



M

VA M Programming Introduction

Lesson 1

Writing Your First M Routine

VA M Programming

Introduction

Table of Contents

Guide to Symbols Used	4
Optional Reading References	4
References	4
Document Update Note	4
Objectives	5
Computer Programs	5
Input	5
Process	5
Output	5
Types of Programs	5
Batch	5
Interactive	5
Hybrid	5
Program Development Process	6
The Characteristics of M	7
General M Syntax	8
Commands and Arguments	9
Constants	9
String Constants (strings)	9
Numeric Constants	10
Variables	10
Beginning M Commands	12
Input Command READ	12
Output Command WRITE	15
Assignment Command SET	18
Assignment Command SET and Arithmetic Operators	20
Conditional Command IF	22
Flow Control Command QUIT and HALT	24
Putting a routine together	25
Step 1--Define the Problem	25
Step 2--Identify the Output	25
Step 3--Define the Input	25
Step 4--Develop a Method of Solution (Algorithm)	26
Step 5 --Pseudocode as Programming Tool	26

VA M Programming

Introduction

Table of Contents *(Continued)*

Step 6--Convert the Algorithm into an M Routine	29
Writing the routine first on paper	29
Entering the Routine into the Computer	34
M Direct Mode	34
M Deferred Mode	35
Line Labels	35
Offsets	36
ZREMOVE	37
ZPRINT	37
ZSAVE	38
Editing Your Routine	38
Summary of DSM-11 Editing	39
Step 7--Test and Debug the Routine	39
More M Commands	40
Comments	40
Flow Control Command GOTO	42
Assignment Command KILL	43
Running Your Routine	45
Assignment (optional)	46
Assignment Specifications	46

Acknowledgements

Developing a comprehensive training package for the **M** computer programming language as it is used in VA was not an easy task. Much gratitude goes to the members of the M Programming Language Training Development Team who spent countless hours developing and reviewing material. Sincere appreciation to:



Debbie Channell
Tuscaloosa OI Field Office, Customer Service

Harris H. Lloyd
Central Arkansas Veterans Health Care System

Bob Lushene
Hines OI Field Office, Technical Service

VA M Programming Introduction

Guide to Symbols Used

<ENTER>	Press the Enter or Return key
<TAB>	Press the Tab key
<space>	One space
	What follows is from part of the VA Programming Standards and Conventions.
	This indicates an OPTIONAL reading reference. Additional information on this topic may be found in the references listed below.

Optional Reading References

Lewkowicz, John. *The Complete M: Introduction and Reference Manual for the M Programming Language*. 1989.

Walters, Richard. *M Programming – A Comprehensive Guide*. 1997.

The optional reading references are available through:

M Technology Association
1738 Elton Road, Suite 205
Silver Spring, MD 20903
301-431-4070

References

The VA M Programming Standards and Conventions shown were taken from “VA DHCP Programming Standards and Conventions” document, prepared March 19, 1991, modified in 1995 and approved January 22, 1996.

<http://vaww.vista.med.va.gov/Policies/sacc.htm>.

Document Update Note

January 2007 – To meet current standards on personal data privacy and accessibility, this document was updated. It was edited to include document metadata and edited to de-identify any personal data. Only the information in this document is affected. This should have no effect on the M routines.

VA M Programming

Introduction

Objectives

The objective of this lesson is to prepare you to write your first interactive M routine using the following programming techniques and M commands:

- The seven steps of the program development process, including pseudocode as a programming tool
- General M syntax
- Constants and variables
- Input command READ
- Output command WRITE
- Assignment command SET
- Conditional command IF
- Flow control commands QUIT, HALT, GOTO
- Comments
- Assignment command KILL

Computer Programs

All computer programs regardless of the programming language consist of input, processing, and output.

Input identifies all data needed for a program such as patient name, SSN, date of birth, etc.

Process is what is done to the input data - an algorithm.

Output is what reports or information the process produces.

Types of Programs

There are three main types of programs.

One type of program is the **Batch** program. This is where several processes are done concurrently, usually in the background.

Another type is an **Interactive** program. This is where the user is prompted for input during processing.

Then, there are **Hybrid** programs. These usually involve interfaces, such as HL7 messages, laboratory instruments, etc.

Program Development Process

Developing a program involves the following steps, which should be accomplished in the following order:

First, **define the problem** or your expected outcome. Identify what the program should accomplish.

Second, **define the output or expected results**. By identifying the results, you get a better indication of the input data required and the algorithm needed to produce the desired outcome.

Third, **define the input**. List all the information or data needed by the program.

Fourth, **develop a method of solution** (algorithm). This method may include arithmetic, logical, or control processes. In other words, what is the formula to be used, how are the calculations done, etc.

Fifth, **use pseudocode as a programming tool**. Write the program code using plain English sentences.

Sixth, **convert the algorithm into a computer program**. Write your program!

Seventh, **test the program**. A program is not necessarily correct if it merely runs. Are the calculations correct? Did you receive the correct results? Are the reports formatted properly? Debug the program, if necessary.

VA M Programming Introduction

The Characteristics of M(UMPS)

Originally, the language was designated MUMPS, an acronym which stood for Massachusetts General Hospital Utility Multi-Programming System. Several years ago, the name was changed to "M." You may still see references to MUMPS, however, M will be used throughout this course.

Some of the characteristics of M language are:

- It is a general-purpose database management system.
- It has many attributes of an operating system built-in.
- It is an interpreted language (although some versions can be compiled); routines can be run immediately after editing.
- M is one of four programming languages having an ANSI standard and is supported by the MUMPS Development Committee (MDC). The standard is enhanced frequently (every 2-4 years).
- M is available for use on a wide range of computer systems.



VA Programming Standards and Conventions

The 1995 ANSI X11/M Standard will be adhered to unless modified by the "SAC Programming Standards and Conventions" document approved January 22, 1996.

VA M Programming Introduction

General M Syntax

An M program, which may be one or more routines, is comprised of statements; statements are comprised of commands or commands with arguments.

Examples of M Syntax

Linelabel<tab>Command ← *A line label is optional in all statements.*

Command

Command<space>Argument(s)

Command<space><space>Command<space>Argument(s)

Command<space>Argument(s)<space>Command<space>Argument...

Note the symbol <space> indicates one space.

Statements may begin with an optional line label. Commands may have optional arguments and more than one command can appear on a single line. A command without arguments must be followed by two spaces when another command follows on the same line.



