

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

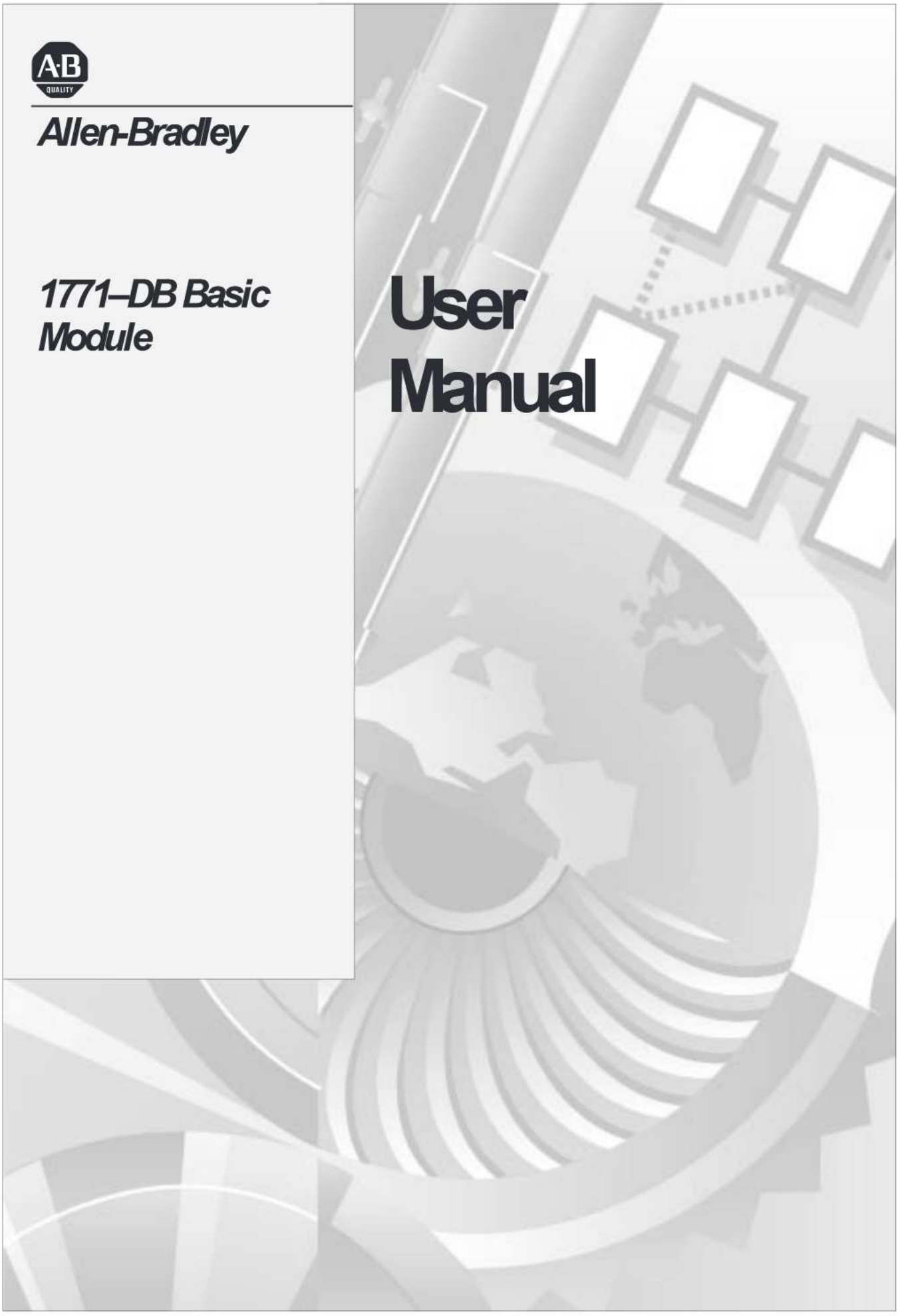
Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Rockwell / Allen-Bradley 1771-DB* at our website: ***Click HERE***



Allen-Bradley

1771–DB Basic Module

User Manual

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Using This Manual 1

1.1 Chapter Objectives 1
1.2 What this manual contains 1
1.3 Audience 1
1.4 Definitions of major terms 1
1.5 Important information 2
1.6 Conventions 2

Introducing the BASIC Module 1

2.1 Chapter Objectives 1
2.2 General Features 1
2.3 Hardware Features 2
2.4 Software Features 3
2.5 Specifications 4

Installing the BASIC Module 1

3.1 Chapter Objectives 1
3.2 Installing the BASIC module 1
3.2.1 Power Requirements 2
3.2.2 Module Location in the I/O Chassis 2
3.2.3 Module Keying 2
3.2.4 Configuration Plugs 3
3.2.5 Module Installation 5
3.2.6 Initial Start-up Procedure 5
3.3 Module Status LED's 6
3.4 Installing the User Prom 7
3.4.1 Electrostatic Discharge 9
3.5 Battery 9
3.5 Battery (continued) 10

Using the Serial Ports I–1

4.1 Chapter Objectives I–1
4.2 Using the BASIC Module Program and Peripheral Communication Ports I–1
4.2.1 Pin Descriptions I–3
4.3 Program Port I–5
4.3 Program Port (continued) I–6
4.3.1 Using the XOFF/XON Commands for the Program Port I–7
4.3.2 Connecting a T3/T4 Industrial Terminal to the Program Port . I–8
4.3.3 Connecting a T30 Industrial Terminal (Cat. No. 1784-T30) to the Program Port I–9

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.4 Connecting a T50 Industrial Terminal (Cat. No. 1784-T50) to the Program Port I–9
4.3.4.1 Configuring the Software I–10
4.3.4.2 Wiring I–10
4.4 Peripheral Port I–11
4.4.1 Using the XON/XOFF Commands for the Peripheral Port . . . I–12
4.4.2 Connecting A T30 Industrial Terminal (1784-T30) to the Peripheral Port I–12
4.4.2.1 Hardware Configuration I–14
4.4.3 Connecting a 1770-SA/SB Recorder to the Peripheral Port . . I–15
4.4.4 Connecting a 1770-HC Printer to the Peripheral Port I–15
4.4.5 Connecting RS-422 Devices I–16
4.5 Cable Assembly Parts I–19

Operating Functions 1

5.1 Chapter Objectives 1
5.2 Definition of Terms 1
5.2.1 Commands 1
5.2.2 Statements 1
5.2.3 Format Statements 2
5.2.4 Data Format 2
5.2.5 Integers 2
5.2.6 Constants 3
5.2.7 Operators 3
5.2.8 Variables 3
5.2.9 Expressions 4
5.2.10 Relational Expressions 4
5.2.11 System Control Values 4
5.2.12 Argument Stack 5
5.2.13 Control Stack 5
5.3 Description of Commands 6
5.3.1 Command: RUN 6
5.3.2 Command: CONT 7
5.3.3 Command: LIST 7
5.3.4 Command: LIST# or LIST[. 8
5.3.5 Command: NEW 9
5.3.6 Command: NULL [integer] 9
5.3.7 Command: Control C 9
5.3.7.1 Command: Disabling Control C 9
5.3.8 Command: Control S 10
5.3.9 Command: Control Q 10
5.3.10 Overview of EPROM File Commands 10
5.3.11 Commands: RAM and ROM [integer] 10
5.3.11.1 RAM 11
5.3.11.2 ROM 11

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.12 Command: XFER 12
5.3.13 Command: PROG 12
5.3.13.1 User PROM Check and Description – CALL 81 13
5.3.14 Command: PROG1 14
5.3.15 Command: PROG2 14
5.4 Description of Statements 16
5.4.1 Statement: CALL [integer] 16
5.4.2 Statement: CLEAR 16
5.4.3 Statement: CLEARI (clear interrupts) 16
5.4.4 Statement: CLEARS 17
5.4.5 Statements: CLOCK1 and CLOCK0 17
CLOCK1 17
CLOCK0 17
5.4.6 Statements: DATA – READ – RESTORE 18
DATA 18
READ 18
RESTORE 18
5.4.7 Statement: DIM 19
5.4.8 Statements: DO – UNTIL [rel expr] 19
5.4.10 Statement: END 22
5.4.11 Statements: FOR – TO – (STEP) – NEXT 22
5.4.12 Statements: GOSUB [ln num] – RETURN 23
GOSUB 23
RETURN 24
5.4.13 Statement: GOTO [ln num] 24
5.4.14 Statements: ON [expr] GOTO [ln num], [ln num],...[ln num],
ON [expr] GOSUB[ln num], [ln num],...[ln num] 25
5.4.15 Statements: IF – THEN – ELSE 25
5.4.16 Statement: INPUT 26
5.4.17 Statement: LD@ [expr] 28
5.4.18 Statement: LET 29
5.4.19 Statement: ONERR [ln num) 30
5.4.20 Statement: ONTIME [expr],[ln num] 31
5.4.21 Statement: PRINT or P. 32
5.4.22 Special Print Formatting Statements 33
5.4.22.1 PRINT TAB([expr]) 33
5.4.22.2 PRINT SPC([expr]) 33
5.4.22.3 PRINT CR 33
5.4.22.4 PRINT USING (special characters) 34
5.4.22.5 PRINT USING(Fx) 34
5.4.22.6 PRINT USING(#.#) 35
5.4.22.7 PRINT USING(0) 35
5.4.22.8 Reset Print Head Pointer – CALL 99 36
5.4.23 Statement: PRINT# or P.# 36
5.4.24 Statements: PH0., PH1., PH0.#, PH1.# 37
5.4.25 Statement: PUSH[expr] 38

