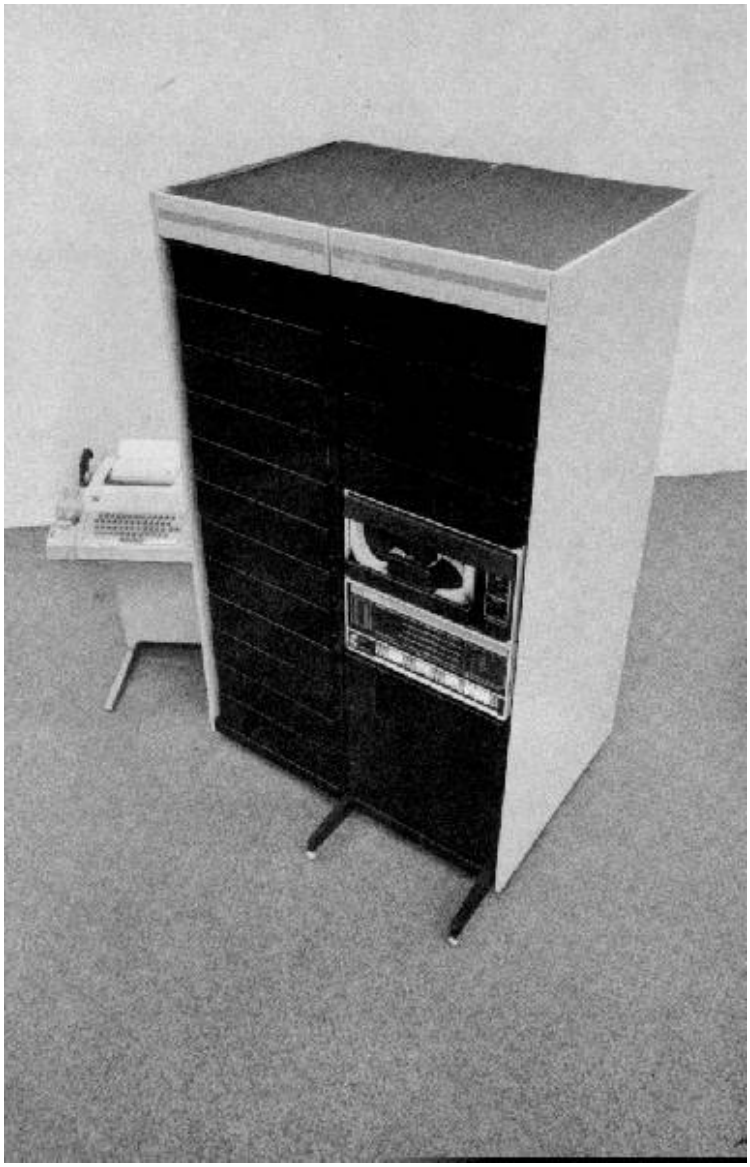
- Type the character X ("X")
- Type the value of X (X)
- Type 5 spaces (spaces are non " " printing characters
- Type the characters X^2 ("X^2")
- Compute X^2 and outputs the value (X^2)
- Type 5 spaces (" ")
- Type the characters SQRT ("SQRT")
- Compute $\sqrt{X}$ and output the value (FSQT(X))
- Generate a carriage return and line feed to the typewriter (!)

> A series of commands was given in the TYPE command as explained above. Each term of the series is separated by a comma (,).
>
> The computer interprets an exclamation mark (!) as "Return the Carriage to left margin and move paper up one space"

Let's run the program!

To cause the computer to execute the program we give the command "GO".

13



PDP-8/I -- a full scale digital computer with an internal peripheral control and data break panel for plug-in expansion.

14

```
X=     0.0000    X^2=     0.0000    SQ. RT.=     0.0000
X=    10.0000    X^2=   100.0000    SQ. RT.=     3.1623
X=    20.0000    X^2=   400.0000    SQ. RT.=     4.4721
X=    30.0000    X^2=   900.0000    SQ. RT.=     5.4772
X=    40.0000    X^2=  1600.0000    SQ. RT.=     6.3246
X=    50.0000    X^2=  2500.0000    SQ. RT.=     7.0711
X=    60.0000    X^2=  3600.0000    SQ. RT.=     7.7460
X=    70.0000    X^2=  4900.0000    SQ. RT.=     8.3666
X=    80.0000    X^2=  6400.0000    SQ. RT.=     8.9443
X=    90.0000    X^2=  8100.0000    SQ. RT.=     9.4868
X=   100.0000    X^2= 10000.000     SQ. RT.=    10.0000
```

Computer programming with FOCAL is really that easy!!

Another type of problem we might want to solve, stated in English, is:

Find the intersection of the two functions:

$Y_1 = -X^2 + 4X + 3$

$Y_2 = -X - 3$

Traditionally, one method of solution would be to set $Y_1 = Y_2$ and solve the resulting expression for values of X.

A FOCAL program is as follows:

```
01.10 FOR X=-10,1,10;DO 2.0
01.20 QUIT

02.10 SET Y1=-X^2+4*X+3;SET Y2=-X-3
02.20 IF (Y2-Y1) 2.4,2.3,2.4
02.30 TYPE "A POINT OF INTERSECTION IS ","X",X,"   ","Y1=Y2",Y1,!!
02.40 RETURN
*
```

This program gives the computer a series of tasks to perform:

- Start with X= -10 and do everything in group 2 before changing the value of X.
- Statement No. 2.1 says "compute the numerical value (for X= -10) of $-X^2 + 4X + 3$ and set Y_1 equal to that value. Compute the numerical value (for X= -10) of -X -3 and set Y equal to that value."
- Statement 2.2 says,"IF $Y_2 - Y_1$ is less than 0, execute Statement 2.4 next;"IF $Y_2 - Y_1$ is equal to 0, execute

Statement 2.3 next;"IF Y_2 -Y_1 is greater than 0, execute Statement 2.4 next".

The IF statement has the format:

IF (variable)<,=,>

15

The IF statement compares the value of the variable to zero and decides which statement to execute next.

When Y_2 = Y_1, we want the computer to print a message to the user and give him information relating to the values of X,Y_1, and Y_2 at that point of intersection. Statement 2.3 commands the computer to output this information.

Statement 2.4 commands the computer to return to Statement 1.1, increment the value of X by 1; 1.1 then transfers control to group 2 again. This transfer of control back and forth continues until X = 10 and then statement 1.2 is executed.