VA Programming Standards and Conventions

In Standard M, an individual statement can be a maximum of 255 characters in length. In the VA, however, no one statement should be greater than 245 characters.

VA M Programming

Introduction

Commands and Arguments

Examples of M Commands

READ, WRITE, GOTO, QUIT, SET, IF, HALT, KILL

Examples of M Arguments

Special variables:	\$TEST, \$SHOROLOG
Functions:	\$PIECE(), \$SELECT()
Constants:	15, 18.5, "HELLO"
Variables:	NAME, T1
Operators:	+, -, <, =
Expressions:	HOURS*RATE

Some examples of commands covered in this lesson are: READ, WRITE, GOTO, QUIT, SET, IF, HALT, and KILL. These commands may be abbreviated to their first letter. However, for instructional purposes, the full name of the command will be used throughout this lesson.

M arguments may consist of one or more of special variables, functions, constants, variables, operators, and expressions. These will be explained in more detail during this course.



VA Programming Standards and Conventions

Commands can be typed either in uppercase or lowercase letters. However, in the VA, commands must always be used in uppercase letters.

String Constants

A **String Constant** is any collection of characters.

Strings may be a maximum of 255 characters.

Strings must be enclosed in quotes (").

Examples of String Constants

```
"123A"  
"Greetings!"  
"Enter your name: "  
"123&^(@#($&ABC"
```

VA M Programming Introduction

Numeric Constants

A **Numeric Constant** is when all characters are numbers (0-9), usually not enclosed in quotes. The numeric constant can have a decimal point. It may also have a sign (+ or -) if the sign is the leftmost character.

Types of numeric constants:

integers

floating point numbers (real numbers)

exponential notation (a special case for representing numbers)

Examples of Numeric Constants

Integers	6	-234	999
Floating Point	+6.1	34.894	345
Exponential Notation	6.5E5 (6.5 x 10 to 5th power)	4.5E-3 (4.5 x 10 to -3 power)	1.0E2 (1.0 x 10 to 2nd power)



VA Programming Standards and Conventions

Any data element, which may be interpreted as a number, must contain no more than 15 significant digits.

Variables

A variable is a name, which represents a value. The variable name stays the same but the value can be changed at any time during the execution of the routine.

Rules for naming variables:

- The first character must be an alphabetic character or the % (percent) character.
- Up to seven alphanumeric characters may follow. Any characters beyond the eighth are ignored and can be confusing.
- No punctuation (including spaces) should be found in the variable name.
- Names can be in uppercase or lowercase characters, but M is case sensitive (i.e., an uppercase letter is different than a lowercase letter).

VA M Programming

Introduction

Valid Variable Names

NAME	T1	ABC123
AGE	%SET	%AGE
NORMAL	PRICE	Age (different than AGE)
name (different than NAME)		
Birthdate (is nine characters but only first eight are recognized)		

Invalid Variable Names

SET%	GROSS-PAY	DISC?
1T	CITY,STATE	
123ABC	TOTAL PAY	



VA Programming Standards and Conventions

Variable names must use uppercase characters

The following is NOT a formal VA Programming Convention but a "suggestion" when working within the VA: Application routines should not contain variables that are named beginning with % or the letter D.

The use of % is reserved for specialized variables.
Variables beginning with the letter D are used by FileMan.

VA M Programming

Introduction

Beginning M Commands

Input Command READ

The purpose of the READ command is to accept input from the user and put the response into a variable.

M Code Sample

```
READ ADDRESS
```

M Code Sample

```
>READ ADDRESS
123 Any Lane
>ZWRITE ADDRESS
ADDRESS="123 Any Lane"
>
```

In the example above, the command is typed in followed by <ENTER>. The computer will wait for the user to type a response followed by <ENTER>. That response is stored in the variable address.

The user, in this example, types in 123 Any Lane followed by <ENTER>. The computer returns with the programmer prompt and waits for the next command.

The command ZWRITE is typed to display the value stored in a variable name, which in this example, is address.

The variable was returned with the value that was entered.

Another example of a read command, where the user is prompted with a description of what response is expected, is shown below.

M Code Sample

```
READ NAME
```

M Code Sample

```
READ !,?20,"Enter your name: ",NAME
```

where ! and ? are format control characters:

! tells M to move to the next line

?n tells M to move to the column number specified by n

VA M Programming Introduction

The user is prompted to enter their name and the computer waits for the user's response, which will be stored in the variable name.

The exclamation and question marks are used for formatting. The exclamation mark tells M to move to the next line. The question mark tells M to move to the column number specified with the question mark, for example, question mark twenty means to move to column twenty.

The string `Enter your name:` is used for a description to the user as to what response is expected. The user's response is stored in the variable name.

M Code Sample

```
>READ !,?20,"Enter your name: ",NAME  
  
Enter your name: First Last  
>ZWRITE NAME  
>NAME = "First Last"
```

The READ command with the description is typed followed by <ENTER>.

The computer cursor will skip a line and move over to column twenty, display the prompt `Enter your name:`, and wait for the user to type a response followed by <ENTER>. That response is stored in the variable name.

The user, in this example, types Mickey Mouse, followed by <ENTER>. The computer returns with the programmer prompt and waits for the next command.

The command ZWRITE is typed to display the value stored in a variable name.

The variable was returned with the value that was entered.

VA M Programming Introduction

READ with time-out feature

(automatically sets value of a special variable \$TEST which is discussed later)

>READ !,?20,"Enter your name: ",NAME:5

Single-character READ (STAR READ)

>READ !,"Press any key to continue",*XYZ

Fixed character READ (POUND READ)

>READ !,?20,"Enter your name: ",NAME#25

! move to the next line
?n move to the column number specified by n
form feed (clear screen, new page)



VA Programming Standards and Conventions

#A2d: (1) All user-input READs must have time-outs. If the duration of the time-out is not specified by the variable DTIME and the duration exceeds 300 seconds, each occurrence must be documented in the technical manual...

#A2d: (2) User input shall be terminated by a carriage return. – Therefore, *READ and #READ are not allowed with user interaction but is allowed for interfaces, etc.

VA M Programming Introduction

Output Command WRITE

The purpose of the WRITE command is to display the values of constants and/or variables.

