TABLE OF CONTENTS (Continued)

Section Title Page
6.13 Input words 8-69
6.13.1 Input a Character from the Keyboard 8-69
6.13.2 Input a String from the Keyboard 8-69
6.13.3 Get the Alphabetic Input Device with the IT 8-69
6.13.4 Input a Character from the Alphabetic Input Device 8-69
6.13.5 Input a String from the Alphabetic Input Device with the IT 8-69
6.13.6 Test for terminal input with 'TESTINPUT' 8-69

5 Advanced Operations
5.1 Other Simple-Expression Arithmetic operations 5-1
5.1.1 Boolean operators 'AND' and 'XOR' 5-1
5.1.2 Absolute 'ABS' and Round 'ROUND' 5-1
5.1.3 Increment and Decrement '1', '++', '--', '+++', '++++' 5-2
5.1.4 Minimum 'MIN' and Maximum 'MAX' 5-2

5.2 Complex, Mixed and Multiple-Expression Arithmetic 5-3
5.2.1 Ignoring special-assignment numbers 5-3
5.2.2 Ignoring special-assignment words 5-3
5.2.3 Ignoring special-assignment phrases 5-3
5.2.4 Ignoring special-assignment blocks 5-3
5.2.5 Ignoring multiply in and divide in 5-3
5.2.6 Ignoring multiply in and divide in 5-3
5.2.7 Bolding 'BOLD' 5-3

5.3 Output Formatting 5-4
5.3.1 Output character 'A', 'B', 'C', 'D', 'E', 'F' 5-4
5.3.2 Double and single field output 5-4

4 Strings
4.1 Addressing string data with 'CHRS' 4-23
4.2 Output string data with 'PRINT' 4-23
4.3 Output string data with 'PRINTF' 4-23
4.4 Output string data with 'PRINTF' 4-23

5.5 Arithmetic Structure 5-5
5.5.1 Form Main Structure 5-5
5.5.2 Handling FORM word indentation 5-5
5.5.3 FORM word handling examples 5-5

iv

TABLE OF CONTENTS (Continued)

Section Title Page
8 Reading and Writing to FORM 8-1
8.1 Types of Interfacing Signals 8-1
8.2 Machine level interfacet reading 8-1
8.2.1 INPUT Definition Data 8-1
8.2.2 Code Propagation Data 8-1
8.2.3 Interruptable Data/Enable Words 8-1
8.2.4 Example 8-1

7.3 Interpreters Interface Model 7-5
7.3.1 Microform Source Generation 7-5
7.3.2 Microform Source Word Processing 7-5
7.3.3 Example 7-5
8 Understanding the DATA I/O 8-1
8.1 Variable Organization and Register 8-1
8.2 Stamp 8-18 and the VLA 8-18
8.2.1 Data Consideration 8-18
8.2.2 Example 8-18
8.3 Translating source symbols 8-18
8.3.1 Conversion Table 8-18
8.3.2 Examples 8-18
8.4 Producing machine programs 8-18
8.4.1 Modifications 8-18
8.4.2 Options 8-18
8.4.3 Examples 8-18
8.4.4 Consideration 8-18
8.4.5 Consideration 8-18

7 VLA and CM Operation 7-1
7.1 CM Operation 7-1
7.1.1 Switching control to the CM 7-1
7.1.2 Executing Control in the CM 7-1
9 VLA Internal operation using the VP/AM Current 9-1
9.1 VLA Internal operation using the VP/AM Current 9-1
9.2 Switching control to the CM 9-1
9.3 Switching control to the CM 9-1

v

TABLE OF CONTENTS (Continued)

Section Title Page
6.4 Verbs 6-54
6.4.1 معظم الاعمال 6-54
6.4.2 اعمال 6-54
6.4.3 Additional Instructions 6-54
6.4.4 Specific Instructions 6-54
6.4.5 Creating your own data/operation types 6-54

5 ARE and FORTRAN Advantage 5-1
5.1 The Assembly Process 5-1
5.1.1 COSE Definitions 5-1
5.1.2 Assembly Example Routines 5-1
5.2 Assembly Operators 5-1
5.2.1 Single Mode Op-Codes 5-1
5.2.2 Multi Mode Op-Codes 5-1
5.3 Addressing Modes 5-1
5.4 Memory Connections 5-1
5.5 Stack Addressing 5-1
5.6 Memory Data 5-1
5.7 Memory Stack 5-1
5.8 Memory Array 5-1