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.26 Statement: POP[var] 39
5.4.27 Statement: REM 40
5.4.28 Statement: RETI 41
5.4.19 Statement: ST@[expr] 41
5.4.30 Statement: STOP 41
5.4.31 Statement: STRING 42
5.5 Description of Arithmetic, and Logical Operators and Expressions 43
5.5.1 Dual Operand (dyadic) Operators 43
5.5.1.1 Comments on logical operators .AND.,.OR. and .XOR. . . 44
5.5.2 Unary Operators 45
5.5.2.1 General Purpose Operators 45
5.5.2.2 Log Functions 46
5.5.2.3 Trig Functions 46
5.5.2.4 Comments on Trig Functions 47
5.5.3 Understanding Precedence of Operators 48
5.5.4 How Relational Expressions Work 49
5.6 Special Operators 50
5.6.1 Special Function Operators 50
5.6.1.1 GET 50
5.6.1.2 TIME 50
5.6.2 System Control Values 52
5.6.2.1 MTOP 52
5.6.2.2 LEN 52
5.7 Data Transfer Support Routines 52
5.7.1 Update Block-Transfer-Read Buffer (timed) – CALL 2 53
5.7.2 Update Block-Transfer-Write Buffer (timed) – CALL 3 53
5.7.3 Set Block-Transfer-Write Length – CALL 4 54
5.7.4 Set Block-Transfer-Read Length – CALL 5 54
5.7.5 Update Block-Transfer-Write Buffer – CALL 6 54
5.7.6 Update Block-Transfer-Read Buffer – CALL 7 54
5.7.7 Disable Interrupts – CALL 8 54
5.7.8 Enable Interrupts – CALL 9 55
5.7.9 Input Call Conversion Routines 55
5.7.9.1 3-Digit Signed, Fixed Decimal BCD to Internal Floating Point "XXX – CALL 10 55
5.7.9.2 16-Bit Binary (4-digit hex) to Internal Floating Point – CALL 11 55
5.7.9.3 4-Digit Signed Octal to Internal Floating Point "XXXX – CALL 12 55
5.7.9.4 6-Digit, Signed, Fixed Decimal BCD to Internal Floating Point "XXXXXX – CALL 13 56
5.7.9.5 4-Digit BCD to Internal Floating Point XXXX – CALL 17 . . 56
5.7.10 Output Call Conversion Routines 56
5.7.10.1 Internal Floating Point to 3-Digit, Signed, Fixed Decimal BCD "XXX – CALL 20 57

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.7.10.2 Internal Floating Point to 16-Bit Unsigned Binary (4 digit hex) – CALL 21 57
5.7.10.3 Internal Floating Point to 4-Digit, Signed Octal ”XXXX–Call 22 57
5.7.10.4 Internal Floating Point to 6-Digit, Signed, Fixed Decimal BCD ”XXXXXX – CALL 23 58
5.7.10.5 Internal Floating Point to 3.3-digit, Signed, Fixed Decimal BCD ”XXX.XXX – CALL 26 58
5.7.10.6 Internal Floating Point to 4-digit BCD XXXX – CALL 27 . 58
5.8 Peripheral Port Support 59
5.8.1 Peripheral Port Support – Parameter Set – CALL 30 59
5.8.2 Peripheral Port Support – Display Peripheral Port Parameters –CALL 31 60
5.8.3 Save Program to Data Recorder – CALL 32 61
5.8.4 Verify Program with Data Recorder – CALL 33 61
5.8.5 Load Program from Data Recorder – CALL 34 62
5.8.6 Get Numeric Input Character from Peripheral Port – CALL 35 63
5.8.7 Get the Number of Characters in the Peripheral Port Buffers – CALL 36 65
5.8.8 Clear the Peripheral Ports Input or Output Buffer – CALL 37 . 65
5.8.9 Save Labeled Program to Data Recorder (1770-SB only) – CALL 38 65
5.8.10 Load Labeled Program from Data Recorder (1770-SB only) – CALL 39 66
5.8.11 Print the Peripheral Port Output Buffer and Pointer – CALL 110 66
5.8.12 Print the Peripheral Port Input Buffer and Pointer – CALL 111 67
5.8.13 Reset the Peripheral Port to Default Settings – CALL 119 . . 67
5.9 Wall Clock Support Calls 67
5.9.1 Setting the Wall Clock Time (Hour, Minute, Second) – CALL 40 68
5.9.2 Setting the Wall Clock Date (Day, Month, Year) – CALL 41 . . 68
5.9.3 Set Wall Clock – Day of Week – CALL 42 68
5.9.4 Date/Time Retrieve String – CALL 43 69
5.9.5 Date Retrieve Numeric (Day, Month, Year) – CALL 44 (2) . . . 69
5.9.6 Time Retrieve String – CALL 45 69
5.9.7 Time Retrieve Number – Call 46 70
5.9.8 Retrieve Day of Week String – CALL 47 70
5.9.9 Retrieve Day of Week Numeric – CALL 48 70
5.9.10 Date Retrieve String – CALL 52 71
5.10 Description of String Operators 71
5.10.1 The ASC Operator 72
5.10.2 The CHR Operator 74
5.10.2.1 Clearing the Screen on an Allen-Bradley Industrial Terminal 74
5.10.2.2 Cursor Positioning on an Industrial Terminal 75
5.10.3 String Support Calls 75
5.10.3.1 String Repeat – CALL 60 80

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.10.3.2 String Append (Concatenation) – CALL 61 81
5.10.3.3 Number to String Conversion – CALL 62 81
5.10.3.4 String to Number Conversion – CALL 63 82
5.10.3.5 Find a String in a String – CALL 64 83
5.10.3.6 Replace a String in a String – CALL 65 84
5.10.3.7 Insert String in a String – CALL 66 85
5.10.3.8 Delete String from a String – CALL 67 86
5.10.3.9 Determine Length of a String – CALL 68 87
5.11 Memory Support Calls 87
5.11.1 ROM to RAM Program Transfer – CALL 70 88
5.11.2 ROM/RAM to ROM Program Transfer – CALL 71 88
5.11.3 RAM/ROM Return – CALL 72 89
5.11.4 Battery-backed RAM Disable – CALL 73 90
5.11.5 Battery-backed RAM Enable – CALL 74 90
5.11.6 Protected Variable Storage – CALL 77 90
5.12 Miscellaneous Calls 92
5.12.1 Program Port Baud Rate – CALL 78 92
5.12.2 Blink the Active LED by Default – CALL 79 93
5.12.3 Check Battery Condition – CALL 80 93
5.12.4 User PROM Check and Description – CALL 81 93
5.12.5 Reset Print Head Pointer – CALL 99 94
5.12.6 Print the Argument Stack – CALL 109 94
5.12.7 Print the Peripheral Port Output Buffer and Pointer – CALL 110 94
5.12.8 Print the Peripheral Port Input Buffer and Pointer – CALL 111 94
5.12.9 Reset the Peripheral Port to Default Settings – CALL 119 95

Programming 1

6.1 Chapter Objectives 1
6.2 Block-Transfer with the BASIC Module 1
6.2.1 Block-Transfer-Write and Block-Transfer-Read Buffers 1
6.3 Block-Transfer with PLC-2 Family Processors 2
6.3.1 PLC-2 Processor Program 2
6.3.1.1 Rung Description 3
6.4 PLC-3 Family Processors 5
6.4.1 Rung Description for Sample PLC-3 Family Ladder Logic – Single Data Set 7
6.5 PLC-5 Family Processors 7
6.5.1 Rung Description for Sample PLC-5 Family Ladder Logic . . 9
6.6. Block-Transfer Programming Tips 10

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Types 1

7.1 Chapter Objectives 1
7.2 Output Data Types 1
7.2.1 16-bit Binary (4 Hex Digits) 1
7.2.2 3-digit, Signed, Fixed Decimal BCD 2
7.2.3 4-digit, Unsigned, Fixed Decimal BCD 2
7.2.4 4-digit, Signed, Octal 3
7.2.5 6-digit, Signed, Fixed Decimal BCD 4
7.2.6 3.3-digit, Signed, Fixed Decimal BCD 5
7.3 Input Data Types 6

Editing A Procedure 1

8.2 Chapter Objectives 1
8.2 Entering the Edit Mode 1
8.3 Editing Commands/Features 1
8.3.1 Move 1
8.3.2 Replace 1
8.3.3 Insert 2
8.3.4 Delete 2
8.3.5 Retype 2
8.3.6 Exits 2
8.3.7 Renumber 2
8.4 Editing a Simple Procedure 4

Error Messages and Anomalies 1

9.1 Chapter Objectives 1
9.2 Error Messages from BASIC 1
9.3 Error Messages from CALL Routines 3
9.3.1 Data Conversion CALL Error Messages 3
9.3.2 Peripheral Port Support CALL Error Messages 3
9.3.3 Wall Clock CALL Error Messages 4
9.3.4 String Support CALL Error Messages 5
9.3.5 Memory Support CALL Error Messages 6
9.3.6 Miscellaneous CALL Error Messages 7
9.4 Anomalies 7

Quick Reference Guide 1

Decimal/Hexadecimal/Octal/ASCII Conversion Table 1

Basic Module Programming Hints 1

BASIC Module Programming Hints 1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using This Manual

1.1 Chapter Objectives

Read this chapter before you use the BASIC Module. It tells you how to use this manual properly and efficiently.

1.2 What this manual contains

This manual shows you how to install and operate your module. It gives you information about:

- hardware specifications.
- installing the module.
- the BASIC instruction set.
- programming the module.

This manual is not a BASIC tutorial document. We assume that you are familiar with BASIC programming.

1.3 Audience

Before you read this manual or try to use the BASIC Module, you should be familiar with the operation of the 1771 I/O structure as it relates to your particular processor. Refer to our Publication Index (publication number SD499) for the appropriate Programming and Operations manual.

1.4 Definitions of major terms

To make this manual easier for you to read and understand, we avoid repeating product names where possible. We refer to the:

- BASIC Language Module (Cat. No. 1771-DB) as the BASIC Module.
- Industrial Terminal System (Cat. No. 1770-T3/T4) as the industrial terminal.
- Data Recorder (Cat. No. 1770-SA/SB) as the 1770-SA/SB Recorder.
- RS-232-C compatible devices which communicate with the BASIC Module, such as the Industrial Terminal, SA/SB Recorder, computers, robots, barcode readers, or data terminals, as RS-423A/RS-232C devices.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.5 Important information

There are three different types of precautionary statements in this manual: Important, CAUTION and WARNING.