M Code Samples

```
READ !,?20,"Enter your name: ",NAME  
WRITE !,NAME
```

```
WRITE !!,?31,"Good afternoon ",NAME
```

```
WRITE !,?5,"Hurry up ",NAME,". It is time to move on."
```

```
WRITE "Hi There"
```

```
WRITE !,?20,"ABC COMPANY"
```

```
WRITE !,"The sum of 5+6=",5+6
```

```
ZWRITE (Shows the symbol table.)
```



A breakdown of each of the code samples from above follows.

VA M Programming

Introduction

M Code Sample

```
READ !,?20,"Enter your name: ",NAME
WRITE !,NAME
```

The READ statement is used to prompt the user to type a response, which will be stored in the variable NAME. Then, the WRITE statement is used to display the value of the variable NAME. This code reads: READ one line feed, move the cursor to column 20, and display the prompt Enter your name:. The cursor will then blink waiting for the user to type in a response, which will be stored in the variable NAME. After the user types a response followed by <ENTER>, the WRITE statement is executed, WRITE one line feed, and the value of the variable NAME is displayed.

M Code Sample

```
WRITE !!,?31,"Good afternoon ",NAME
```

Here, the WRITE statement is used to produce two line feeds, move the cursor to column 31, display the phrase Good afternoon, a space, and the value of the variable NAME.

M Code Sample

```
WRITE !,?5,"Hurry up ",NAME,". It is time to move on."
```

This WRITE statement is used to write one line feed, move the cursor to column five, display the phrase Hurry up, display the value of the variable NAME, and then add a period (.) and the phrase It is time to move on.

M Code Sample

```
WRITE "Hi There"
```

Here, the WRITE statement is used to display the phrase Hi There.

M Code Sample

```
WRITE !,?20,"ABC COMPANY"
```

This WRITE statement is used to write one line feed, move the cursor to column 20 and display the phrase ABC COMPANY.

VA M Programming Introduction

M Code Sample

```
WRITE !,"The sum of 5+6=",5+6
```

Here, the WRITE statement is used to write one line feed, the phrase The sum of 5+6=, and display the value of 5+6, which is eleven. Eleven will be displayed. The computer does the math of the equation.

M Code Sample

```
ZWRITE
```

Argumentless WRITE; shows the symbol table

The ZWRITE command displays all the values of the variables for the current computing session. This is called an argumentless WRITE statement, meaning it has no arguments. The program is not told what to display, only the command is issued. In this case, all values of all variables are displayed to the user.

VA M Programming Introduction

Assignment Command SET

The purpose of the SET command is to assign a value to a variable.

M Code Samples

```
SET AGE=20
WRITE !,AGE
20
```

```
SET CITY="SAN FRANCISCO"
WRITE !,CITY
SAN FANCISCO
```

```
SET A=5
SET B=6
SET C=6          (or SET A=5,B=6,C=6)
SET ANS=A+B+C
WRITE !,ANS
17
```

or

```
SET A=5
SET (B,C)=6
SET ANS=A+B+C
WRITE !,ANS
17
```

A breakdown of each of the code samples from above follows.

M Code Sample

```
SET AGE=20
WRITE !,AGE
20
```

In this sample code the SET command is used to assign a value to the variable AGE. A value of 20 is being assigned to the variable AGE. The WRITE statement is used to write one line feed and the value of the variable AGE, which in this case would display the number 20.

VA M Programming Introduction

M Code Sample

```
SET CITY="SAN FRANCISCO"  
WRITE !,CITY  
SAN FRANCISCO
```

The SET statement is used to assign the variable CITY a value of SAN FRANCISCO in this sample code. The WRITE statement is used to display the value of the variable CITY, which is SAN FRANCISCO.

M Code Sample

```
SET A=5  
SET B=6  
SET C=6          (or SET A=5,B=6,C=6)  
SET ANS=A+B+C  
WRITE !,ANS  
17
```

The SET statement is used to assign the variable A with a value of 5 in this code sample. The next statement is another SET statement, which assigns the variable B a value of 6. The next SET statement assigns the variable C a value of 6. The next SET statement is used to assign the variable ANS the value of the result of A plus B plus C, which is 5 plus 6 plus 6, which subsequently is the value 17. The WRITE statement produces one line feed and displays the value of the variable ANS, which is 17.

As a variation to the previous M code sample, we could have used one SET statement to assign the value of the variables B and C the value of 6, since both variables are assigned the same value. This set of code and the previous set of code equate to the same operations but this example is combining two SET statements into one by using the parenthesis to assign all variables within the parenthesis the same value of 6.

M Code Sample

```
SET A=5  
SET (B,C)=6  
SET ANS=A+B+C  
WRITE !,ANS  
17
```

VA M Programming Introduction

Assignment Command SET and Arithmetic Operators

There are seven arithmetic operators which can be used to form arithmetic expressions: **+**, **-**, *****, **/**, **#** (modulo division), **** (integer division), and ****** (exponentiation). Unlike other languages, M evaluates an arithmetic expression from left to right (no hierarchy of operations). You can, however, use parentheses to alter the order of evaluation since M will do operations in parentheses first.

M Code Samples

```
SET A=10,B=3,C=2,D=4,E=1
SET X=A+B*C-D/E+A+B*C/E
WRITE !,X
70
```

```
SET A=10,B=3,C=2,D=4,E=1
SET X=A+B*C-D/E+(A+(B*C)/E)
WRITE !,X
38
```



A breakdown of each of the code samples from above follows.

VA M Programming Introduction

M Code Sample

```
SET A=10,B=3,C=2,D=4,E=1
SET X=A+B*C-D/E+A+B*C/E
WRITE !,X
70
```

The SET statement is used to assign the variable A equal to 10, the variable B equal to 3, the variable C equal to 2, the variable D equal to 4, and the variable E equal to 1. The next SET statement assigns the variable X equal to a value as a result of performing the arithmetic operation A plus B times C minus D divided by E plus A plus B times C divided by E, which is 10 plus 3 times 2 minus 4 divided by 1 plus 10 plus 3 times 2 divided by 1. The arithmetic is performed left to right, 10 plus 3 is 13, times 2 which is 26, minus 4 which is 22, divided by 1 which is 22, plus 10 which is 32, plus 3 which is 35, times 2 which is 70, divided by 1 which is 70. Therefore the value of X is 70. The next statement is a WRITE statement, which performs a line feed and displays 70 which is the value of X.