6.5Fortran 6-11
6.5.1 Array Declaration 6-11
6.5.2 Data Declaration 6-11
6.5.3 DO Statement 6-11
6.5.4 IF Statement 6-11
6.5.5 Case Statement 6-11
6.5.6 READ Statement 6-11
6.5.7 WRITE Statement 6-11
6.5.8 READ List Statement 6-11
6.5.9 WRITE List Statement 6-11
6.6 I/O Control 6-11
6.6.1 READ Statement 6-11
6.6.2 WRITE Statement 6-11
6.6.3 OPEN Statement 6-11
6.6.4 CLOSE Statement 6-11
6.6.5 READ List Statement 6-11
6.6.6 WRITE List Statement 6-11
6.6.7 Flush statement 6-11
6.6.8 Set Format 6-11
6.6.9 Get Format 6-11

6.4.1 Variable Initialization 6-54
6.4.2 Data Operation 6-54
6.4.3 Common Instructions 6-54
6.4.4 Specific Instructions 6-54
6.4.5 Creating your own data/operation types 6-54

6.3 Writing Assembly Code from Visual-Machine 6-21

vii
CHAPTER 1

INTRODUCTION

FTM is a unique programming system that is well suited to a variety of applications. Because it was originally developed for real-time control applications, FTM has features that make it ideal for monitoring and process control, data acquisition, energy and environmental monitoring, automatic testing, and other similar applications. The speed performance of assembly language is required in many of those applications, however, a high-level language is often desired to improve 
maintenance development productivity and program reliability. 

GMN is designed to satisfy both speed and screening 

efficiency requirements.

GMN can be called a computer language, an operating system, an interactive compiler, a data structure, or an interpreter, depending upon your point of view. It was designed to provide the advantages of both compilers and interpreters. The result is a unique language based on pre-defined operations that instructs software development tasks and creates, compiles, structured programming and extended modularity, compile

independently of the machine language and to reduce pre-programming

work, compiled into machine code, and executed extremely fast. Additional speed may be obtained by

Nevertheless, N

FMT 85 FORM LN CL

contains the benefits of FMT and the

features of the FMT 85 Microcomputer with its resident processor, display, keyboard, and interactive Monitor and Test Editor

clutters, to produce a stable development and run-time system.
1.1 AIM 65 FORUM USER'S MANUAL DESCRIPTION

This manual is designed to provide both introductory instruction and detail language reference information. If you are new to FORUM, be sure to read and follow the manual chapter-by-chapter until you are familiar with the language and operations of the FORUM system. If you already know the FORUM language you can probably skip certain sections and still use the language, however it is recommended to review all sections to become familiar with the AIM 65 FORUM mechanization and unique features.

Section 1. Introduction, introduces the AIM 45 FORUM language and the AIM 65 FORUM User's Manual.

Section 2. Installation and Operation, explains how to install the AIM 65 FORUM ROM and use it to access, edit and recover AIM 65 FORUM.

Section 3. FORUM Concepts, provides a general overview into FORUM concepts and advantages. This is a good chapter to read if you are new to FORUM.

Section 4. Elementary Operations, leads you through elementary and complex FORUM operations. By following this section step-by-step you will learn how FORUM operates at a sufficient level to implement simple applications in FORUM.

Section 5. Advanced Operations, takes you into more complex FORUM operations once you have become familiar with the elementary FORUM operations described in Section 4.

Section 6. AIM 65 FORUM Assembler, describes concepts and operating procedures associated with the AIM 65 FORUM Assembler.

Section 7. Handling Interrupts in FORUM, explains how to use a machine level and interruptive interrupts in FORUM.

Appendix A. User Turn and Use Memory Map, defines the address, variable name and general scope of page into parameters.

Appendix B. User Variable ROM Map, defines the address, variable name and purpose of each user variable. The code and word data initialization values are also listed.

Appendix C. ASCII Character Set, provides a list of 7-bit ASCII codes corresponding to the 96 upper and lower case alphabetic, numeric, and special characters along with their decimal and hexadecimal equivalents.

Appendix D. USER кино Development Notes, describes how to create utility handling functions in FORUM.

Appendix E. User Language Consultant Program in FORUM, illustrates a program written in FORUM color and FORUM instructions, i.e., FORUM high-level words and ROM assembly language.

Appendix F. Ascending FORUS 65 Handwriting, explains how to determine the time it takes for a FORUS 65 hand to write.

Appendix G. AIM 65 FORUM User's Guide, lists a program written in FORUS 65 to compute and display a FORUS 65 checkbook.

Appendix H. AIM 65 FORUS Version 10 FORUS, identifies words incorporated in each FORUS 65 FORUS that are not included in the other FORUS.

Appendix I. Selected Bibliography, lists references to many popular and important FORUS articles and books.

1.2 REFERENCE SOURCES

Rockwell

269535R

USER Programming Manual

269653R

USER HARDWARE Manual

269653R

AIM 65 User's Guide

Section 5, Programming the FORUS 65, explains how to use FORUS to program the memory FORUS (Virtual Interface Adapter (VIA)). These techniques can easily be applied to other peripheral devices.