- Important: used to point out specific areas of concern when operating your BASIC Module.
- CAUTION: used to make you aware of instances where damage to your equipment could occur.
- WARNING: used to make you aware of instances where personal injury could occur.

1.6 Conventions

In this manual, we use certain notational conventions to indicate keystrokes and items displayed on a CRT or printer. A keystroke is shown in parentheses:

(ENTER)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introducing the BASIC Module

2.1 Chapter Objectives

This chapter discusses the functions and features of the BASIC Module. When you finish reading this chapter, you should:

- understand and be able to identify the hardware components of the BASIC Module.
- understand the basic features and functions of the BASIC Module.

2.2 General Features

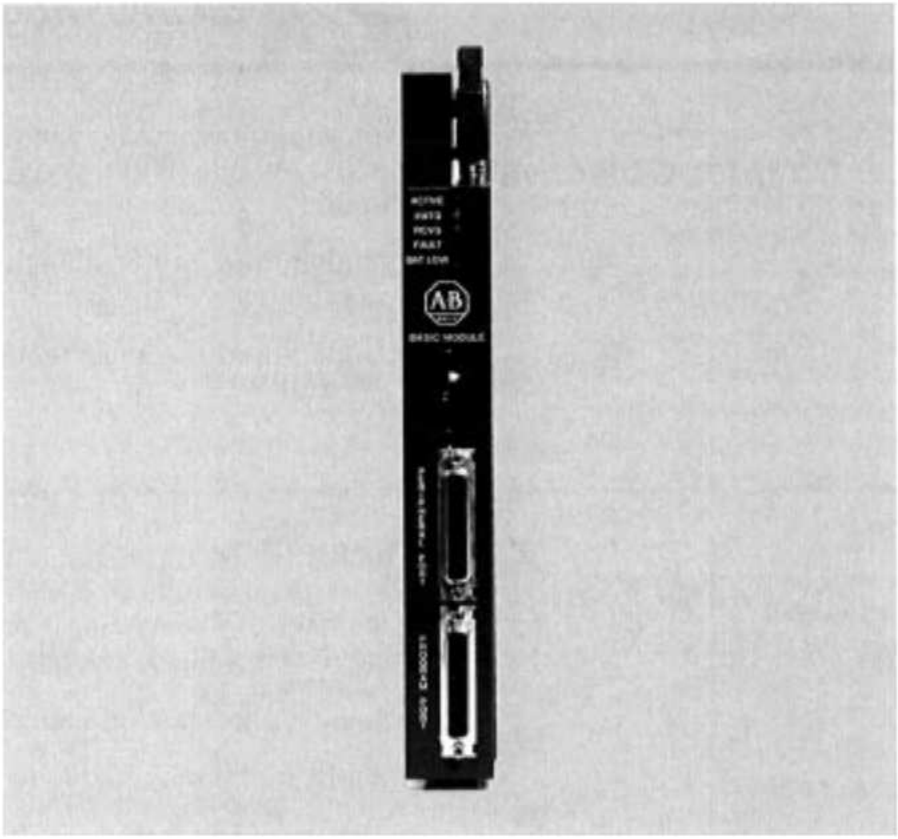
The BASIC Module (figure 2.1) provides math functions, report generation and BASIC language capabilities for any Allen-Bradley processor that communicates with the 1771 I/O system using block-transfer. It provides:

- basic programming using the Intel BASIC-52 language.
- math functions consistent with the BASIC-52 definition.
- two independently configurable serial ports capable of connecting to various user devices.
- user accessible real-time clock with 5 ms resolution.
- user accessible "wall" clock/calendar with 1 second resolution.
- program generation and editing using a dumb ASCII terminal or a T3/T4 Industrial Terminal in alphanumeric mode.
- program storage and retrieval using the 1770-SA/SB Recorder.
- block-transfer communication capability from a PLC-2, PLC-3 or PLC-5 family processor.
- on board program PROM burning.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.2 General Features (continued)

Figure 2.1
BASIC Module Front Edge



2.3 Hardware Features

Your module is a one-slot module with the following functions and features:

- 13 K bytes of battery backed RAM for user programs.
- 32 K bytes of EPROM storage for user software routines.
- One RS-423A/232C compatible serial communications port(PROGRAM port) which works with ASCII terminals providing operator program interaction, command level input printer output, etc. The program port baud rate defaults to 1200 baud. Initially you must set your terminal for 1200 baud. Use CALL 78 to change the program port baud rate. The program port is fixed at no parity, 1 start bit, 1 stop bit and 8 data bits.

  It also supports XON/XOFF for interruption of LISTing or to suspend data output from the program port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.3 Hardware Features (continued)

- One RS-423A/232C/RS-422 compatible serial communications port (PERIPHERAL port), supporting bi-directional XON/XOFF software handshaking and RTS/CTS, DTR, DSR, DCD hardware handshaking for interfacing to printers and commercial asynchronous modems. You can change the peripheral port configuration using a CALL 30. (Refer to Section 5.8.1). Default values are: 1 start bit, 1 stop bit, 8 bits/character, no parity, handshaking off and 1200 baud. The baud rate is jumper selectable (300 to 19.2 K bps). (Refer to Section 3.2.4 titled, "Configuration Plugs").
- Interface to the 1771 I/O rack backplane to support block-transfer.
- Wall clock/calendar with battery back-up available for program access.
- Battery replacement without removing the module from the I/O rack.
- All power derived from the backplane (1.5 A).
- Multiple BASIC modules can reside in the same I/O rack and function independently of each other.

2.4 Software Features

Your module runs BASIC language programs in an interactive mode through the dumb terminal/programming port interface, or on power-up. The execution of these programs allows a direct interface with programmable controller ladder programs.

Your module uses the following devices and features:

- terminal for programming, editing, system commands, displaying data and interactive program dialog
- serial port for report generation output, upload/download to 1770-SA/SB Recorder
- PLC-2, PLC-3 and PLC-5 data table reads and writes using block-transfer

We provide routines to use both the real-time clock and the wall-clock/calendar. The wall-clock time base is seconds.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.4 Software Features (continued)

You can start program execution:

- by entering commands at the interactive terminal.
- at power-up initialization.

You can store and execute programs in RAM or EPROM. You can store one user-program in RAM and up to 255 (depending on program size) independent user-programs simultaneously in EPROM memory.

The programs run single-task mode only. You can generate the following data types with the BASIC Module:

- 16-bit binary (4 hex digits)
- 3-digit, signed, fixed decimal BCD
- 4-digit, unsigned, fixed decimal BCD
- 4-digit, signed, octal
- 6-digit, signed, fixed decimal BCD
- 3.3 digit, signed, fixed decimal BCD

Refer to Chapter 7, "Data Types" for more information.

2.5 Specifications

Isolation
- The Programming Port is isolated from the 1771 I/O backplane. (+500 V)
- The Peripheral Port is isolated from the 1771 I/O backplane. (+500 V)
- The Programming Port is isolated from the Peripheral Port. (+500 V)

Communication Rates
- 300, 600, 1200, 2400, 4800, 9600, 19.2 K bits
- Communication rates/distances

Communication Rate (bps)	Maximum Distance Allowed		
	RS-232-C	RS-423	RS-422
300	50	4000	4000
600	50	3000	4000
1200	50	2500	4000
4800	50	800	4000
9600	50	400	4000
19,200	50	200	4000

Real-time clock accuracy
- Absolute @ 25° C
- Drift ≤ ± 50 ppm/°C
- Drift ≤ ± 50 ppm/year @ 25° C

Port driver and receiver
- Drive output +3.6 V minimum
- Receiver sensitivity 200 mV minimum

Math
- Precision: 8 significant digits
- Range: ± 1E-127 to ± 99999999E+127

- Formats: integer, decimal, hexadecimal and exponential

Module location
- One 1771 I/O chassis module slot

Backplane power supply load
- 1.5 A

Environmental Conditions
- Operational temperature: 0°C to 60°C (32°F to 140°F)
- Storage temperature: -40°C to 85°C (-40°F to 185°F)
- Relative humidity: 5% to 95% (non-condensing)

Keying (top backplane connector)
- Between 8 and 10
- Between 32 and 34

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Installing the BASIC Module

3.1 Chapter Objectives

This chapter describes how to install your BASIC module in a 1771 I/O rack. After reading this chapter you should be able to:

- configure the module using the configuration plugs.
- insert the module into a 1771 I/O backplane.
- understand module status indicators.
- install additional EPROM's.

3.2 Installing the BASIC module



WARNING: Disconnect and lockout all AC power from the programmable controller and system power supplies before installing modules to avoid injury to personnel and damage to equipment.

Read this installation section completely before beginning. Re-check all option selections and connections before you begin programming.

Before installing your module in the I/O chassis you must:

1. calculate the power requirements of all the modules in each chassis. (Refer to Section 3.2.1 below).
2. determine the location of the module in the I/O chassis. (Refer to Section 3.2.2 below).
3. key the backplane connectors in the I/O chassis. (Refer to Section 3.2.3 below).
4. set the module configuration plugs. (Refer to Section 3.2.4 below).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.1 Power Requirements

Your module receives its power through the 1771 I/O chassis backplane from the chassis power supply. It does not require any other external power supply to function. When planning your system you must consider the power usage of all modules in the I/O chassis to prevent overloading the chassis backplane and/or power supply. Each BASIC module requires 1.5 A at +5V DC. Add this to the requirements of all other modules in the I/O chassis.



CAUTION: Do not insert or remove modules from the I/O chassis while system power is on. Failure to observe this rule may result in damage to module circuitry.

3.2.2 Module Location in the I/O Chassis

You can place your module in any I/O slot of the I/O chassis except for the extreme left slot. This slot is reserved for processors or adapter modules. You can place your module in the same module group as a discrete high density module if you are using processors or adapters with single-slot addressing capabilities.

Important: Certain processors restrict the placement of block-transfer output modules. Refer to the user manual for your particular processor for more information.