M Code Sample

```
SET A=10,B=3,C=2,D=4,E=1
SET X=A+B*C-D/E+(A+(B*C)/E)
WRITE !,X
38
```

The same SET statements are used for the variables A, B, C, D, and E. A similar SET statement is used for the variable X but this time, parenthesis control the order of arithmetic operation. In this case, the math within the innermost parenthesis is performed first, B times C (3 times 2), which equals 6. The math within the next set of parenthesis, which is A plus B times C divided by E (10 plus 6 divided by 1), equals 16, is performed next. There are no more parenthesis so the remaining math is performed from left to right, A plus B times C minus D divided by E plus the result of the math that was done in the parenthesis, which equaled 16. So this part of the mathematical operation would be 10 plus 3, then 13 times 2, then 26 minus 4, then 22 divided by 1, then 22 plus the value of the operations previously done within the parenthesis, which equaled 16. This gives us a total of 38, which is 22 plus 16.

VA M Programming

Introduction

Conditional Command IF

The purpose of the IF command is to allow a command or commands to be executed only if a specified condition is true. If the condition is false, the command or commands are skipped.

M Code Samples

```
SET AGE=65
IF AGE=65 WRITE !,"Time to retire"
SET AGE=67
IF AGE=65 WRITE !,"Time to retire"
```

```
SET AGE=67
IF AGE>65 WRITE !,"Senior Citizen"
SET AGE=65
IF AGE>65 WRITE !,"Senior Citizen"
```

```
SET AGE=3
IF AGE<5 WRITE !,"Nursery school"
SET AGE=12
IF AGE<5 WRITE !,"Nursery school"
```



A breakdown of each of the code samples from above follows.

M Code Sample

```
SET AGE=65
IF AGE=65 WRITE !,"Time to retire"
SET AGE=67
IF AGE=65 WRITE !,"Time to retire"
```

In this code sample, the variable AGE is set to equal 65. The IF command evaluates the variable AGE to determine if the value equals 65. If true, the remainder of the line is executed. In this case it is true. Therefore the statement: Time to retire is written. The next line sets AGE equal to 67. Then the IF command evaluates the condition if AGE is equal to 65. In this case the condition is false. The remainder of the line is skipped and nothing is written.

VA M Programming

Introduction

M Code Sample

```
SET AGE=67
IF AGE>65 WRITE !,"Senior Citizen"
SET AGE=65
IF AGE>65 WRITE !,"Senior Citizen"
```

In this code sample the variable AGE is set to equal 67. The IF command evaluates the variable AGE to determine if the value is greater than 65. If true, the remainder of the line is executed. In this case it is true. Therefore, the statement: Senior Citizen is written. The next line sets AGE equal to 65. Then the IF command evaluates if AGE is greater than 65. In this case the condition is false. 65 is not greater than 65. Therefore the remainder of the line is skipped and nothing is written.



VA M Programming Introduction

Flow and Control Commands QUIT and HALT

QUIT

In its simplest form, QUIT will terminate the routine or return you from a subroutine.

QUIT may have an argument. However, for this lesson, we will only address the *argumentless* form of the command.

When you QUIT a routine, you are returned back to the programmer prompt to execute additional programs or routines.

HALT

HALT will terminate the programming session and return you back to your logon screen. This means you must logon again in order to run additional programs or routines.

The HALT command must not be used in application routines. The QUIT command should be used instead.



VA Programming Standards and Conventions

The HALT command must not be used in application routines.

Putting a Routine Together

To develop your first M routine, use the following scenario.

STEP 1: Define the Problem

The Computers-R-Us computer store has offered to sell you a computer system through either an outright purchase or a lease with the option to buy at the end of the lease. If you purchase the system for \$10,000, you can finance the full amount for four years at a 10% interest rate. If you lease the equipment for four years and decide to purchase the system at the end of the lease, the monthly cost will be \$250 with a final payment of \$3,000. Which option will minimize the total cost of the computer system?

STEP 2: Identify the Output

Identify the output to determine what information or data is needed to produce the desired output. In this scenario, the output will be the total purchase price and the total lease purchase price with a decision of which is lower.

The total purchase price is \$xx.xx.

The total lease-purchase price is \$xx.xx.

Which is lower?

STEP 3: Define the Input

For the purchase option, the purchase price, the interest rate, and the length of time of finance are needed.

For the lease purchase option the monthly lease cost, the number of months financed, and the amount of the final payment at the end of the lease are needed.

User will enter:

1. For the purchase option:
 - The purchase price
 - The interest rate
 - The time

VA M Programming Introduction

2. For the lease-purchase option:

- The monthly lease cost
- The number of months
- The final payment

STEP 4: Develop a Method of Solution (Algorithm)

Calculate the total cost of the purchase.

Determine the amount of interest to be paid. Use the simple interest formula, which is the principal purchase amount times the rate of interest times the number of months financed. You derive the total purchase price by adding the purchase price and the interest paid.

- $\text{Simple interest} = \text{principal} * \text{rate} * \text{time}$
- $\text{Total purchase price} = \text{purchase price} + \text{interest}$

Calculate the total cost of a lease-purchase.

Determine the lease cost which is the monthly payment amount times the number of months leased. You derive the total lease purchase cost by adding the lease cost and the final payment.

- $\text{Lease cost} = \text{monthly payment amount} * \text{number of months}$
- $\text{Total lease-purchase cost} = \text{lease cost} + \text{final payment}$

STEP 5: Pseudocode as Programming Tool

Use pseudocode as a programming tool. In other words, write the program code using plain English sentences.

First, we ask the user to enter the input for the purchase option, which is the purchase price, the interest rate, and the length of time financed.

A. ***Input All Data for the Purchase Option***

- Purchase price
- Interest rate
- Time period

VA M Programming Introduction

B. Process

Calculation of the costs.

Calculate the finance cost by taking the purchase price, multiplying it by the interest rate, and then multiplying the result by the length of time financed.

1. Calculate the finance cost of the purchase option:

Finance cost=purchase price times interest rate times time period

Calculate the total purchase cost by taking the purchase price and adding the finance cost.

2. Calculate the total purchase cost:

Total purchase cost=purchase price plus finance cost

These values will have already been entered by the user as asked for in the previous step.

C. Output: Write the Total Purchase Cost

Output or display to the screen the total purchase cost just calculated.