Section 6, VGA and CRT Operation, describes how to switch control between the AIM 65 and a TV or CRT terminal.

Section 7, WRITING AND CRT Operation, explains how to structure and locate a FORUS application program in a FORUS which will operate in conjunction with the AIM 65 FORUS.

Section 8, Using an Audio Cassette Recorder, describes the FORUS and users operating procedures for program written in FORUS.

Section 9, Interfacing to Mass Storage, tells how to preprogram programs to store and retrieve program data from mass storage. Blocks, sectors, and buffers are described. The technique to handle program overlays is also explained.

Section 10, Implementation of Program Development, discusses the general approach to programming in FORUS and provides an example program.

Appendix A, AIM 65 FORUS Functional Summary, summarizes FORUS word operation by general area of scope.

Appendix B, AIM 65 FORUS Glossary, defines each FORUS word in ASCII sort order.

Appendix C, FORUS ASSEMBLER Functional Summary, summarizes FORUS assembler word operation by area of scope.

Appendix D, AIM 65 FORUS Assembler Glossary, defines each FORUS Assembler word in ASCII sort order.

Appendix E, Error Messages and Recovery, identifies each FORUS error number and/or message, defines the error meaning, and describes the recovery action.

1-3

SECTION 3

INSTALLATION AND OPERATION

The AIM 65 FORUS object code is provided on the Rockwell 32515 floppy diskette ROM device. After installing the ROM in AIM 65, FORUS is ready for use. (Figure 2-1 shows the overall FORUS memory map.)

2.1 INSTALLING THE FORUS ROMS

Before removing the ROMS from the shipping package, be sure to observe the handling precautions listed in Section 5.4 of the AIM 65 User's Guide. Since ROM devices may be damaged by the static electrical charge accumulated on your body by touching a grounded object (e.g., a grounded wall outlet) before touching the ROMS or the AIM 65, this precaution is especially important if you are working in a crowded area or in an environment with low relative humidity.

Ensure that power is turned off to the AIM 65. Carefully remove any ROMS or ROM devices that may be installed in sockets 22S and 24S of the AIM 65 Master Module. Remove the FORUS ROMS from the shipping package. Insert the ROMS to ensure the pins are straight and free of foreign material. While supporting the AIM 65 Master Module beneath the ROM socket, insert ROM number 3222S in socket 22S, being careful to observe the device orientation. Now insert ROM number 3222S into socket 24S. Be certain that both ROMS are completely inserted into their sockets, then turn on power to the AIM 65.

CAUTION

If expansion memory is installed off-board the AIM 65 Master Module, be sure that addresses assigned with FORUS and GPM are not selected on the off-board memory.
2.2 ENTERING, EXITING AND RE-ENTERING FORTH

2.2.1 Entering FORTH

Press 5 to enter and initialize FORTH when the AIM 65 Monitor prompt is displayed. AIM 65 will respond with

```
c
AIM 65 FORTH VI.1
```

To re-initialize FORTH while in FORTH, type

```
cOLD
```

followed by pressing the (return) key. AIM 65 will respond with

```
cOLD
AIM 65 FORTH VI.1
```

Initializing FORTH with either of the above methods will remove any user words previously defined and added to the FORTH vocabulary or to any other application vocabulary (see Section 4.5). User variables are initialized to the default values described in Appendix F. The FORTH number base is also initialized to DECIMAL for input/output operations.

2.2.2 Exiting FORTH

Two methods can be used to exit FORTH. The ESC key can be pressed any time FORTH is in a command input mode. Control will be immediately returned to the AIM 65 Monitor, however any values currently in the stack will still be saved. The significance of this will become apparent as you become more familiar with FORTH.

Control can also be returned to the AIM 65 Monitor from the FORTH command input mode by typing

```
NEW
```

followed by pressing the (return) key. This causes an EDITED NEW machine instruction to be executed and AIM 65 to display

```
NEW CTRB 4C JMP BPIA
```

More importantly, exiting FORTH in this manner preserves any values on the stack, it also sets the Program Counter to 0000 for easy re-entry.

2.2.3 Re-Entering FORTH

Once FORTH has been entered and control returned to the AIM 65 Monitor, you can re-enter FORTH by either of two methods without re-initializing the user variables or clearing previously defined words.

You can re-enter FORTH by pressing a space between the AIM 65 monitor prompts in displayed. AIM 65 will respond with

```
c
AIM 65 FORTH VI.1
```

Note that re-entering FORTH with the S key will delete any values previously stored in the stack, however the Y0 number base is retained (see Section 4.1.3).

If FORTH has been exited using the NEW command, FORTH can be re-entered by typing S followed by pressing the (space) bar or the . (period) key. This can usually be done without changing the program counter; since the FORTH NEW command sets the Program Counter to 0000 before exiting FORTH (see Section 2.2.2).