Statement 1.2 causes the program to quit and returns control to the user.

Let's run the program!

```
GO
A POINT OF INTERSECTION IS X=-   1.0000   Y1=Y2=-    2.0000

A POINT OF INTERSECTION IS X=    6.0000   Y1=Y2=-    9.0000

*
```

If we were interested in the values of Y_1 and Y_2 for all values of X we could restate the problem:

"Compute and print Y_1 and Y_2 for all values of X and identify the values of Y_1, Y_2 and X at the point of intersection."

We can very easily modify the previous FOCAL program to do this:

```
*
 01.10 FOR X=-10,1,10;DO 2.0
 01.20 QUIT

 02.10 SET Y1=-X^2+4*X+3;SET Y2=-X-3
 02.20 IF (Y2-Y1) 2.4,2.3,2.4
 02.30 TYPE !,"A POINT OF INTERSECTION IS ","X",X,"   ","Y1=Y2",Y1,!!
 02.40 TYPE "X",X,"     ","Y1",Y1,"   ","Y2",Y2,!
 02.41 RETURN
*
```

This program gives the computer a slightly different series of tasks to perform:

- Start with X= - 10 and do everything in group 2 before changing the value of X.
- Statement No. 2.1 savs "compute the numerical value (for X= -10) of $-X^2 + 4X + 3$ and set Y_1 equal to that value. Compute the numerical value of (for X= -10) in -X -3 and set Y_2 equal to that value."
- Statement 2.2 says,"IF $Y_2 - Y_1$ is less than 0, execute Statement 2.4 next;"IF $Y_2 - Y_1$ is equal to 0, execute Statement 2.3 next;"IF $Y_2 - Y_1$ is greater than 0, execute Statement 2.4 next".

16

- When ($Y_2 - Y_1$) is something other than 0, control is transferred to Statement 2.4 and the computer types out the values of X, Y_1 and Y_2.
- Statement 2.41 returns control to Statement 1.1 and the value of X is incremented by 1 and control is again returned to group 2.0
- When ($Y_2 - Y_1$) is 0 and statement 2.2 is executed, control is transferred to statement 2.3
- Statement 2.3 says "Generate a carriage and line feed (the ! mark does this), type A POINT OF INTERSECTION IS, type the character X, type the value of X, type some spaces, type the characters $Y_1 = Y_2$, type the

computed value of Y_1, type two carriage returns and line feeds (!!)

- Statement 2.41 says return control to statement 1.1; 1.1 says increment the value of X and transfer control to group 2.0

Let's run the program!!

```
 GO
X=-  10.0000     Y1=- 137.0000   Y2=    7.0000
X=-   9.0000     Y1=- 114.0000   Y2=    6.0000
X=-   8.0000     Y1=-  93.0000   Y2=    5.0000
X=-   7.0000     Y1=-  74.0000   Y2=    4.0000
X=-   6.0000     Y1=-  57.0000   Y2=    3.0000
X=-   5.0000     Y1=-  42.0000   Y2=    2.0000
X=-   4.0000     Y1=-  29.0000   Y2=    1.0000
X=-   3.0000     Y1=-  18.0000   Y2=    0.0000
X=-   2.0000     Y1=-   9.0000   Y2=-   1.0000

A POINT OF INTERSECTION IS X=-   1.0000   Y1=Y2=-   2.0000

X=-   1.0000     Y1=-   2.0000   Y2=-   2.0000
X=    0.0000     Y1=    3.0000   Y2=-   3.0000
X=    1.0000     Y1=    6.0000   Y2=-   4.0000
X=    2.0000     Y1=    7.0000   Y2=-   5.0000
X=    3.0000     Y1=    6.0000   Y2=-   6.0000
X=    4.0000     Y1=    3.0000   Y2=-   7.0000
X=    5.0000     Y1=-   2.0000   Y2=-   8.0000

A POINT OF INTERSECTION IS X=    6.0000   Y1=Y2=-   9.0000

X=    6.0000     Y1=-   9.0000   Y2=-   9.0000
X=    7.0000     Y1=-  18.0000   Y2=-  10.0000
X=    8.0000     Y1=-  29.0000   Y2=-  11.0000
X=    9.0000     Y1=-  42.0000   Y2=-  12.0000
X=   10.0000     Y1=-  57.0000   Y2=-  13.0000
*
```

Once the computer has generated the coordinate pairs for each expression and identified the point(s) of intersection, we as humans would probably want to graph the two expressions to see what they looked like. This is a rather

tedious and time-consuming job so let's modify our program and have the computer plot a graph of the functions.