D. Input All Data for the Lease-Purchase Option

Now, for the lease-purchase option. We will ask the user to enter the monthly lease cost, the number of payments, and the lump sum cost at the end of the lease.

- Monthly lease cost
- Number of payments
- Lump sum cost

E. Process

Calculate the total lease cost of the lease-purchase option.

This is calculated by multiplying the monthly lease cost value by the number of payments.

VA M Programming Introduction

1. Calculate the total lease cost of the lease-purchase option:

Total lease cost=monthly lease cost times number of payments

The total lease purchase option cost is calculated by adding the total lease cost and the lump sum payment.

2. Calculate the total lease-purchase cost:

Total lease-purchase option cost=total lease cost plus lump sum cost

F. Output

Display to the screen the total lease purchase cost.

Write the total lease-purchase option cost.

G. IF

We want to make a decision, based on which value is lower, whether we lease it, purchase it, or it doesn't matter, meaning both values are equal. Use the IF statement for this.

- Possibility number one. If the total cost of the purchase option is greater than the total cost of the lease purchase option we will display to the screen LEASE IT.
- Possibility number two. If the total cost of the lease purchase option is greater than the total cost of the purchase option we will display to the screen PURCHASE IT.
- The third possibility is if the total costs for both options are the same, we will display the statement IT DOESN'T MATTER. There are no other possibilities.

1. The total cost of the purchase option is greater than the total cost of the lease-purchase option, write LEASE IT.
2. The total cost of the lease-purchase option is greater than the total cost of the purchase option, write PURCHASE IT.
3. The total costs of both options are the same, write IT DOESN'T MATTER.

VA M Programming Introduction

STEP 6: Convert the Algorithm into a M Routine

Writing the Routine First on Paper

Now convert the pseudocode displayed into M code. Using the information and pseudocode you have developed, it's time to write your M code on paper first. For the practice of writing your routine on paper, you should write the code out as shown here in the manual.

Pseudocode

A. Input all data for the purchase option:

- Purchase price
- Interest rate
- Time period

The pseudocode asks for input of the purchase price, interest rate, and time period for the purchase option.

- The M code begins with a line label, which will be **START**.
- The next line will use a **WRITE** statement to display a heading to the user.
- The next line uses a **READ** statement to display a description of the data required and stores the value entered in the variable **PURPRICE**.
- The next line uses a **READ** statement to display a description of the data needed, the interest rate, and stores the value in the variable **INT**.
- The last line uses a **READ** statement to display a prompt to the user to enter the time period and stores the value in the variable **TIME**.

M Code

```
START;  
  WRITE !,"THE PURCHASE OPTION"  
  READ !!,"ENTER PURCHASE PRICE: ",PURPRICE  
  READ !,"ENTER INTEREST RATE: ",INT  
  READ !,"ENTER TIME PERIOD: ",TIME
```

VA M Programming Introduction

Pseudocode

B. Process

1. Calculate the finance cost of the purchase option:

Finance cost=purchase price times interest rate times time period

2. Calculate the total purchase cost:

Total purchase cost=purchase price plus finance cost

For processing, calculate the finance cost using the formula: purchase price, times interest rate, times time period. Subsequently, calculate the total purchase cost by adding the purchase price and the finance cost together.

- M code uses a SET statement to assign a value to the variables FINCOST and PURCOST. The total purchase cost is now stored in the variable PURCOST.

M Code

```
SET FINCOST=PURPRICE*INT*TIME
SET PURCOST=PURPRICE+FINCOST
```

Pseudocode

C. Output: write the total purchase cost

Display the total purchase cost to the screen.

The WRITE statement is used to display a description of the data and the value stored in the variable PURCOST, which is the total purchase cost.

So far, the M code has prompted the user for the required data, performed calculations, stored the total purchase cost as a variable, and displayed the value to the user.

Remember the exclamation mark means the cursor will move to the next line. In this case the cursor moves down two lines.

M Code

```
WRITE !!, "THE TOTAL PURCHASE COST IS $", PURCOST
```

VA M Programming Introduction

Pseudocode

D. Input all data for the lease-purchase option:

- Monthly lease cost
- Number of payments
- Lump sum cost

Prompt the user for the required data for the lease purchase option. The data required is the monthly lease cost, the number of payments, and the lump sum payment.

- Begin with a new line label LEASE.
- Use a WRITE statement to display a heading.
- Use READ statements to display a descriptive prompts to the user indicating data required.

M Code

```
LEASE ;  
    WRITE !!, "THE LEASE-PURCHASE OPTION"  
    READ !, "ENTER MONTHLY LEASE COST: ", MONCOST  
    READ !, "ENTER NUMBER OF PAYMENTS: ", NRPAYM  
    READ !, "ENTER LUMP SUM: ", LUMP
```

Pseudocode

E. Process

1. Calculate the total lease cost of the lease-purchase option:

Total lease cost=monthly lease cost times number of payments

2. Calculate the total lease-purchase cost:

Total lease-purchase option cost=total lease cost plus lump sum cost

Calculate the total lease purchase cost.

The SET statement is used to assign values to variables based on calculations.

The variable TLEASPUR stores the value of the Total Lease Purchase Cost.

M Code

```
SET TLEASE=MONCOST*NRPAYM  
SET TLEASPUR=TLEASE+LUMP
```

VA M Programming Introduction

Pseudocode

F. Output: Write the total lease-purchase option cost.

Use the WRITE statement to display a descriptive statement and the value of the variable TLEASPUR, which is the total lease purchase cost.

M Code

```
WRITE !!, "THE TOTAL COST OF LEASE IS $", TLEASPUR
```

Pseudocode

G. IF

1. The total cost of the purchase option is greater than the total cost of the lease-purchase option, write LEASE IT.

IF statements will be used to determine a true value.

Determine if the value stored in the variable PURCOST is greater than the value stored in the variable TLEASPUR. If this is a true statement, the statement LEASE IT will be displayed to the screen.

The line label DECISION is used for this segment of the code.

M Code

```
DECISION ;  
    IF PURCOST>TLEASPUR WRITE !!, "LEASE IT"
```

Pseudocode

G. IF

2. The total cost of the lease-purchase option is greater than the total cost of the purchase option, write PURCHASE IT.

Determine if the value stored in the variable TLEASPUR is greater than the value stored in the variable PURCOST. If this is a true statement, the statement PURCHASE IT will be displayed to the screen.