3.2.3 Module Keying

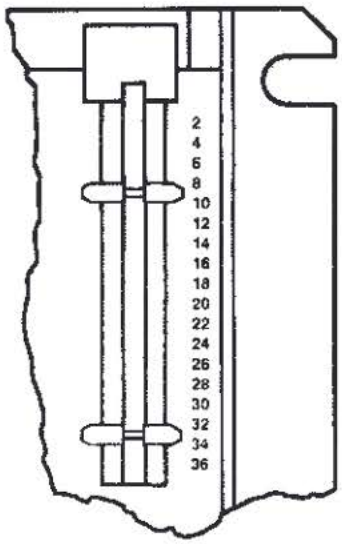
Initially you can insert your module into any I/O module slot in the I/O chassis. However, once you designate a slot for a module you must not insert other modules into these slots. We strongly recommend that you use the plastic keying bands shipped with each I/O chassis, to key I/O slots to accept only one type of module. Your module is slotted in two places on the rear edge of the board. The position of the keying bands on the backplane connector must correspond to these slots to allow insertion of the module. You may key any I/O rack connector to receive the module assembly. Snap the keying bands onto the upper backplane connectors between the numbers printed on the backplane (figure 3.1).

- Between 8 and 10
- Between 32 and 34

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.3 Module Keying (continued)

Figure 3.1
Keying Diagram for Placement of Module Keying Bands



You may change the position of these bands if subsequent system design and rewiring makes insertion of a different type of module necessary. Use needle-nose pliers to insert or remove keying bands.

3.2.4 Configuration Plugs

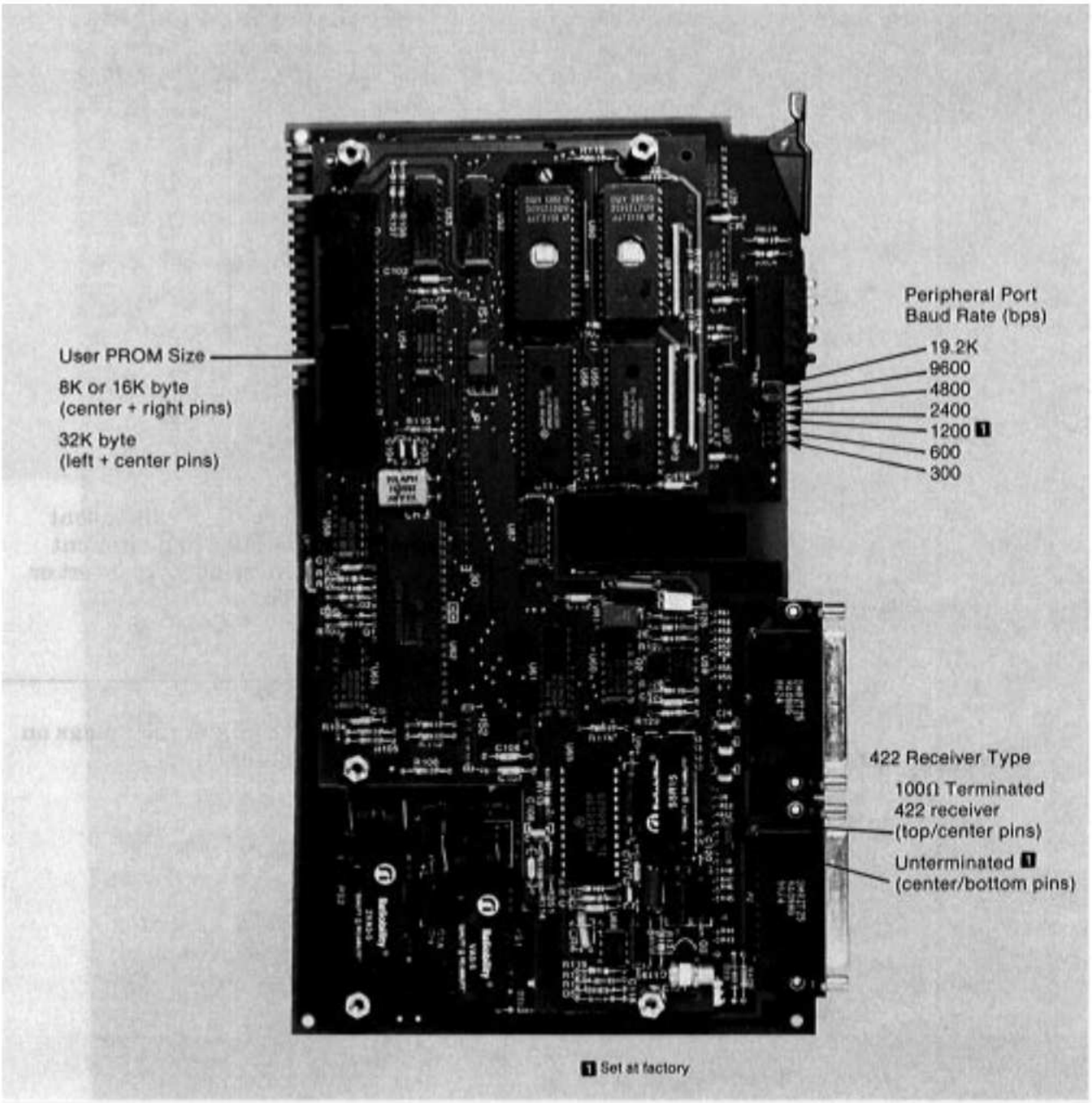
There are three sets of user selectable configuration plugs on the BASIC Module (figure 3.2). You can use these configuration plugs to select:

- PROM size.
- peripheral port baud rate (bps).
- 422 receiver termination.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.4 Configuration Plugs (continued)

Figure 3.2
The Configuration Plugs



All other configuration plugs are factory set. Do not reset these factory set configuration plugs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.5 Module Installation

Now that you have determined the configuration, power requirements, location, keying and wiring for your module, you are ready to install it in the I/O chassis.

1. Turn off power to the I/O chassis.
2. Insert your module in the I/O rack. Plastic tracks on the top and bottom of the slots guide the module into position. Do not force the module into its backplane connector. Apply firm, even pressure on the module to seat it properly. Note the rack, module group and slot numbers and enter them in the module address section of the block-transfer instructions.
3. Snap the I/O chassis latch over the module. This secures the module in place.

3.2.6 Initial Start-up Procedure

You must use the following procedure when powering up the module for the first time. This procedure is a continuation of the installation procedure presented above.

4. Connect the cable from your program terminal to the BASIC Module program port.



CAUTION: Be sure you properly ground the system be fore turning on power. A difference in ground potential between the BASIC Module serial connectors and your program terminal or other serial device can cause damage to the equipment or loss of module programs.

5. Turn on your program terminal. Select 1200 baud. If you are using an industrial terminal, select the Alpha Numeric mode, baud rate and press [RETURN].
6. Turn on power to the rack. The following sequence takes place:
   - Fault (FLT) and ACTIVE LED's go on.
   - FLT and ACTIVE LED's go out until power-up diagnostics is complete. Ignore any other LED activity during power-up.
   - ACTIVE LED goes on.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.6 Initial Start-up Procedure (continued)

When the ACTIVE LED comes on observe the sign-on message displayed on the terminal followed by < READY.

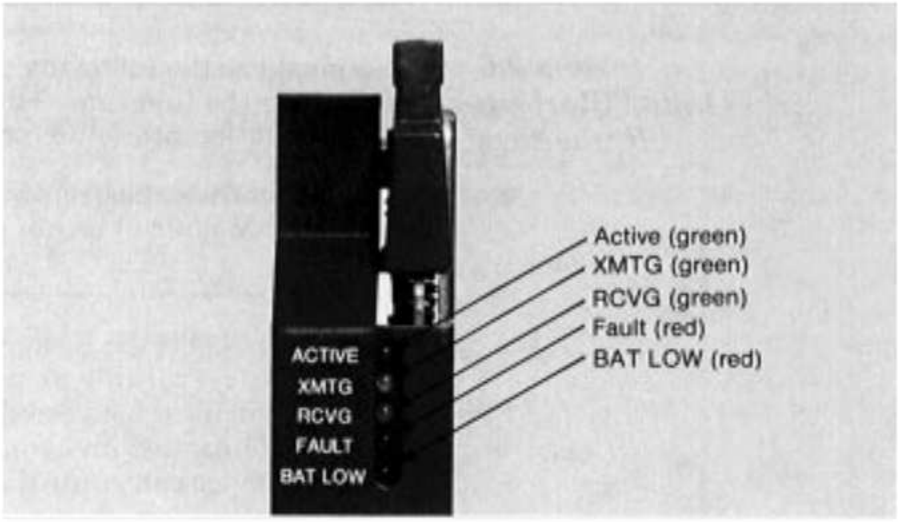
You are now ready to begin BASIC programming. Refer to Chapter 6 for an example program to help you get your processor and BASIC Module communicating properly.

Important: If you break communications with the module check that the terminal is set at the proper baud rate.

3.3 Module Status LED's

There are five LED's (figure 3.3) on the front panel of the module which indicate the status of the module.

Figure 3.3
Module Status Indicators



LED	Description
ACTIVE (green)	Indicates the module has passed power-up diagnostics. You can program using CALL 79 to: □ remain on (default). □ remain on in RUN mode and blink every second when in COMMAND mode. Refer to Chapter 5 for an explanation of CALL 79.
XMTG (green)	ON when data is transmitting on the peripheral port. Lights for either RS-422 or RS-423/RS-232C output.
RCVG (green)	ON when data is transmitting on the **peripheral** port. Lights for either RS-422 or RS-423/RS-232C input. This LED does not indicate whether or not valid data was received.
FAULT (red)	When LED is on, indicates either a hardware problem or block-transfer problem. See below.
BAT LOW (red)	Lights when the battery voltage drops below about 3.0V DC.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 Module Status LED's (continued)

If the FLT LED lights after the module has been operating properly check the following troubleshooting chart.