17

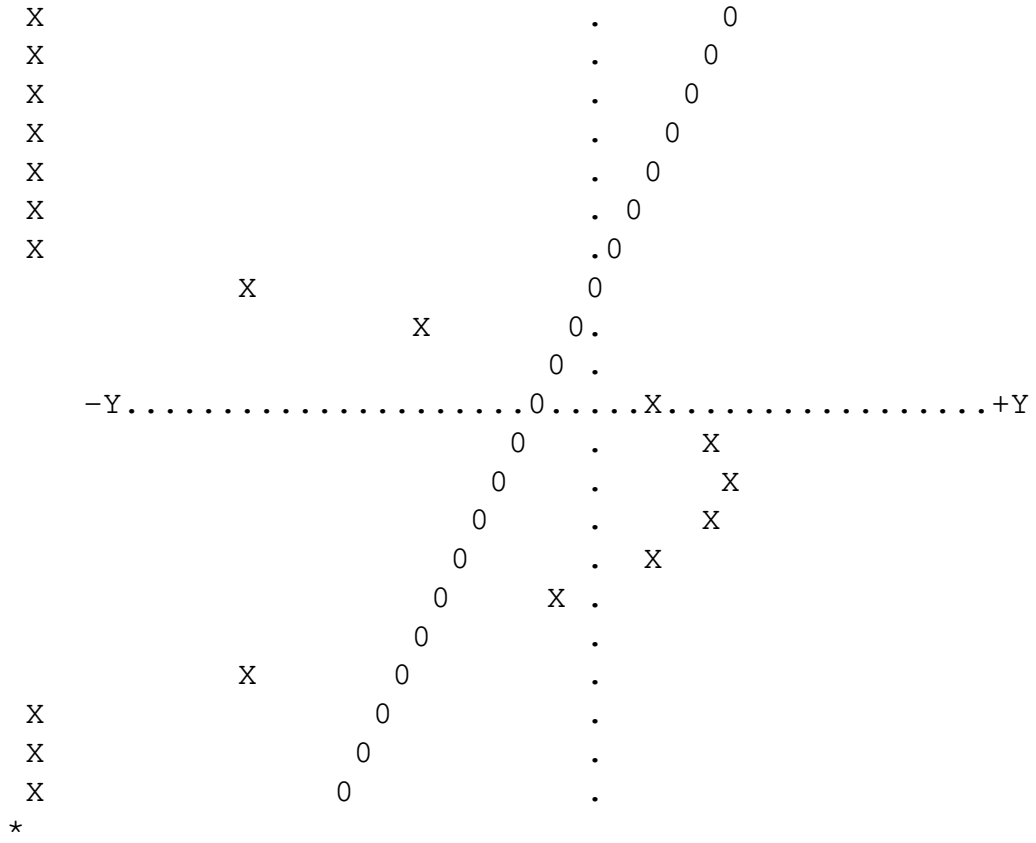
```
01.10 FOR X=-10,1,10;DO 2.0
01.20 TYPE !!!!!
01.30 FOR X=-10,1,10;DO 3.0
01.40 QUIT

02.10 SET Y1=-X^2+4*X+3;SET Y2=-X-3
02.20 IF (Y2-Y1) 2.4,2.3,2.4
02.30 TYPE !,"A POINT OF INTERSECTION IS ","X",X,"   ","Y1=Y2",Y1,!!
02.40 TYPE "X",X,"     ","Y1",Y1,"   ","Y2",Y2,!
02.41 RETURN

03.01 IF (X) 3.1,3.02,3.1
03.02 TYPE "    -Y.........................................+Y",#
03.10 FOR YS=0,1,29;TYPE " "
03.20 TYPE ".",#
03.30 FOR Y=0,1,29+(-X^2+4*X+3);TYPE " "
03.40 TYPE "X",#
03.50 FOR Y=0,1,29+(-X-3);TYPE " "
03.60 TYPE "0",!
03.70 RETURN
*
```

Let's run the program before we explain it this time.

```
 GO
X=-  10.0000     Y1=- 137.0000   Y2=    7.0000
X=-   9.0000     Y1=- 114.0000   Y2=    6.0000
X=-   8.0000     Y1=-  93.0000   Y2=    5.0000
X=-   7.0000     Y1=-  74.0000   Y2=    4.0000
```

```
X=-   6.0000     Y1=-  57.0000   Y2=    3.0000
X=-   5.0000     Y1=-  42.0000   Y2=    2.0000
X=-   4.0000     Y1=-  29.0000   Y2=    1.0000
X=-   3.0000     Y1=-  18.0000   Y2=    0.0000
X=-   2.0000     Y1=-   9.0000   Y2=-   1.0000

A POINT OF INTERSECTION IS X=-   1.0000   Y1=Y2=-   2.0000

X=-   1.0000     Y1=-   2.0000   Y2=-   2.0000
X=-   0.0000     Y1=    3.0000   Y2=-   3.0000
X=    1.0000     Y1=    6.0000   Y2=-   4.0000
X=    2.0000     Y1=    7.0000   Y2=-   5.0000
X=    3.0000     Y1=    6.0000   Y2=-   6.0000
X=    4.0000     Y1=    3.0000   Y2=-   7.0000
X=    5.0000     Y1=-   2.0000   Y2=-   8.0000

A POINT OF INTERSECTION IS X=    6.0000   Y1=Y2=-   9.0000

X=    6.0000     Y1=-   9.0000   Y2=-   9.0000
X=    7.0000     Y1=-  18.0000   Y2=-  10.0000
X=    8.0000     Y1=-  29.0000   Y2=-  11.0000
X=    9.0000     Y1=-  42.0000   Y2=-  12.0000
X=   10.0000     Y1=-  57.0000   Y2=-  13.0000
```

18

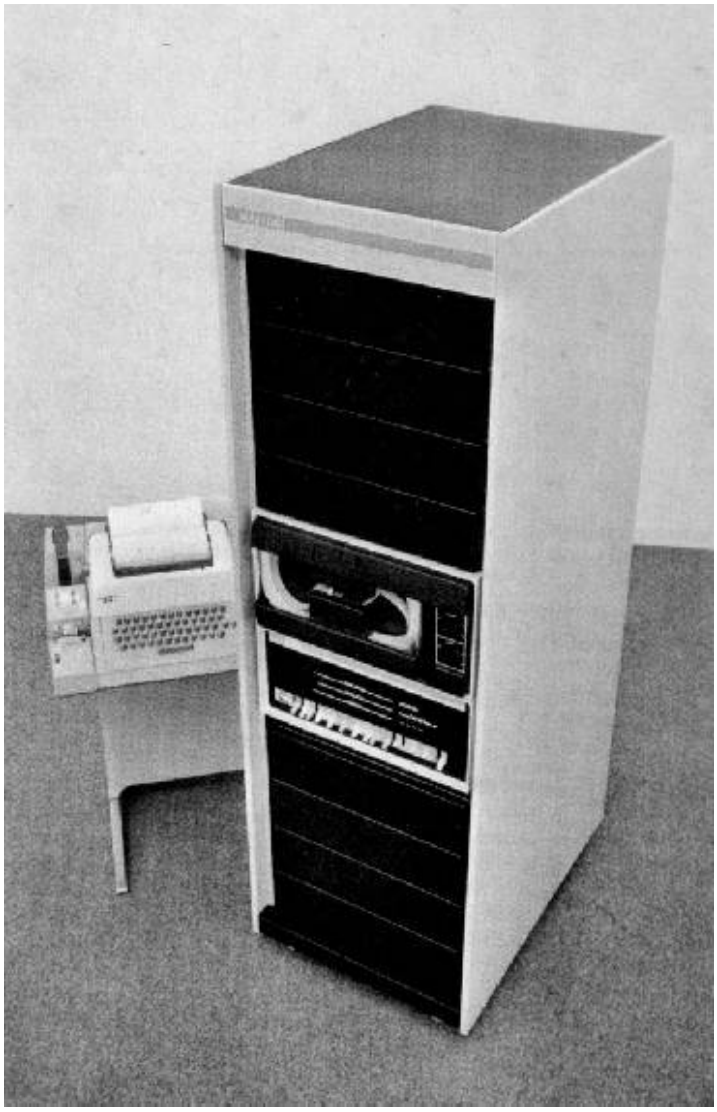



```
*
```

The first portion of the program ran as before so let's start at Statement 1.2: (i.e. first half of the program includes computations generated by 01.10 and all of group 2.0)