M Code

```
IF TLEASPUR>PURCOST WRITE !!, "PURCHASE IT"
```


VA M Programming Introduction

Pseudocode

G. IF

3. The total costs of both options are the same, write IT DOESN'T MATTER.

Determine if the value stored in the variable TLEASPUR is equal to the value stored in the variable PURCOST. If this is a true statement, the statement IT DOESN'T MATTER will be displayed to the screen.

Also added to the M code is the line label EXIT and a QUIT statement to stop processing.

M Code

```
        IF TLEASPUR=PURCOST WRITE !!,"IT DOESN'T MATTER"
EXIT    ;
        QUIT
```

The Complete M Routine

```
START    ;
        WRITE !!,"THE PURCHASE OPTION"
        READ !!,"ENTER PURCHASE PRICE: ",PURPRICE
        READ !!,"ENTER INTEREST RATE: ",INT
        READ !!,"ENTER TIME PERIOD: ",TIME
        SET FINCOST=PURPRICE*INT*TIME
        SET PURCOST=PURPRICE+FINCOST
        WRITE !!,"THE TOTAL PURCHASE COST IS $",PURCOST
LEASE    ;
        WRITE !!,"THE LEASE-PURCHASE OPTION"
        READ !!,"ENTER MONTHLY LEASE COST: ",MONCOST
        READ !!,"ENTER NUMBER OF PAYMENTS: ",NRPAYM
        READ !!,"ENTER LUMP SUM: ",LUMP
        SET TLEASE=MONCOST*NRPAYM
        SET TLEASPUR=TLEASE+LUMP
        WRITE !!,"THE TOTAL COST OF LEASE-PURCHASE IS $",TLEASPUR
DECISION ;
        IF PURCOST>TLEASPUR WRITE !!,"LEASE IT"
        IF TLEASPUR>PURCOST WRITE !!,"PURCHASE IT"
        IF TLEASPUR=PURCOST WRITE !!,"IT DOESN'T MATTER"
EXIT     ;
        QUIT
```

VA M Programming

Introduction

Entering the Routine into the Computer

M Direct and Deferred Modes

There are two modes of operation:

Direct mode (also called programmer mode): is a command that executes immediately

When working in direct mode, the user enters a command followed by arguments and <ENTER>, with the command being immediately executed. The commands are executed line by line as they are typed.

Deferred mode: is where the first command on a line is preceded by <TAB> (when you see the <TAB> symbol, press the Tab key; statements are stored in a workspace until they're executed as a complete routine.

In deferred mode, the user enters a line label followed by the Tab key or the line begins with the Tab key. Statements or lines of code are typed but not immediately executed. The statements are stored in a workspace in memory until they are executed as a complete routine by issuing the DO or GO command.

M Direct Mode

The M commands are executed as each line is entered.

In the example below:

The first command is SET A equals ten and then press <ENTER>.

The next command is SET B equals five and then press <ENTER>.

The next command is WRITE A star B and press <ENTER>.

The star, or the astrisk (*) means multiply. We are multiplying the value of the variable A and the value of the variable B, and writing the result to the display screen.

The result is immediately displayed after the WRITE command is typed.

M Code Sample

```
SET A=10  
SET B=5  
WRITE A*B
```

VA M Programming Introduction

To fix errors before you press <ENTER>, use the backspace key.

To fix errors after you press <ENTER>, just retype the line.

M Deferred Mode

The code, in deferred mode, is not executed and statements in quotes are not displayed to the screen. The code is being stored in memory for execution later. No result is displayed to the screen at this time. Each line of code begins with a Tab key for deferred mode.

As in the previous example, the star or asterisk (*) means multiply.

M Code Sample

```
<TAB>READ !,"Enter first value: ",A<ENTER>  
<TAB>READ !,"Enter second value: ",B<ENTER>  
<TAB>WRITE !,"The answer is ",A*B<ENTER>
```

To fix errors before you press <ENTER>, use the backspace key.

To fix errors after you press <ENTER>, use an editor.

Line Labels

A line label is a way of identifying a line and sections of M code in a routine that is deferred.

Rules for naming line labels (also called tags) are:

1. Labels can start with an alphabetic character (uppercase or lowercase), a numeric digit (0-9), or the % (percent) character.
2. The first character can be followed by any number of alpha or numeric characters (no punctuation); however, only the first eight are used to identify a label.
3. If the first character is numeric, however, the remaining characters must be numeric.
4. Labels are case sensitive; i.e., the label START is different than Start.

The first line of every routine should have a line label. A suggestion for using line labels is that you have a label at least every 10 lines or so.

VA M Programming Introduction

Valid Line Label Names

START	PRINT	TOTAL1
010	100	305
Start (different from START)		

Invalid Line Label Names

1A	DISC%	TOTAL-PAY
NET PAY	PRICE/TAPE	



VA Programming Standards and Conventions

Line labels must be all uppercase characters.

Line labels are limited to eight characters and should be meaningful as mnemonics.

Offsets

Since every line does not have to have a label, there needs to be a way of identifying lines without labels. Each line, with or without a label, has a line reference which is made up of the name of the nearest labeled line plus an indication of the line's relative distance from that labeled line. This indicator is called an offset. A line with a label has an offset of +0; each line is numbered from the line label down until you reach the next labeled line.

Sample M Code

```
PROG1<TAB>;  
<TAB>WRITE !, "LINE 1"  
<TAB>WRITE !, "LINE 2"  
INPUT<TAB>;  
<TAB>READ !, "VAR 1", X  
<TAB>READ !, "VAR 2", Y  
<TAB>READ !, "VAR 3", Z  
EXIT<TAB>;  
<TAB>WRITE !, "BYE"  
<TAB>QUIT
```

Line Reference

(Line label+offset)

```
PROG1+0  
PROG1+1  
PROG1+2  
INPUT+0  
INPUT+1  
INPUT+2  
INPUT+3  
EXIT+0  
EXIT+1  
EXIT+2
```

Note: The semicolon (;) on the labeled line is a comment "command" which will be discussed in a section that follows.

VA M Programming Introduction

ZREMOVE

As you begin entering a routine, the first step you need to do is to issue the ZREMOVE command, abbreviated as ZR. This command clears the workspace of any previous routines already stored in the workspace (or memory).

After clearing the workspace you are now ready to enter the M code.