Problem	Probable Cause	Recommended Action
Module's programming port does not respond	Hardware failure	Send module for repair
Module's programming port continues to function but FLT LED goes on and off	Problem with block-transfers between processor and BASIC module	Verify ladder logic
Module's programming port continues to function and FLT LED goes out when processor is switched to program mode		
Module's programming port continues to function and FLT LED remains on	Problem with block-transfer circuitry on the BASIC Module	Send module for repair

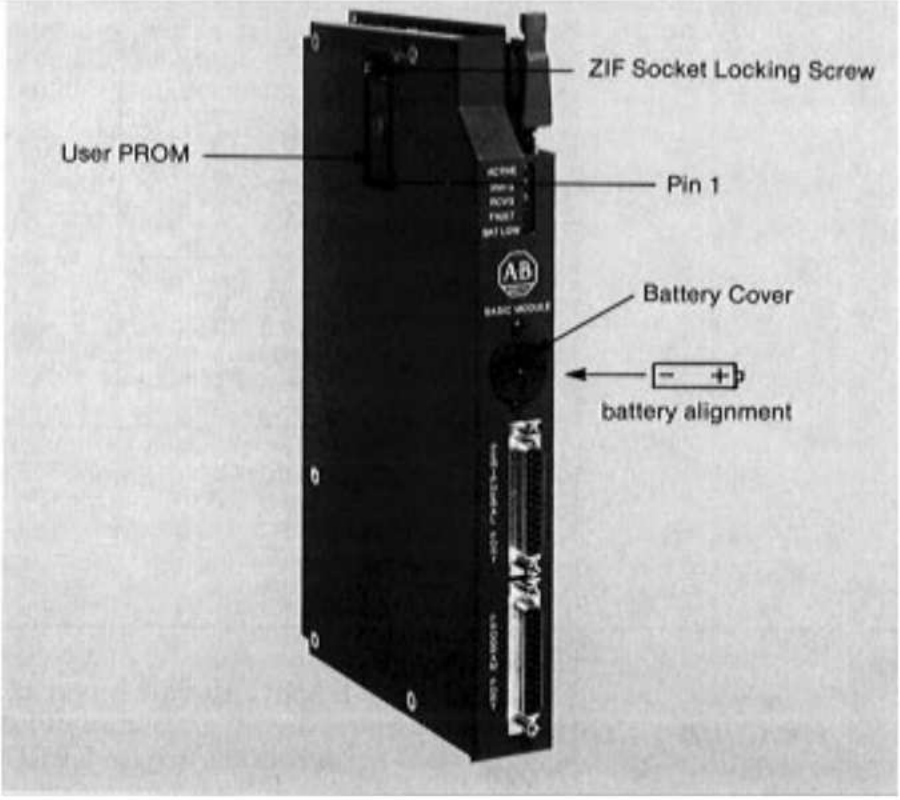
3.4 Installing the User Prom

The BASIC Module has a 32 K byte EPROM installed (figure 3.4). We recommend that you keep JEDEC standard 8 K, 16 K or 32 K byte EPROMs which use 12.5V DC programming voltage as spares. You can buy 32 K byte EPROMs from Allen-Bradley (part numbers 940654-02 or 9406454-03).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Installing the User Prom (continued)

Figure 3.4
User PROM and Battery Holder



To replace the EPROM:

1. Turn the small screw in the socket just above the chip (figure 3.4) 1/4 turn counterclockwise.

2. Remove the old chip.

3. Insert the new chip with pin one down and the center notch down as shown in the socket diagram.

4. Turn the small screw in the socket above the chip 1/4 turn clockwise.

5. Refer to the above section titled, "Configuration Plugs" for the proper setting of the corresponding configuration plug.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4.1 Electrostatic Discharge

Electrostatic discharge can damage integrated circuits or semiconductors in this module if you touch backplane connector pins. It can also damage the module when you set configuration plugs and/or switches inside the module. Avoid electrostatic damage by observing the following precautions:

- Touch a grounded object to rid yourself of charge before handling the module.
- Do not touch the backplane connector or connector pins.
- If you configure or replace internal components, do not touch other circuit components inside the module. If available, use a static-safe work station.
- When not in use, keep the module in its static-shield bag.



CAUTION: Electrostatic discharge can degrade performance or damage the module. Handle as stated above.

3.5 Battery

The 13 K bytes of user RAM and the clock/calendar are battery backed. Drain on the battery should be less than 0.5 mA DC during battery back-up (no power) and less than 50 uA while the module is powered. Battery life during no-power conditions is about 2000 hours. Battery shelf life is about 20,000 hours. When the BAT LOW indicator comes on the battery should maintain the clock and program data for about three days. We recommend immediate replacement.

To replace the battery (figure 3.4):

1. Place a screwdriver in the battery cover slot.
2. Press inwards slightly.
3. Rotate the screwdriver and battery cover counterclockwise 1/4 turn.
4. Release the pressure and remove the battery cover.
5. Replace the battery with the positive (+) terminal out.
6. Replace the battery cover.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 Battery (continued)

The BAT LOW indicator should go out.

You can monitor the battery low condition in revision A and revision B modules using a XBY(77B4H) statement. Bit 2 high indicates the battery low condition.

With revision C modules use CALL 80 to monitor battery status.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Serial Ports

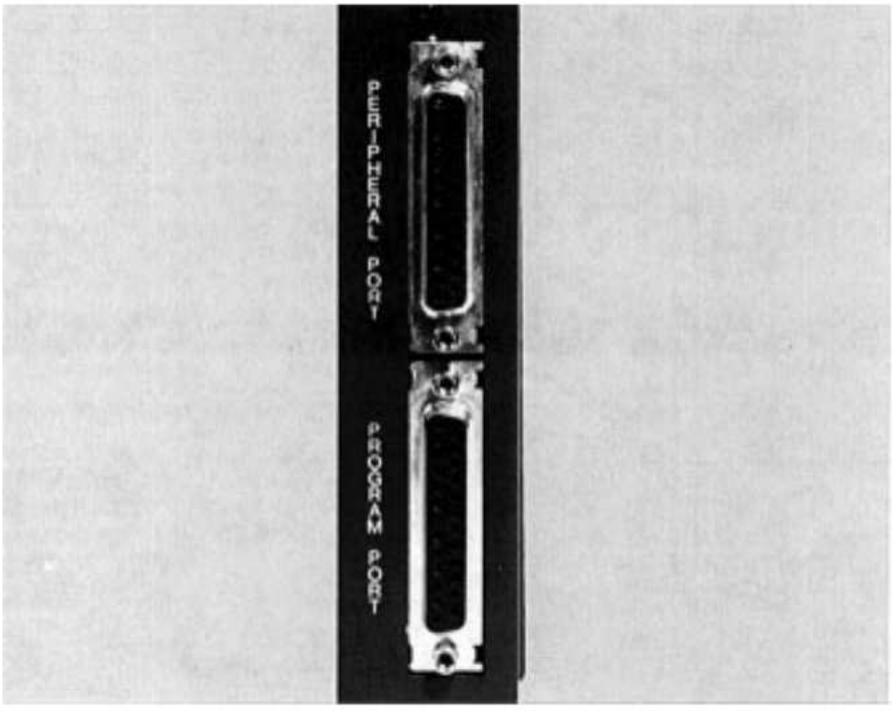
4.1 Chapter Objectives

This chapter describes how to use the program serial port and the peripheral serial port to connect terminals, Data Cartridge Recorders, Digital Cassette Recorders, printers and other compatible devices.

4.2 Using the BASIC Module Program and Peripheral Communication Ports

The BASIC Module has a program serial port and a peripheral serial port capable of connecting to various user devices (figure 4.1). You can configure each port independently. Both ports are electrically isolated from each other and from the backplane up to 500 V with no external power needed. Both ports operate from 300 baud to 19.2K baud and default to 1200 baud.

Figure 4.1
Program/Peripheral Port Locations



If you use an RS-423/RS-232 device or an Allen-Bradley Industrial Terminal you can use up to a 50 foot maximum cable length for connections from either the program or peripheral ports. Refer to the specifications section in Chapter 2 for cable length recommendations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.2 Using the BASIC Module Program and Peripheral Communication Ports (continued)

CAUTION: Be sure you properly ground the system before turning on power. A difference in ground potential between the BASIC Module serial connectors and your program terminal or other serial device can cause damage to the equipment or loss of module programs.

4.2.1 Pin Descriptions

Use the following pins for connections made to the program or peripheral ports. Refer to figure 4.2 for pin descriptions. Not all signals are available on both ports.

Signal states are:

mark = logical 1 = – voltage
space = logical 0 = + voltage

Pin	Name	Description
1[1]	Chassis/ Shield	Connect this pin to chassis ground for shielding purposes.
2[1]	TXD	TXD is an RS-423A compatible serial output port.
3[1]	RXD	RXD is an RS-423A compatible serial input data port.
4	RTS	RTS is an RS-423 compatible hardware handshaking output line. This line changes to a mark (1) state when the BASIC Module has data in the output queue and is requesting permission to transmit to the data communications equipment.
5	CTS	CTS is an RS-423A compatible hardware handshaking input line. This line must be in a mark (1) state for the BASIC Module to transmit on the peripheral port. If no corresponding signal exists on the data communications equipment, connect CTS to RTS.
6	DSR	DSR is a general purpose RS-423A compatible input line. The BASIC Module transmits or receives in the mark (1) or space (0) state. Use this line for data recorder interface.
7[1],9,10	Signal Common	Use the signal common pins to reference all RS-423A/RS-422 compatible signals.
8	DCD	If DCD is enabled using CALL 30, the BASIC Module does not transmit or receive characters until the DCD line is in the mark (1) state. When disabled, the module ignores the state of this line.
11,12,13,15 17,19,21,22 23,24	NC	No connection
14,25	422 TXD	RS-422A compatible equivalent of the RS-423A TXD line. Differential serial output lines.
16, 18	422 RXD	Differential RS-422A compatible serial input lines.
20	DTR	DTR is an RS-423A compatible hardware handshaking output line. This line changes to a space (0) state when the BASIC Module input queue has accumulated more than 223 characters. The DTR line changes to a mark (1) state when the input queue contains less than 127 characters.
[1]program port pins		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3 Program Port

The program port is an RS-423A/232C compatible serial port that provides minimum signal line connection to terminals, printers and other serial devices for operator-program interaction, command level input, printer output etc. Figure 4.2 shows the signals available on both the program port and the peripheral port described later.