- Statement 1.2 says move the paper up five spaces
- Statement 1.3 says "start with X = - 10 and do everything in group 3 before incrementing the value of X.
- Statement 3.01 says "if X is less than or greater than 0, transfer control to statement 3.1; if X equals 0, transfer control to Statement 3.02
- Assume X= - 10 Statement 3.01 transfers contro to Statement 3.1
- Statement 3.1 says "move the typewriter carriage 29 spaces (for YS=0, 1,29; TYPE " ")
- After the carriage has been positioned, Statement 3.2 says "type the symbol (".") and then move the carriage back to the left margin (#)"
- Statement 3.3 says "evaluate the expression X^2 + 4*X + 3. Add 29 to that value and move the carriage over that many spaces". (for YS=0,1,29 + (- X^2 + 4*X + 3); TYPE " ")
- Statement 3.4 says "Type a symbol X ("X") to identify the point and then move the carriage back to the left margin (#).
- Statement 3.5 says "evaluate the expression -X - 3, add 29 to that value and move the carriage over that many spaces.
- Statement 3.6 says "type the character Ø, and then generate a carriage return and feed the paper upward one space.
- Statement 3.7 says "Return control to 1.3, increment the value of X by 1, and return control to group 3.

19



PDP-8/L -- one of the lowest-cost full scale digital computers available.

20

Assume X = 0

> Statement 3.01 now transfers control to statement 3.02 and causes a Y axis to be generated; once the Y axis has been generated the carriage is returned to the left margin (#) and the program continues to Statement 3.1 through 3.7.

We have seen that in our original effort we instructed the computer to find the intersection of two functions. By changing the program ever so slightly we instructed the computer to output coordinate pairs in our second effort. By adding a plotting routine as group 3, we instructed the computer to graph the function for us.

By writing a simple and straight forward set of instructions in the English language, we have instructed the computer to evaluate two expressions and produce a graphical output.

By again making simple modifications to the program, we can change the program to a general form:

Let's add a statement prior to 1.1, and change statements 2.1, 3.3, and 3.5 slightly. The changes will modify the program to have each user input values of A, B, C, D, and E.

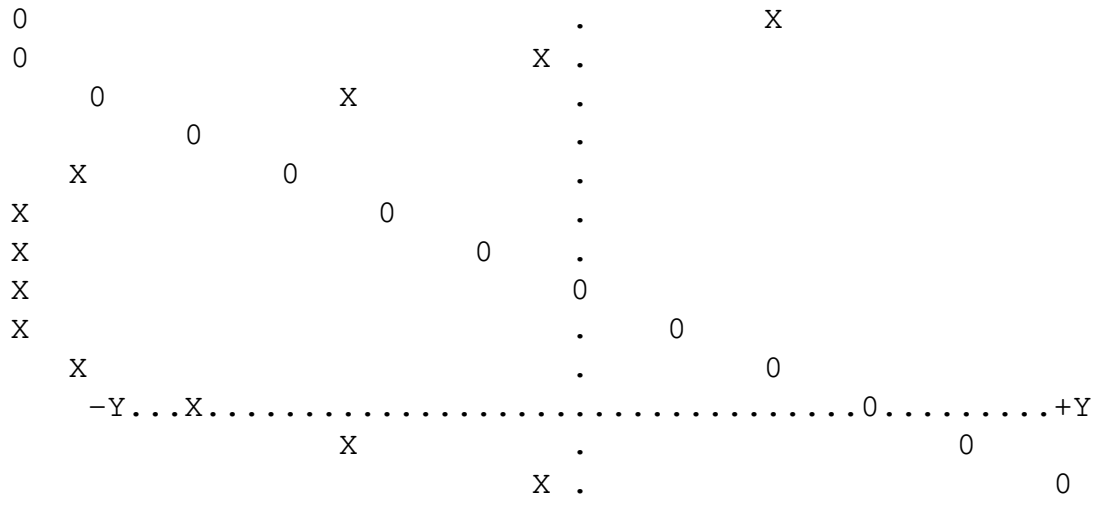
```
01.01 ASK "A",A,"B",B,"C",C,"D",D,"E",E
01.10 FOR X=-10,1,10;DO 2.0
01.20 TYPE !!!!!
01.30 FOR X=-10,1,10;DO 3.0
01.40 QUIT

02.10 SET Y1=A*X^2+B*X+C;SET Y2=D*X+E
02.20 IF (Y2-Y1) 2.4,2.3,2.4
02.30 TYPE !,"A POINT OF INTERSECTION IS ","X",X,"   ","Y1=Y2",Y1,!!
02.40 TYPE "X",X,"     ","Y1",Y1,"   ","Y2",Y2,!
02.41 RETURN

03.01 IF (X) 3.1,3.02,3.1
03.02 T "     -Y.............................................+Y",#
03.10 FOR Y=0,29;TYPE " "
03.20 TYPE ".",#
03.30 FOR Y=0,29+(A*X^2+B*X+C);TYPE " "
03.40 TYPE "X",#
03.50 FOR Y=0,29+(D*X+E);TYPE " "
03.60 TYPE "0",!
03.70 RETURN
*
```

```
 GO
A:1 B:7 C:-20
```

```
D:5 E:15
X=-  10.0000     Y1=   10.0000   Y2=-  35.0000
X=-   9.0000     Y1=-   2.0000   Y2=-  30.0000
X=-   8.0000     Y1=-  12.0000   Y2=-  25.0000
```