M Code Sample

```
>ZR<ENTER>
```

We use the ZR (ZREMOVE) command to clear our workspace before we begin to type our routine.

```
>START<TAB>;<ENTER>
```

```
><TAB>READ !!,"ENTER PURCHASE PRICE: ",PURPRICE<ENTER>
```

```
><TAB>READ !,"ENTER INTEREST RATE: ",INT<ENTER>
```

```
><TAB>READ !,"ENTER TIME PERIOD: ",TIME<ENTER>
```

```
><TAB>SET FINCOST=PURPRICE*INT*TIME<ENTER>
```

```
><TAB>SET PURCOST=PURPRICE*FINCOST<ENTER>
```

```
><TAB>WRITE !!,"THE TOTAL PURCHASE COST IS $",PURCOST<ENTER>
```

ZPRINT

We use the ZP (ZPRINT) command to print the contents of our workspace.

M Code Sample

```
>ZP<ENTER>
```

```
START      ;
          READ !!,"ENTER PURCHASE PRICE: ",PURPRICE
          READ !,"ENTER INTEREST RATE: ",INT
          READ !,"ENTER TIME PERIOD: ",TIME
          SET FINCOST=PURPRICE*INT*TIME
          SET PURCOST=PURPRICE*FINCOST
          WRITE !!,"THE TOTAL PURCHASE COST IS $",PURCOST
```

VA M Programming Introduction

ZSAVE

M Code Sample

```
>ZS routine-name<ENTER>
```

Use the ZS (ZSAVE) command to save lines of code currently in the workspace to the disk. After the ZS, type a space, then give the routine a unique name under which it will be stored. For this course, we will use the following guidelines:

MCD1nn for this sample routine
where: 1 is the lesson number
 nn is number of our routine

For example, the namespace would be MCD101 for this first M routine.

Routine names can be 8 characters or less.



VA Programming Standards and Conventions

Routine names be all uppercase alphabetic/numeric characters and begin with an alphabetic character. You will be assigned a namespace (a way of organizing and identifying your routines), which you must use consistently.

Editing Your Routine

If there are corrections or additions to be made, we must use a specialized M routine called an ***editor*** to make changes to a routine, which is stored on disk or in the current workspace. There are several different editors available which are specific to the operating system being used. Consult your operating systems manuals for instructions on using the editor available to you.

VA M Programming Introduction

Summary of DSM-11 Editing

M commands used in editing begin with the letter Z and are called implementation-specific commands.

ZR (or ZREMOVE)	To clear your workspace before you begin.
ZL<space>routine-name (or ZLOAD)	To load a routine stored previously.
ZS<space>routine name (or ZSAVE)	To store a routine located in your workspace.
ZP (or ZPRINT)	To print the routine located in your workspace.

Test and Debug the Routine

STEP 7: Test and Debug the Routine

To actually run the routine, we use the DO command followed by the routine's first line label.

M Code Sample

```
>DO START<ENTER>
```

```
THE PURCHASE OPTION
```

```
ENTER PURCHASE PRICE: 10000  
ENTER INTEREST RATE: .10  
ENTER TIME PERIOD: 4
```

What the user entered in response to the routine is double-underlined.

If you see wrong results or errors when running the routine, use the editor to edit the routine and test again.

VA M Programming Introduction

More M Commands

Comments

There are two primary uses for comments.

One is documenting a routine with additional information. Document a routine with such things as a variable list which, shows all the variable names used with a brief description of the use of that variable. The variable list is for you, the programmer; M ignores it.

The second is that it is a good technique to make every line label a comment line. This will make it easier to add new lines at the beginning of labeled sections and making every line label a comment line will improve readability.



VA Programming Standards and Conventions

The first line of every routine should contain:

Routine name<TAB>; site/programmer-brief description<space>; date/time

The second line should contain the version number in the format:

<TAB>; nn;package name;patch list;version date
(where nn is the version number)



VA M Programming

Introduction

Example of a routine using Comments

Here is the complete sample routine. Note the use of comment lines for the first and second line, the variable list, and lines with line labels: INPUT, LEASE, DECISION, EXIT.

M Code Sample

```
MCD101      ;ATL/ACE PROGRAMMER-COMPARISON PROGRAM ;1-1-90
            ;;1
            ;    VARIABLE LIST
            ;
            ;
            ;    PURPRICE      =    PURCHASE PRICE (from input)
            ;    INT           =    INTEREST RATE (from input)
            ;    TIME          =    TIME PERIOD (from input)
            ;    FINCOST       =    FINANCE COST (calculated)
            ;    PURCOST       =    PURCHASE COST (calculated)
            ;    MONCOST       =    MONTHLY LEASE COST (from input)
            ;    NRPAYM        =    NUMBER OF PAYMENTS (from input)
            ;    LUMP          =    LUMP SUM PAYMENT (from input)
            ;    TLEASE        =    TOTAL LEASE PAYMENTS (calculated)
            ;    TLEASPUR      =    TOTAL LEASE-PURCHASE COST (calc.)
INPUT      ;
            WRITE !,"THE PURCHASE OPTION"
            READ !,"ENTER PURCHASE PRICE: ",PURPRICE
            READ !,"ENTER INTEREST RATE: ",INT
            READ !,"ENTER TIME PERIOD: ",TIME
            SET FINCOST=PURPRICE*INT*TIME
            SET PURCOST=PURPRICE+FINCOST
            WRITE !,"THE TOTAL PURCHASE COST IS $",PURCOST
LEASE      ;
            WRITE !,"THE LEASE-PURCHASE OPTION"
            READ !,"ENTER MONTHLY LEASE COST: ",MONCOST
            READ !,"ENTER NUMBER OF PAYMENTS: ",NRPAYM
            READ !,"ENTER LUMP SUM: ",LUMP
            SET TLEASE=MONCOST*NRPAYM
            SET TLEASPUR=TLEASE+LUMP
            WRITE !,"THE TOTAL COST OF LEASE IS $",TLEASPUR
DECISION   ;
            IF PURCOST>TLEASPUR WRITE !,"LEASE IT"
            IF TLEASPUR>PURCOST WRITE !,"PURCHASE IT"
            IF TLEASPUR=PURCOST WRITE !,"IT DOESN'T MATTER"
EXIT      ;
            QUIT
```

VA M Programming

Introduction

Flow Control Command GOTO

GOTO command redirects execution of the routine to another line label.