Figure 4.2
Program Port and Peripheral Port Wiring Connections

Program Port	Peripheral Port	Description
1[1]	1[1]	Chassis/Shield
2[1]	2[1]	TXD-Output
3[1]	3[1]	RXD-Input
4	4[1]	RTS-Output
5	5[1]	CTS-Input
6	6[1]	DSR-Input
7[1]	7[1]	Signal Common
8	8[1]	DCD-Input
9	9[1]	Signal Common
10	10[1]	Signal Common
11	11	NC
12	12	NC
13	13	NC
14	14[1]	RS-422 TXD
15	15	NC
16	16[1]	RS-422 RXD
17	17	NC
18	18[1]	RS-422 RXD'
19	19	NC
20	20	DTR-Output
21	21	NC
22	22	NC
23	23	NC
24	24	NC
25	25[1]	RS-422 TXD'

[1]Signal is provided on this pin

The baud rate is initially set at 1200 baud. You can use CALL 78 to change the baud rate from 300 to 19.2K bps.

The program port has the following fixed format:

- parity: none
- start bits: 1
- stop bits: 1
- data bits: 8
- receiver threshold: 200 mV
- driver output (loaded): +3.6V

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3 Program Port (continued)

Important: The program port always resets the most significant bit of all its data inputs. The range of each byte of data is 0 to 127 ($7F_H$). On output, the module transmits all bits as specified when using the PRINT CHR() command except for the XOFF (13H) character. The range of each byte of data is 0 to 255 (OFF_H).

Important: The program port automatically inserts a CR, LF sequence after the 79th character column. Use CALL 99 to reset the column counter to zero to allow PRINT page width's in excess of 79 characters.

You enter BASIC programs through a dumb ASCII terminal, such as an industrial terminal in alphanumeric mode. Refer to section 4.3.2, "Connecting a T3/T4 Industrial Terminal to the Program Port".

4.3.1 Using the XOFF/XON Commands for the Program Port

Use the XOFF/XON commands to disable outputs from the program port in the following way.

1. Use XOFF only on PRINT statements.
2. When XOFF is received during a PRINT, data output and program execution are suspended immediately.
3. When XOFF is received at any other time, program execution continues until a PRINT is encountered. When a PRINT is encountered program execution is suspended.
4. Use XON to resume program execution.

   The program port accepts uppercase or lowercase input, however, the input receiver changes all commands, keywords or variables to upper case before storing in memory, thus:

   > 10 print "hello"(CR)

   appears as

   10 PRINT "hello"

   when listed.

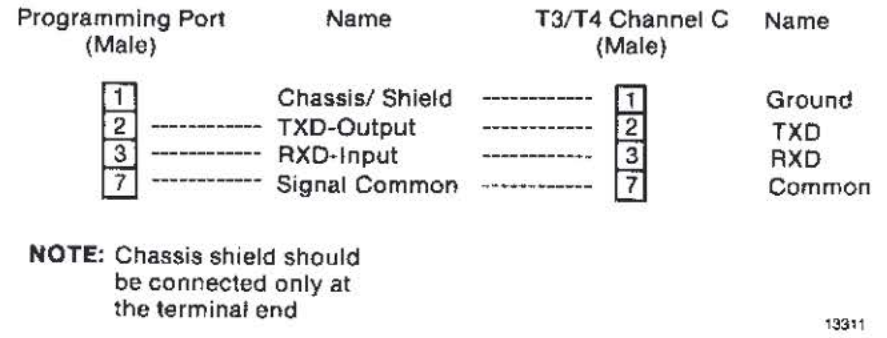
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.2 Connecting a T3/T4 Industrial Terminal to the Program Port

You can use an Industrial Terminal System as the programming system for the BASIC Module. Connect the module to CHANNEL C only. You can construct cable for distances up to 50 feet. Figure 4.3 shows cable connections to a T3/T4 Industrial Terminal from the program port.

Important: You may continue to use CHANNEL B in existing installations.

Figure 4.3
Cable Connection to T3/T4 Terminal from the Program Port



You can use a T3 or T4 Industrial Terminal with the following keyboard revisions:

- T3 Series A, Revision H or later
- T3 Series B, Revision H or later
- T3 Series C, Revision A or later
- T4 Series A, Revision F or later

Refer to the Industrial Terminal Systems User's Manual (Cat. No. 1770-T1, T2, T3), publication number 1770-6.5.3, and PLC-3 Industrial Terminal User's Manual (Cat. No. 1770-T4), publication number 1770-6.5.15 for additional information.

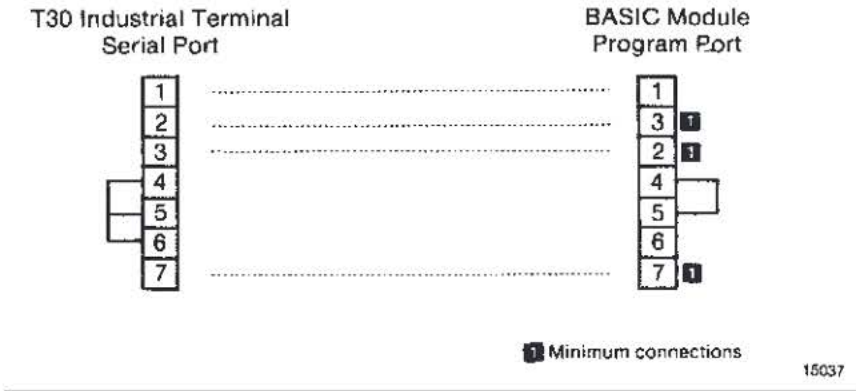
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.3 Connecting a T30 Industrial Terminal (Cat. No. 1784-T30) to the Program Port

You can connect a T30 Industrial Terminal to the BASIC Module program port to act as a dumb terminal.

Refer to the following figure 4.4 for BASIC Module/T30 connections.

Figure 4.4
Connecting a T30 Industrial Terminal to a BASIC Module



Important: Jumper T30 Industrial Terminal pin 4,5 and 6; and BASIC Module pins 4 and 5 if you do not use them.

4.3.4 Connecting a T50 Industrial Terminal (Cat. No. 1784-T50) to the Program Port

You can use your T50 Industrial Terminal as a BASIC Module programming device. You must use a terminal driver package to configure the industrial terminal serial port and communications protocol to match the BASIC Module.

These packages include the ability to upload to and download from the hard or floppy disk drives in the industrial terminal.

To upload and download you must:

1. configure the software.
2. construct a cable with the pin connections shown in figure 4.5 under, "Wiring".
3. use the upload and download commands of the driver package.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.4 Connecting a T50 Industrial Terminal (Cat. No. 1784-T50) to the Program Port (continued)

4.3.4.1 Configuring the Software

Configure the driver package for compatibility with the BASIC Module by setting:

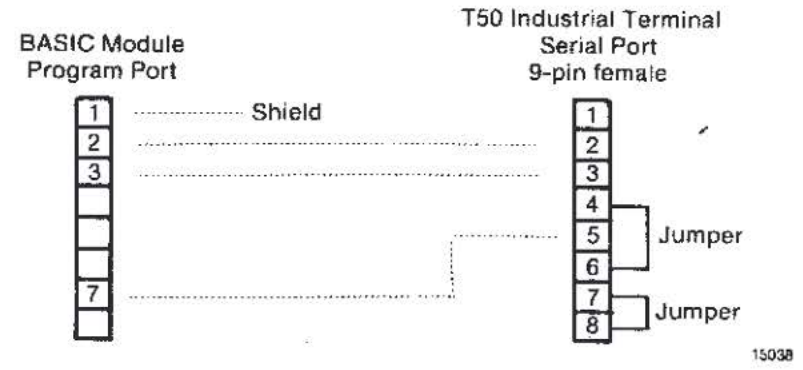
- baud rate – 9600 baud recommended
- parity – none
- data bits – 8
- start bits – 1
- stop bits – 1

To download to the BASIC Module, you must use a line wait function. The industrial terminal waits for the " > " BASIC Module prompt before sending the next line to the module. You must enter a line delay of 1.5 seconds for terminal drivers that do not have the "wait for character" function, so that you do not lose subsequent lines. Most drivers allow storage of the complete set of parameters in a file for later recall.

The industrial terminal stores the BASIC Module program in a text file on the hard or floppy disc depending on where you store the terminal driver package. We recommend you store the driver package on the hard drive to increase execution speed. Most driver packages have upload and download capability. Refer to the driver documentation for these commands.

4.3.4.2 Wiring

Figure 4.5
Connecting a T50 Industrial Terminal to a BASIC Module



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4 Peripheral Port

The peripheral port is an asynchronous serial communication channel compatible with RS-423A/232C or RS-422 interfaces. It uses bi-directional XON/XOFF software handshaking and RTS/CTS, DTR, DSR, DCD hardware handshaking for interfacing with printers, terminals and commercial asynchronous modems. Use a CALL routine to change peripheral port configuration. Configure the baud rate (300 to 19.2K bps) by setting a configuration plug. Refer to figure 3.2 for configuration plug locations.

In addition, the peripheral port has the following format requirements:

- configurable parity: odd, even or none
- fixed start bits: 1
- configurable stop bits: 1, 1.5 or 2
- configurable data bits: 5,6,7 or 8
- receiver threshold: 200 mV
- driver output (loaded): +3.6V

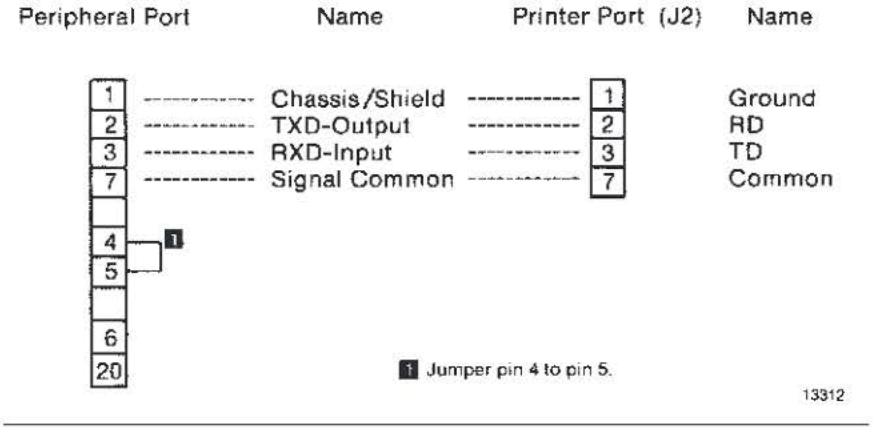
Defaults are 1 start bit, 1 stop bit, 8 bits/character, no parity, handshaking off and 1200 baud.