21

```
A POINT OF INTERSECTION IS X=-   7.0000   Y1=Y2=-  20.0000

X=-   7.0000     Y1=-  20.0000   Y2=-  20.0000
X=-   6.0000     Y1=-  26.0000   Y2=-  15.0000
X=-   5.0000     Y1=-  30.0000   Y2=-  10.0000
X=-   4.0000     Y1=-  32.0000   Y2=-   5.0000
X=-   3.0000     Y1=-  32.0000   Y2=    0.0000
X=-   2.0000     Y1=-  30.0000   Y2=    5.0000
X=-   1.0000     Y1=-  26.0000   Y2=   10.0000
X=    0.0000     Y1=-  20.0000   Y2=   15.0000
X=    1.0000     Y1=-  12.0000   Y2=   20.0000
X=    2.0000     Y1=-   2.0000   Y2=   25.0000
X=    3.0000     Y1=   10.0000   Y2=   30.0000
X=    4.0000     Y1=   24.0000   Y2=   35.0000

A POINT OF INTERSECTION IS X=    5.0000   Y1=Y2=   40.0000

X=    5.0000     Y1=   40.0000   Y2=   40.0000
X=    6.0000     Y1=   58.0000   Y2=   45.0000
X=    7.0000     Y1=   78.0000   Y2=   50.0000
X=    8.0000     Y1=  100.0000   Y2=   55.0000
X=    9.0000     Y1=  124.0000   Y2=   60.0000
X=   10.0000     Y1=  150.0000   Y2=   65.0000
```



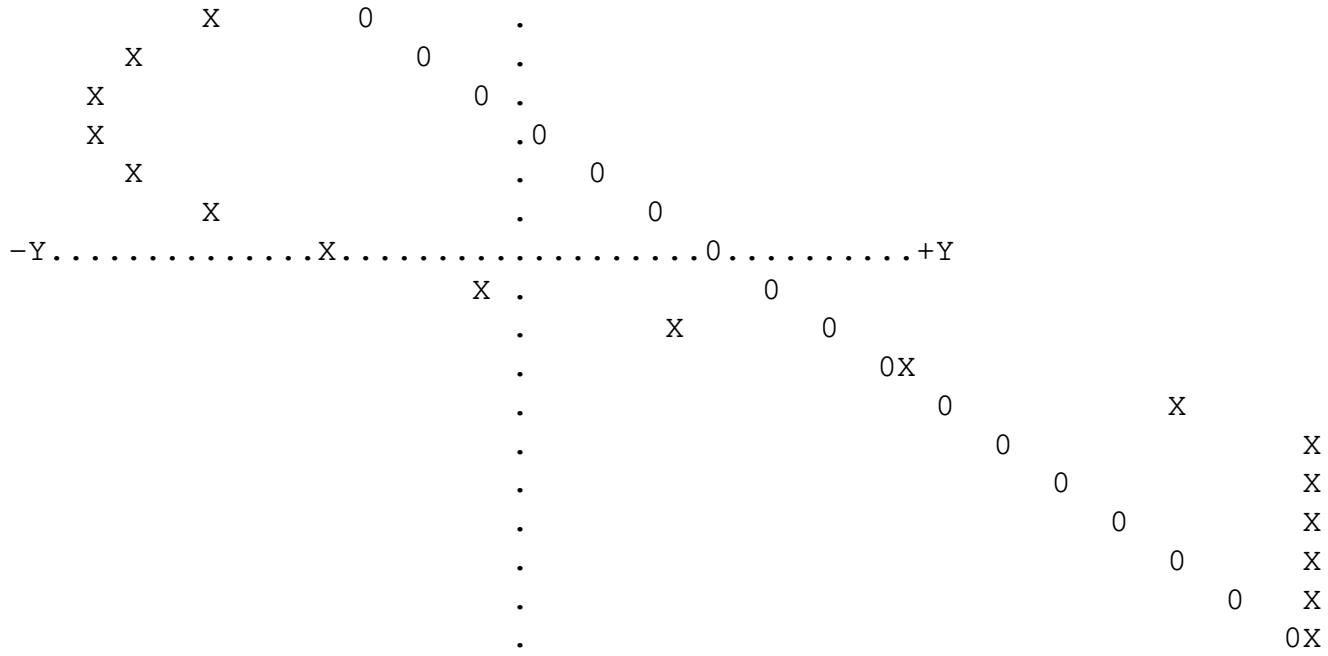
```
                                .          X                    0
                                .                         X          0
                                .                                       0
                                .                                         0
                                .                                         0
                                .                                         0
                                .                                         0
                                .                                         0
*
```

The previous programs to find the intersection of two functions require the functions to have an integer intercept, otherwist the programs do not recognize the points of intersection. A program to find the intercept, whether it is integer or fractional, is shown on the following page

---

```
A:1 B:7 C:-10
D:3 E:10
LOWER LIMIT:-10
UPPER LIMIT:10

A PT. OF INTERSECTION IS X=-   6.9004   Y1=Y2=-  10.6872


A PT. OF INTERSECTION IS X=-   6.8904   Y1=Y2=-  10.7551


A PT. OF INTERSECTION IS X=    2.8893   Y1=Y2=   18.5731


A PT. OF INTERSECTION IS X=    2.8993   Y1=Y2=   18.7010
```

```
          0                     .                     X
             0                  .        X
                0               .
                   0X           .
```




```
01.10 ASK "A",A,"B",B,"C",C,"D",D,"E",E
01.20 ASK "LOWER LIMIT",LL
01.30 ASK "UPPER LIMIT",UL
01.40 FOR X=LL,.1,UL;DO 2.0
01.50 TYPE !!!!!
01.60 FOR X=LL,1,UL;DO 3.0
01.70 QUIT

02.10 SET Y1=A*X^2+B*X+C;SET Y2=D*X+E
02.20 IF (FABS(Y1-Y2)-.1) 2.3,2.3,2.4
02.30 TYPE !,"A PT. OF INTERSECTION IS X",X,"   ","Y1=Y2",Y1,!!
02.40 RETURN
```

23

```
03.01 IF (X) 3.1,3.02,3.1
03.02 TYPE "    -Y.........................................+Y",#
03.10 FOR Y=0,29;TYPE " "
03.20 TYPE ".",#
03.30 FOR Y=0,29+(A*X^2+B*X+C);TYPE " "
03.40 TYPE "X",#
03.50 FOR Y=0,29+(D*X+E);TYPE " "
03.60 TYPE "0",!
03.70 RETURN
```

*

One family of problems that requires lengthy, time-consuming and tedius calculations is sinusoidal expressions. Students learn concepts of sinusoidals to prepare them for college work. College students explore sinusoidals in more depth to prepare them for the industrial world. The industrial world uses sinusoidals in many, many fields.