The GOTO command is used to allow the programmer to redirect M from its normal flow of execution. The syntax is the GOTO command and a line label. Execution of the routine is redirected from the next sequential line to another line or routine using the GOTO command. In the example below, the execution of the routine is redirected to the line label EXIT.

M Code Sample

```
GOTO EXIT
```



VA Programming Standards and Conventions

1. Don't use GOTO line label+offset.
2. When a user is asked a question like:
DO YOU WANT TO DO ANOTHER? YES//

The default (most common) response is shown followed by //
The null response (i.e., typing only <ENTER>) shall select the default.



Shown below is an example using the GOTO command to redirect execution of the routine to the line label INPUT if the value of the variable ANSWER is equal to null, if the user entered <ENTER> without an entry, or if the variable ANSWER is equal to YES in uppercase, or if the variable ANSWER is equal to an uppercase Y. If none of these are true, the execution of the routine continues to the next line of code, which is EXIT, and so on.

VA M Programming

Introduction

M Code Sample

```
MCD102 ;ATL/ACE PROGRAMMER-COMPARISON PROGRAM ;1-1-90
      ;;1
      ;    VARIABLE LIST
      .
      .
      .
      ;    TLEASPUR      =      TOTAL LEASE-PURCHASE COST (calc.)
INPUT  ;
      WRITE !,"THE PURCHASE OPTION"
      .
      .
      .
CHECK  ;
      READ !,"DO YOU WANT TO DO ANOTHER? YES//",ANSWER
      IF ANSWER="" GOTO INPUT
      IF ANSWER="YES" GOTO INPUT
      IF ANSWER="Y" GOTO INPUT
EXIT   ;
      QUIT
```

Assignment Command KILL

The purpose of the **KILL** command is to get rid of variable names and their values.

All variables created by your program should be killed before exiting the routine. The syntax of the KILL command is: KILL <space> and variables, separated by a comma if there is more than one.

The exclusive KILL command is used to kill all variables except those listed within the parenthesis.

If the KILL command is used without any variables listed, it is called an argumentless kill. This KILLs all variables. If it is followed by another command on the same line, there must be two spaces after the command.

VA Programming Conventions prohibit the use of the exclusive and argumentless forms of the KILL command. The argumentless KILL command is prohibited because some system variables would be killed as well as all others.

VA M Programming

Introduction

M Code Sample

KILL A
KILL A,B
KILL (C) This is called the exclusive KILL.. This KILLs all variables except C.
KILL This is called the argumentless KILL. This KILLs all variables.
If it is followed by another command on the same line, there must be two spaces after it.



VA Programming Standards and Conventions

Do not use the exclusive KILL or argumentless KILL.
All variables used should be killed before exiting the routine.

Here is an example using the KILL command to get rid of the variables listed and their values. In this example, the variables and their values used in the routine are gotten rid of or killed. Notice, two lines were used with each line beginning with the command KILL.

M Code Sample

```
MCD103      ;ATL/ACE PROGRAMMER-COMPARISON PROGRAM ;1-1-90
             ;;1
             ; VARIABLE LIST
             ;
             ; PURPRICE = PURCHASE PRICE (from input)
             ; INT      = INTEREST RATE (from input)
             .
             .
             .
CHECK        ;
             READ !,"DO YOU WANT TO DO ANOTHER? YES//",ANSWER
             IF ANSWER="" GOTO INPUT
             IF ANSWER="YES" GOTO INPUT
             IF ANSWER="Y" GOTO INPUT
EXIT        ;
             KILL PURPRICE,INT,TIME,FINCOST,PURCOST,MONCOST,NRPAYM
             KILL LUMP,TLEASE,TLEASPUR,ANSWER
             QUIT
```

VA M Programming Introduction

Running Your Routine

Here's how the routine should work when it is run. The user's response to the prompts is shown double underlined.

>DO ^MCD101

THE PURCHASE OPTION

ENTER PURCHASE PRICE: 10000

ENTER INTEREST RATE: .10

ENTER TIME PERIOD: 4

THE TOTAL PURCHASE COST IS \$14000

THE LEASE-PURCHASE OPTION

ENTER MONTHLY LEASE COST: 250

ENTER NUMBER OF PAYMENTS: 48

ENTER LUMP SUM: 3000

THE TOTAL COST OF LEASE PURCHASE IS \$15000

PURCHASE IT

DO YOU WANT TO DO ANOTHER? YES//<ENTER>

THE PURCHASE OPTION

ENTER PURCHASE PRICE: 9000

ENTER INTEREST RATE: .12

ENTER TIME PERIOD: 5

THE TOTAL PURCHASE COST IS \$14400

THE LEASE-PURCHASE OPTION

ENTER MONTHLY LEASE COST: 200

ENTER NUMBER OF PAYMENTS: 36

ENTER LUMP SUM: 2000

THE TOTAL COST OF LEASE PURCHASE IS \$9200

LEASE IT

DO YOU WANT TO DO ANOTHER? YES//<NO>

>

VA M Programming Introduction

Assignment (Optional)

In General

Follow the VA Programming Conventions as illustrated in this lesson – make sure that:

1. The first line label is the same as the routine name under which the routine is saved.
 2. The first line of your routine contains your name and site
 3. The second line contains the version number or other identifying information as required by your site for storage of routines in a test environment.
- Use only the techniques, commands, and features described in this lesson.
 - Talk to your IRM or IT chief about saving the routines you create under a namespace assigned to you in a test account or environment.
 - Include a variable list.
 - Do not use wrap-around lines.
 - Have only one command per line (except with the IF command, of course).
 - Do not use command abbreviations.

Assignment Specifications

Using the sample routine from page 45 (MCD103), write a comparison shopping routine. The user will be pricing the same item in two different stores. For Store 1, have the user enter the retail price for that item and the item discount percentage (in a decimal fraction, e.g. .10, .15). For Store 2, have the user enter the retail price for that item and the item discount percentage (in a decimal fraction, e.g. .10, .15). Calculate and display the discounted prices for each store:

discounted price=retail price (retail price*discount percentage)

Based on the two discounted prices, display advice to the user as to the best buy (e.g., "Buy from Store 1", "Buy from Store 2", "It doesn't matter"). Ask the user if he/she wants to do another comparison. If so, start the routine over again. If not, stop the routine.

VA M Programming Introduction

Optional Element

Have the user enter the store name for each store. When it comes time to display the best buy, display the store name where you can find this bargain.