When you select 8 bits/character you have full access to all 8 bits of each character on both input and output data bytes.

Refer to figure 4.2 for peripheral port wiring connections.

The peripheral port can connect to printers (figure 4.6), asynchronous modems and to SA/SB recorders for program storage and retrieval (figure 4.7).

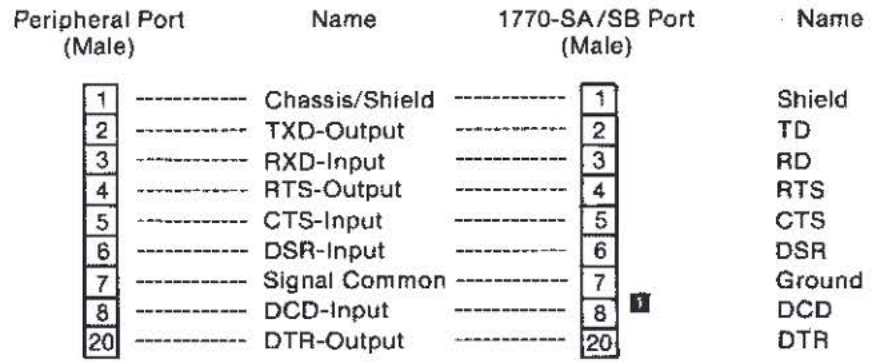
Figure 4.6
Cable Connection to 1771-HC Printer



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4 Peripheral Port (continued)

Figure 4.7
Cable Connection to SA/SB Recorder



1 Pins 9 through 19 and 21 through 25 must be unconnected.

13313

4.4.1 Using the XON/XOFF Commands for the Peripheral Port

Output Data – The BASIC Module stops sending characters within 2 character times after receiving an XOFF. Transmission resumes when XON is received.

Input Data – The BASIC Module sends XOFF when the input buffer reaches 224 characters. The module sends XON when the buffer contains less than 127 characters.

The BASIC Module requires CTS (pin 5) to be true before data can be output. If hardware handshaking is not used with your device then RTS (pin 4) may be connected to CTS to satisfy this requirement. Jumper pin 4 to pin 5.

4.4.2 Connecting A T30 Industrial Terminal (1784-T30) to the Peripheral Port

Communication between a programmable controller and the T30 Industrial Terminal using the BASIC Module requires two data transfers. We use block-transfer-read and write instructions for bi-directional data transfer between the programmable controller data table and the BASIC Module. We use an RS-423/RS-232 communication link for bi-directional data transfer between the BASIC Module and the industrial terminal. Refer to the Plant Floor Terminal Application Data (publication number 1784-4.1) for more information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4.2 Connecting A T30 Industrial Terminal (1784-T30) to the Peripheral Port (continued)

4.4.2.1 Hardware Configuration

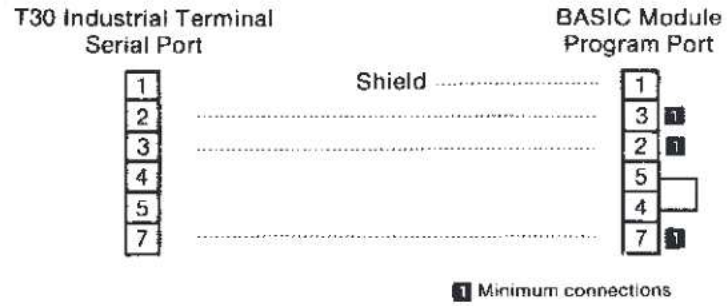
You must configure the BASIC Module peripheral port and the T30 Industrial Terminal serial port in the same way for proper communications to occur. We configure the peripheral port on the BASIC Module as follows:

Baud rate	1200 bps
Parity	disabled
Duplex	full (default setting)
Bits per character	8
Stop bits	1
Handshaking	disabled

Configure the T30 Industrial Terminal serial port the same way.

Refer to the following figure 4.8 for T30 Serial Port/BASIC Module Peripheral Port connections.

Figure 4.8
T30 Serial Port/BASIC Module Peripheral Port Connections



Important: Jumper the T30 Industrial Terminal and BASIC Module pins 4 and 5 if you do not use them.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4.3 Connecting a 1770-SA/SB Recorder to the Peripheral Port

You can use a 1770-SB Data Cartridge Recorder or 1770-SA Digital Cassette Recorder to save and load BASIC programs to the BASIC Module. Figure 4.6 shows cable pin connections. Use the connections shown in figure 4.6 otherwise improper operation could occur. Note that the standard cable does not connect properly with the BASIC Module. Refer to the user manuals for the 1770-SB (publication number 1770-6.5.4) and 1770-SA (publication number 1770-6.5.1) for more information on these recorders.

It is not necessary to set the peripheral port parameters (except baud rate) before CALLing the recorder interface routines. This is done automatically by the software. The parameters are returned to their original state when the routine is complete.

You can find more information on saving and loading programs in Chapter 6 of this manual.

Important: STR LINK II and III Recorders do not function like SA/SB recorders. Do not use them with the BASIC Module.

4.4.4 Connecting a 1770-HC Printer to the Peripheral Port

You can connect a 1770-HC Printer to the peripheral port for program listing, report generation etc. Figure 4.7 shows cable pin connections. Refer to your printer product manual for more information.

We recommend enabling XON/XOFF on the peripheral port (see Chapter 5 section titled, "Peripheral Port Support – Parameter Set") and selecting XON/XOFF(DEC) protocol on the 1770-HC Printer (switch selectable). Refer to your printer manual. You can find more information on printing reports and listing programs in Chapter 5 of this manual.

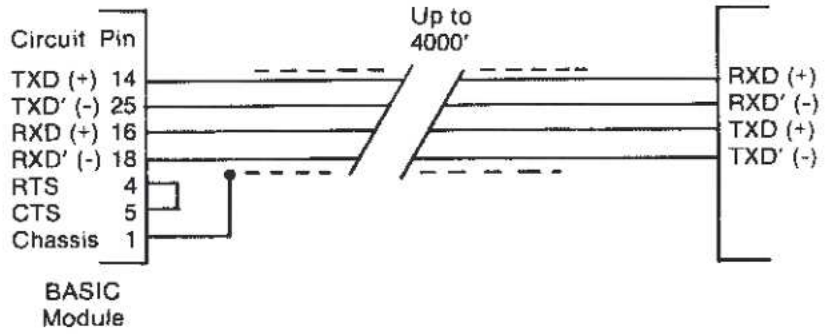
4.4.5 Connecting RS-422 Devices

The BASIC Module can communicate with various RS-422 devices. RS-422 signals for both sending and receiving data are located on the module's peripheral port. Figure 4.9 shows point-to-point signal connections. The RS-422 port floats (i.e no voltages are applied to the output) when it is not sending characters. This allows you to connect two transmitting devices on the same line. Also, you can connect more than one device in a multi-drop configuration (figure 4.10).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

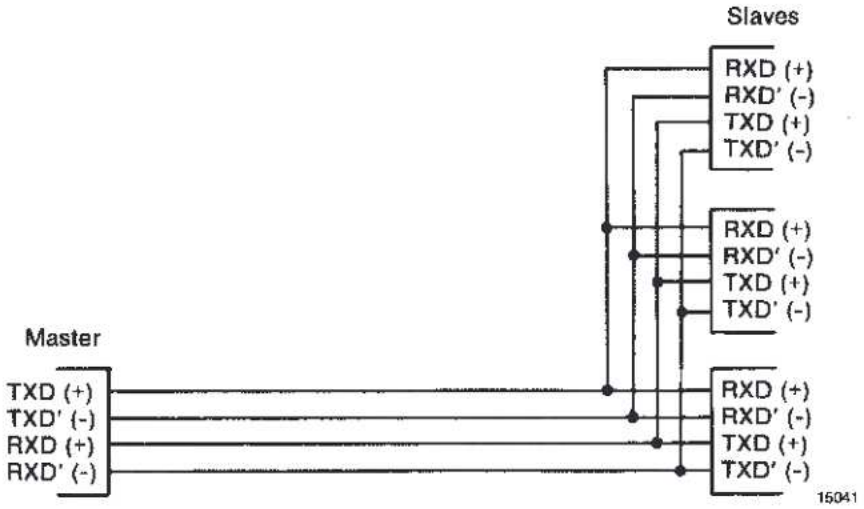
4.4.5 Connecting RS-422 Devices (continued)

Figure 4.9
Point-to-Point RS-422 Connections



Note: Connect 100Ω termination resistor across RXD, RXD' if not internal to device.

Figure 4.10
Multi-drop configuration with master and multiple slaves



Important: When you use the peripheral port as a 422 port, you must connect pin 4 to pin 5 on the port.

When using an RS-422 interface you must install termination resistors at each end of the line. The module has a jumper selectable termination resistor (Refer to figure 3.2). Use a cable with 2 twisted pairs and a nominal impedance of 100 ohms.

Important: Use terminating resistors only at the ends of the link if using multiple RS-422 devices, and at both ends if using point-to-point connections.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.5 Cable Assembly Parts

You must supply cables for connecting devices to the program and peripheral ports. You can construct the cables with the parts listed in Table 4–1.

Table 4–1
Cable assembly parts for connection to the program and peripheral ports

Part	Manufacturer's Part Number
25 pin female connector	Cannon type DB-25S, or equivalent
25 pin male connector	Cannon type DB-25P, or equivalent
Plastic Hood	Amp type 205718-1
2 twisted pair 22 gauge, individually shielded cable	Cat. No. 1778-CR, Belden 8723 or equivalent (Do not use for cable RS-422 connections)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Operating Functions

5.1 Chapter Objectives

After reading this chapter you should be familiar with the BASIC instruction set and be ready to begin BASIC programming. › This chapter is a reference section › to help you with module programming. You should › already › be familiar with BASIC programming.