Let's write a program that solves sinusoidal expressions

In words, our problem says:

Compare the sinusoidal functions:

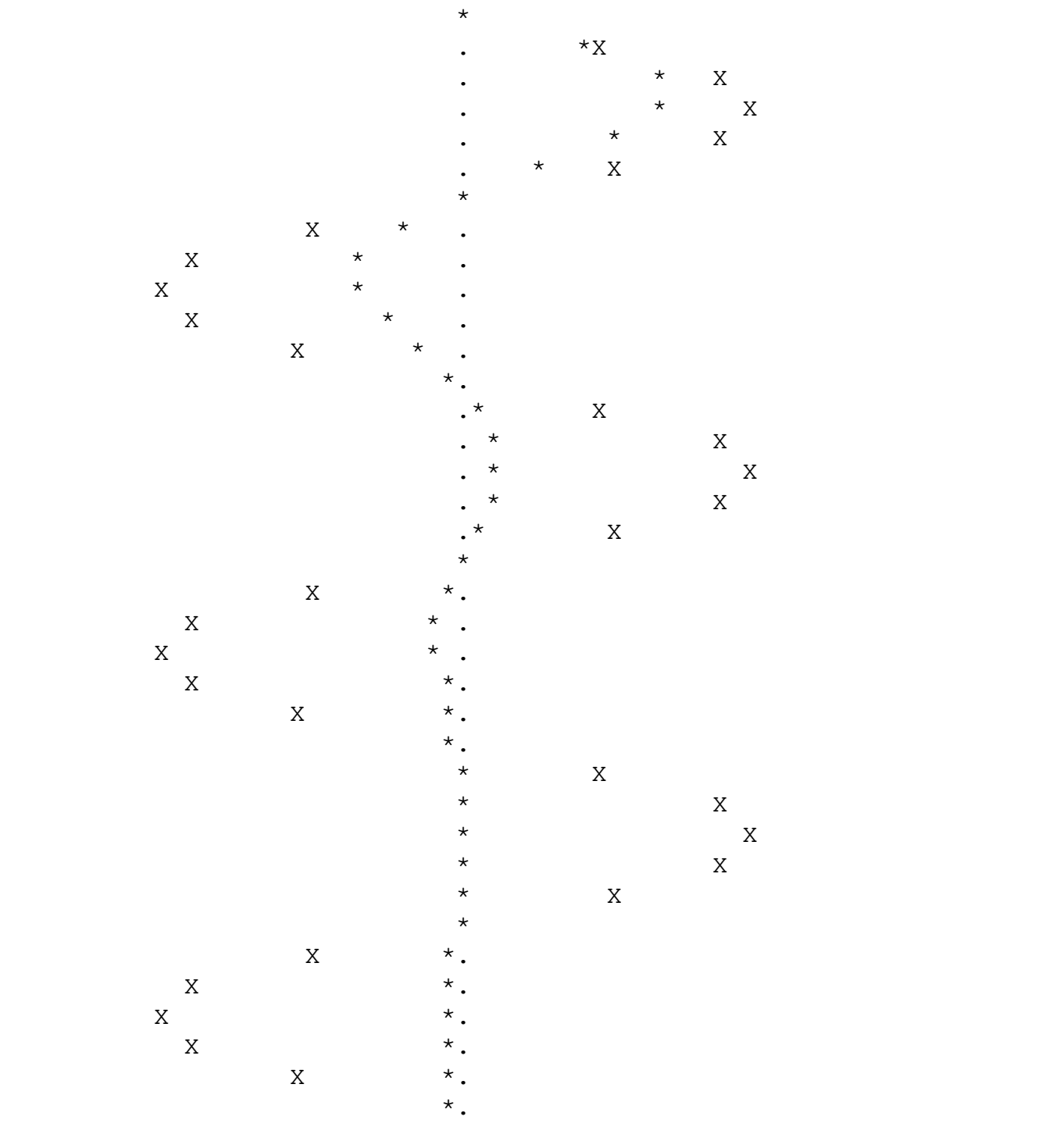
$Y1 = A1 * SIN(WT-T_0)$
and
$Y\ Damped = A1 * e^{-T} * SIN(WT-T_0)$

This is a difficult problem for a human, but no problem for the computer.

```
01.10 ASK "A1",A1,"OMEGA",W,"T0",T0,"DAMPING FACTOR",DAMPINGFACTOR
01.20 ASK "HOW MANY PERIODS DO YOU WANT TO PLOT ?",PERIOD
01.25 ASK "WHAT INCREMENT DO YOU WANT TO PLOT IN ?",INCREMENT
01.30 SET PI=3.14156
01.40 FOR T=0,INCREMENT*PI/180,PERIOD*2*PI;DO 2.0
01.50 QUIT

02.10 SET Y1=A1*FSIN(W*T-T0)
02.20 SET YD=A1*FEXP(-DAMPINGFACTOR*T)*FSIN(W*T-T0)
02.30 FOR Y=0,29;TYPE " "
02.40 TYPE "."#
02.50 FOR Y=0,29+(Y1);TYPE " "
02.60 TYPE "X",#
02.70 FOR Y=0,29+(YD);TYPE " "
02.80 TYPE "*",!
02.90 RETURN
*GO
A1:20
```

```
OMEGA:1
T0:0
DAMPING FACTOR:.25
HOW MANY PERIODS DO YOU WANT TO PLOT ?:3
WHAT INCREMENT DO YOU WANT TO PLOT IN ?:30
```

24



```
*
```

Another type program which requires simple but time-consuming calculations is series evaluation.

Stated in English:

Find the sum of:

$$\sum_{I=0}^{I=10} (1 + I)^I$$

and give partial sum for each value of I.

A FOCAL program looks like this:

```
01.10 SET SUM=0
01.20 SET PARTIALSUM=0
01.30 FOR I=0,1,10;DO 2.0
01.40 TYPE !!!!!,"THE FINAL SUM IS",SUM,!
01.50 QUIT

02.10 SET PARTIALSUM=(1+I)^I
02.20 SET SUM=SUM+PARTIALSUM
02.30 TYPE "I",I,!,"PARTIAL SUM",PARTIALSUM,!,"SUM",SUM,!!
02.40 RETURN
```

25

```
*GO
I=    0.0000
PARTIAL SUM=    1.0000
SUM=    1.0000

I=    1.0000
PARTIAL SUM=    2.0000
SUM=    3.0000

I=    2.0000
PARTIAL SUM=    9.0000
SUM=   12.0000

I=    3.0000
PARTIAL SUM=   64.0000
SUM=   76.0000
```

```
I=    4.0000
PARTIAL SUM=  625.0000
SUM=  701.0000

I=    5.0000
PARTIAL SUM= 7776.0000
SUM= 8477.0000

I=    6.0000
PARTIAL SUM= 117649.00
SUM= 126126.00

I=    7.0000
PARTIAL SUM= 2097150.0
SUM= 2223280.0

I=    8.0000
PARTIAL SUM= 43046700
SUM= 45270000

I=    9.0000
PARTIAL SUM= 0.10000010E+10
SUM= 0.10452700E+10

I=   10.0000
PARTIAL SUM= 0.25937410E+11
SUM= 0.26982700E+11
```

If we aren't interested in partial sums, we can delete statement 2.3 and run the program:

```
THE FINAL SUM IS= 0.26982700E+11
*
```

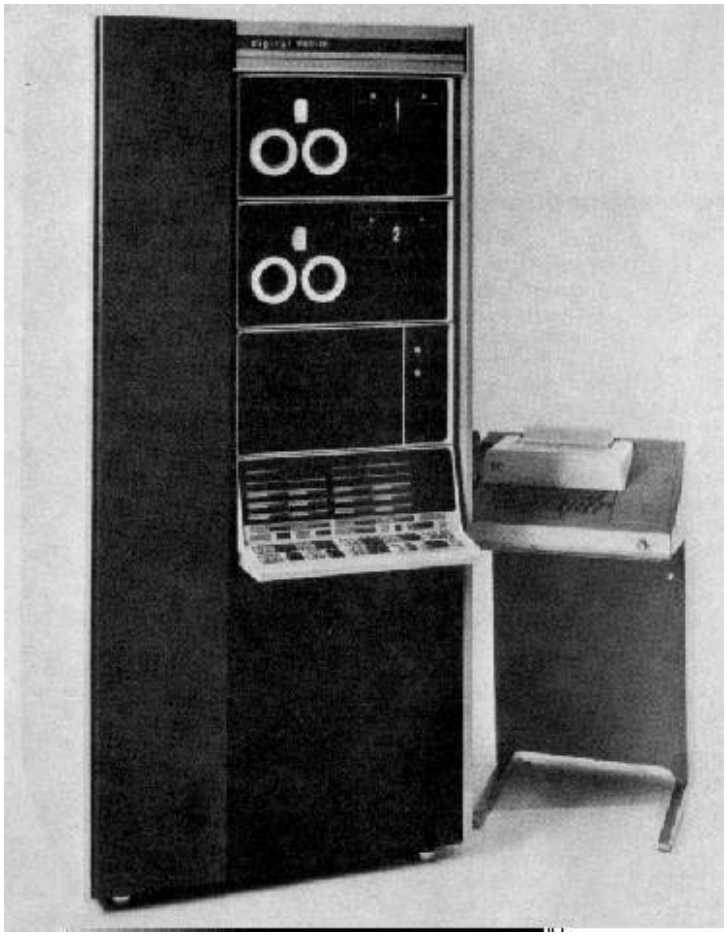
The preceeding examples have illustrated the ease of use of FOCAL. If your problem can be stated in the English language, it can just as easily be programmed in FOCAL. FOCAL allows you to communicate with the computer in your own terms. Think of the possibilities. They're limitless.

Our intent has been to give you an introduction to the FOCAL language. If you would like to learn more about FOCAL and its capabilities, we encourage you to request a complementary copy of "Introduction to Programming" from Digital Equipment Corporation.

In addition to calculating power, FOCAL also has display (CRT) capability, analog to digital capabilities, real-time capabilities and many more capabilities,

all described in "Introduction to Programming".

26



PDP-12 -- unified display-based programming system capable of executing LINC and PDP-8 family instructions.

27

FOCAL'S FUNCTIONS

		FCOS( )	Cosine
FSQT( )	Square Root	FATN( )	Arctangent
FABS( )	Absolute Value	FLOG( )	Naperian Log
FSGN( )	Sign Part of the Expression	FDIS( )	Scope Functions
FITR( )	Integer Part of the Expression	FADC( )	Analog to Digital Input Function
FRAN( )	A noise Generator		

FEXP( )	Natural Base to the Power	FNEW( )	User Function
FSIN( )	Sine	FCOM( )	Storage Function

FOCAL'S ERROR DIAGNOSTICS*

Code	Meaning
?00.00	Manual start given from console.
?01.00	Interrupt from keyboard via CTRL/C.
?01.40	Illegal step or line number used.
?01.78	Group number is too large.
?01.96	Double periods found in a line number.
?01.:5	Line number is too large.
?01.;4	Group zero is an illegal line number.
?02.32	Nonexistant group referenced by 'DO'.
?02.52	Nonexistant line referenced by 'DO'.
?02.79	Storage was filled by push-down-list.
?03.05	Nonexistant line number used after 'GOTO' or 'IF'.
?03.28	Illegal command used.
?04.39	Left of "=" in error in 'FOR' or 'SET'.
?04.52	Excess right terminators encountered.
?04.60	Illegal terminator for 'FOR' command.
?04.:3	Missing argument in Display command.
?05.48	Bad argument to 'MODIFY'
?06.06	Illegal use of function number.
?06.54	Storage is filled by variables.
?07.22	Operator missing in expression or double 'E'.
?07.38	No operator used before parenthesis.
?07.:9	No argument given after function call.
?07.;6	Illegal function name or double operators.
?08.47	Parenthesis do not match.
?09.11	Bad argument to 'ERASE'.
?10.:5	Storage was filled by text.
?11.35	Input buffer has overflowed.
?20.34	Logarithm of zero requested.
?23.36	Literal number is too large.
?26.99	Exponent is too large or negative.
?28.73	Division by zero requested.
?30.05	Imaginary square root requested.
?31.<7	Illegal character, unavailable command, or unavailable function used.

*for FOCAL 1969 only.

28

FOCAL COMMAND SUMMARY

Command	Abbreviation	Example of Form	Explanation
ASK	A	ASK X, Y, Z	FOCAL types a colon for each variable; the user types a value to define each variable.
COMMENT	C	COMMENT	If a line begins with the letter C, the remainder of the line will be ignored.
CONTINUE	C	C	Dummy lines
DO	D	DO 4.1	Execute line 4.1; return to command following DO command.
		DO 4.0	Execute group 4 lines.
		DO ALL	Return to command following DO command, or when a RETURN is encountered.
ERASE	E	ERASE	Erases the symbol table.
		ERASE 2.0	Erases all group 2 lines.
		ERASE 2.1	Deletes line 2.1.
		ERASE ALL	Deletes all user input.
FOR	F	For i = x,y,z; (commands)	Where the command following is executed at each new value.
		For i = x,z; (commands)	x = initial value of i y = value added to i until i is greater than z.
GO	G	GO	Starts indirect program at lowest numbered line number.
GO ?	G ?	GO ?	Starts at lowest numbered line number and traces entire indirect program until another ? is encountered, until an error is encountered, or until completion of program.

GOTO	G	GOTO 3.4	Starts indirect program (transfers control to line 3.4) Must have argument.

IF	I	IF (X) Ln, Ln, Ln
		IF (X) Ln,Ln;
		(commands)
		IF (X) Ln;
		(commands)
			Where X is a defined identifier, a value, or an expression, followed by one to three line numbers. If X is less than zero, control is transferred to the first line number, if X is equal to zero, control is to the second line number. If X is greater than zero, control is to the third line number.

29

LIBRARY CALL	L C	LIBRARY CALL name	Calls stored program from the disk.
LIBRARY DELETE	L D	LIBRARY DELETE name	Removes program from the disk.
LIBRARY LIST	L L	LIBRARY LIST	Types directory of stored program names.
LIBRARY SAVE	L S	LIBRARY SAVE name	Saves program on the disk.
LINK	L L		For disk monitor system; FOCAL types 4 locations indicating start and end of text area, end of variable list and bottom of push-down list.
LOCATIONS	L L		For paper-tape system; types same locations as LINK.
MODIFY	M	MODIFY 1.15	Enables editing of any character on line 1.15 (see below).
QUIT	Q	QUIT	Returns control to the user.
RETURN	R	RETURN	Terminates DO subroutines, returning to the original sequence.

SET	S	SET A = 5/B*C	Defines identifiers in the symbol table.
TYPE	T	TYPE A + B - C;	Evaluates expression and types out = and result in current output format.
		TYPE A - B, C/E;	Computes and types each expression separated by commas.
		TYPE "TEXT STRING"	Types test. may be followed by ! to generate carriage return-line feed, or # to generate carriage return.
WRITE	W	WRITE WRITE ALL	FOCAL types out the entire indirect program.
		WRITE 1.0	FOCAL types out all group 1 lines.
		WRITE 1.1	FOCAL types out line 1.1.

30

FOCAL OPERATIONS AND THEIR SYMBOLS

Mathematical operators:

^ Exponentiation
* Multiplication
/ Division
+ Addition

-	Subtraction	

Control Characters:

%	Output format delimiter	
!	Carriage return and line feed	
#	Carriage return	
$	Type symbol table contents	
( )	Parentheses	
[ ]	Square brackets	(mathematics)
< >	Angle brackets	
" "	Quotation marks	(text string)
? ?	Question marks	(trace feature)
*	Asterisk	(high-speed reader input)

Terminators:

	SPACE key (names)	
	RETURN key (lines)	(nonprinting)
	ALT MODE key (with ASK statement)	
,	Comma (expressions)	
;	Semicolon (compounds and statements)	

FOCAL COMMAND SUMMARY

Command	Abbreviation	Example of Form
ASK	A	ASK X, Y, Z
COMMENT	C	COMMENT
CONTINUE	C	C
DO	D	DO 4.1
		DO 4.0 DO ALL
ERASE	E	ERASE
		ERASE 2.0
		ERASE 2.1
		ERASE ALL
FOR	F	For i = x,y,z; (commands)
		FOR i = x,z; (commands)
GO	G	GO
GO ?	G ?	GO ?
GOTO	G	GOTO 3.4
IF	I	IF (X) Ln, Ln, Ln
		IF (X) Ln, Ln; (commands)
		IF (X) Ln; (commands)
LIBRARY CALL	L C	LIBRARY CALL name
LIBRARY DELETE	L D	LIBRARY DELETE name
LIBRARY LIST	L L	LIBRARY LIST
LIBRARY SAVE	L S	LIBRARY SAVE name
LINK	L	L
LOCATIONS	L	L
MODIFY	M	MODIFY 1.15
QUIT	Q	QUIT
RETURN	R	RETURN
SET	S	SET A = 5/B*C;
TYPE	T	TYPE A + B - C;
		TYPE A - B,C/E;
		TYPE "TEXT STRING"
WRITE	W	WRITE
		WRITE ALL
		WRITE 1.0
		WRITE 1.1

Index to DEC's FOCAL 1969 Promotional Booklet

Notes

Neither the author nor the illustrator of this pamphlet are known, but we do know that FOCAL and the FOCAL 1969 interpreter for the PDP-8 were written by Richard Merrill. It is possible that he wrote this pamphlet. The sloppy proofreading, redundant wording and one particularly poorly chosen example suggest that this pamphlet was probably produced in a hurry!

Under Internet browsers with reasonable support for cascading style sheets, a fair reproduction of the "look and feel" of the original booklet can be obtained by sizing the browser's window so that this paragraph completely fills 3 lines.

In this transcription, every effort has been made to preserve the typography of the original, but the results you see may vary depending on your WWW browser. Page breaks in the original are marked here with horizontal rules; the original was printed on 8 sheets of 8 1/2 by 11 inch newsprint with a lightweight cardstock cover, then stapled and folded to make a 5 1/2 by 8 1/2 booklet. All program examples in the original were teletype listings photoreproduced at a 60 % reduction. The text in the original was set in a sans-serif font, probably News Gothic Light, set 7 lines per inch (8 point type?) in lines 6 3/8 inch long.

There were many typos in the original, and an effort has been made to transcribe these verbatim. The most extreme error, on page 23, pairs a version of an example program with output that came from a

different version of that program! At a higher level, the same example illustrates a numerically incorrect algorithm!

Because HTML contains no provisions for overprinting, some of the teletype output examples are not reproduced accurately here. There are also some problems with the typesetting of the Greek letter sigma Σ and the square root sign √, although this works better after the transition to Unicode. The table facilities of HTML have proven to be remarkably effective for reproducing tha tables in the original, although the layouts of the table of operations and their symbols and the final command summary pushed the limits of these features! Attempts to use style sheets to set the page size relative to the font size failed; this is why you have to resize the window yourself.