5.2 Definition of Terms

The following sections define the following terms: commands, statements, format › statements, › data format, › integers, constants, operators, variables, expressions, relational expressions, › system control › values, argument stack and control stack.

5.2.1 Commands

The BASIC module operates › in two modes, the › command or direct mode › and the interpreter › or run mode. You can only enter commands when the › processor is in the command or direct › mode. This › document uses › the terms run mode and command mode › to refer › to the two different › operation › modes.

5.2.2 Statements

A BASIC program consists › of statements. Every statement begins › with a line › number, followed › by a statement body, and terminated › with › a carriage › return (cr), or a colon (:) in the case of multiple › statements per line. › There are three types of statements: assignments, input/output › and control.

- Every line in a program must have › a statement line › number ranging between › 0 and 65535 › inclusive. › BASIC uses › this to order the program statements in sequence.
- You can use a statement number only › once in a program.
- BASIC automatically › orders › statements in ascending order.
- A statement › may not contain more than › 79 characters.
- BASIC ignores › blanks (spaces) and automatically inserts them during a LIST command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.2 Statements (continued)

- You may put more than › one statement on a line, if separated by a colon › (:). You can use only one statement number per › line.
- You can enter lower case characters › in the COMMAND mode. Any keywords, commands, variable and array names entered in lower case change to upper case when stored › in memory.

5.2.3 Format Statements

You can use format › statements within the print › statement. The format statements › include › TAB((|expr|), › SPC([expr]), USING(special symbols), and CR (carriage return with no line feed).

5.2.4 Data Format

You can represent the following range of numbers in › the BASIC module: +1E–127 › to +.99999999E+127

There are eight › significant › digits. › Numbers are › internally rounded to › fit this precision. You can › enter and display numbers in › four formats: integer, › decimal, hexadecimal and exponential.

Example: › 129, 34.98, 0A6EH, 1.23456E+3

5.2.5 Integers

In the BASIC module, › integers › are numbers that › range from 0 to 65 535 or OFFFFH. You › can › enter all integers in either decimal or hexadecimal › format. You indicate › a hexadecimal number by placing the character › "H" after the number (e.g. 170H). If the hexadecimal number begins › with A – F, then it must be preceded by a zero (i.e. You must enter A567H as OA567H). When an › operator, such as .AND. requires an integer, the BASIC module truncates › the fraction › portion of the number so it fits the integer format. We refer › to integers and line numbers as:

[integer] – [ln-num]

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.6 Constants

A constant is a real × number that ranges from +1E–127 to +.9999999 9E+127. A constant × can be an integer. × We refer to constants as: [const]

5.2.7 Operators

An operator performs a predefined × operation on variables and/or constants. Operators × require × either one or two operands. Typical two operand or dyadic operators include ADD (+), × SUBTRACT (–), MULTIPLY (*) × and DIVIDE(/). × We call × operators that require × only one operand, unary operators. Some typical × unary operators are SIN, COS and ABS.

5.2.8 Variables

A variable can be:

- a letter (e.g. A, X,I)
- a letter followed by a one dimensioned × expression, (e.g. J(4), GA(A + 6), I(10*SIN(X))
- a letter followed by a number followed by a one dimensioned expression × (e.g. × A1(8), P7(10*SIN(X)), W8(A + B).
- a letter followed by a number or letter × (e.g. AA, AC, XX, × A1, X3, G8) except × for the following × combinations: × CR, DO, lE, IF, IP, ON, PI, SP, TO, UI and UO.

We refer to variables that include × a one dimensioned expression [expr] as dimensioned or arrayed variables. × We refer × to variables × that contain a letter × or a letter and a number as scalar variables. Any variables entered in lower case are changed × to × upper case. We refer × to variables as:

[var].

The BASIC module × allocates variables × in a "static" manner. This means that × the first time a variable × is used, BASIC allocates a portion of memory (8 bytes) × specifically × for that variable. This memory cannot be de-allocated × on a variable to variable basis. This means that × if you execute × a statement (e.g. × Q 3), you cannot × tell BASIC × that the variable Q no longer exists to "free up" the 8 bytes of memory that belong to Q. You can × clear × the memory allocated × to variables by executing a CLEAR statement. × The CLEAR statement "frees" all memory allocated × to variables.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.8 Variables (continued)

Important: The BASIC Module requires less time to find a scalar variable because there is no expression › to evaluate. If you want to › run a program as › fast as possible, › use dimensioned › variables only when necessary. Use scalars › for intermediate variables and assign the final result to a dimensioned › variable. Also, put the most frequently › used variables › first. Variables defined first › require the least amount of time to locate.

5.2.9 Expressions

An expression › is a logical › mathematical › expression that involves operators (both unary and dyadic), › constants and variables. › Expressions are simple or complex, (e.g. 12*EXP(A)/100, H(1) › + 55, or (SIN(A)*SIN(A)+COS(A)* COS(A)/2). › A "stand alone" › variable [var] or constant [const] is also considered an expression. We refer › to expressions as:

[expr].

5.2.10 Relational Expressions

Relational › expressions involve › the operators › EQUAL (=), › NOT EQUAL (< >), GREATER THAN › OR › EQUAL TO › (> =), and LESS THAN OR › EQUAL TO (< =). You use them in control statements to test a condition › (i.e. IF A < 100 THEN...). Relational › expressions always require two operands. › We refer to relational expressions › as:

[rel expr].

5.2.11 System Control Values

The system control values include › the following:

- LEN (returns the length of your program).
- MTOP (the last memory location assigned to BASIC).

See the following › Section › 5.6.2 › titled, "System Control Values" for more information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.12 Argument Stack

The argument stack (A-stack) stores all constants that the BASIC Module is currently using. Operations such as add, subtract, × multiply × and divide always operate × on the first two numbers on × the argument × stack × and return × the result to the stack. × The argument × stack × is 203 bytes long. Each floating point × number placed on the stack requires × 6 bytes of storage. The argument stack can hold up to 33 floating × point numbers before × overflowing.

5.2.13 Control Stack

The control stack (C-stack) × stores all information × associated with loop control (i.e. DO-WHILE, DO-UNTIL, × FOR-NEXT, BASIC subroutines and "PUSHed" or "POPed" values). The control × stack is 157 bytes long. DO-WHILE and × DO-UNTIL loops × use 3 bytes of control × stack. FOR-NEXT loops × use 17 bytes. × The control × stack contains × enough space × for up to 9 nestings × of control × loops.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3 Description of Commands

The following sections list and describe the commands you can use with the BASIC Module.

5.3.1 Command: RUN

Action Taken: After you type RUN, all variables are set equal to zero, all BASIC evoked interrupts are cleared and program execution begins with the first line number of the selected program. The RUN command and the GOTO statement are the only way you can place the BASIC Module interpreter into the RUN mode from the COMMAND mode. You can terminate program execution at any time by typing a Control C on the console device.

Variations: Some BASIC interpreters allow a line number to follow the RUN command (i.e. RUN 100). The BASIC Module does not permit this variation on the RUN command. Execution begins with the first line number. To obtain a function similar to the RUN[ln num] command, use the GOTO[ln num] statement in the direct mode. See statement GOTO.

Example:

```
> 10 FOR I=1 TO 3
> 20 PRINT I
> 30 NEXT I
> 40 END
> RUN

  1
  2
  3

READY
>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.2 Command: CONT

Action › Taken: If › you stop a program by › typing a Control C on the console › device or by execution › of a STOP statement, › you can resume execution of the program by typing › CONT. If › you enter › a Control › C during the execution › of a CALL routine › you cannot › CONTinue. Between › the stopping and the re-starting of the program you › may display the values of variables › or change the › values of variables. However, you cannot CONTinue if › the program is › modified › during the STOP or after › an error.

Example:

```
> 10 FOR I=1 TO 10000
> 20 PRINT › I
> 30 PRINT › I
> 40 END
> RUN

  1
  2
  3
  4
  5– (TYPE CONTROL › C › ON CONSOLE)

STOP – IN LINE 20

READY
> PRINT I
  6

> I=10

> CONT
  10
  11
  12
```

5.3.3 Command: LIST

Action › taken: The LIST command prints the program to › the console device. Spaces are inserted after the line number, and before › and after statements. › This helps in the debugging of BASIC Module programs. You can › terminate the "listing" › of a program at anytime › by typing a Control › C on the console device. You can interrupt and continue › the listing using Control S and Control Q.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.3 Command: LIST (continued)

Variations: Two variations of the LIST command are possible with the BASIC Module.

They are:

1. LIST [ln num] (cr) × and
2. LIST [ln num] – [ln num] (cr)

The first × variation × causes the program to print × from the designated line × number (integer) to the end of the program. The second variation causes the program to print from the first line number (integer) × to the second line × number (integer).

Important: You must × separate × the two line × numbers with × a dash (–).

Example:

```
READY
> LIST
> 10 PRINT "LOOP PROGRAM"
> 20 FOR I=1 TO 3
> 30 PRINT I
> 40 NEXT I
> 50 END
READY
> LIST 30
> 30 PRINT I
> 40 NEXT I
> 50 END
READY
> LIST 20–40
> 20 FOR I=1 TO 3
> 30 PRINT I
> 40 NEXT I
```

5.3.4 Command: LIST# or LIST@

Action × taken: The LIST# × or LIST@ command lists the program to the device attached to the peripheral port (LIST device). × All comments that × apply to the LIST command apply to the LIST# or × LIST@ commands. We × include these commands to × permit you to make "hard copy × printouts" of a × program. × A configuration × plug sets the baud rate and must match your list × device (see section 3.2.4 titled, "Configuration Plugs"). × Also, you must configure × the peripheral × port parameters to × match your × particular × list device (see section 5.8.1 titled, × "Peripheral Port × Support × – Parameter Set × – CALL 30").

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.5 Command: NEW

Action taken: When you enter NEW(cr), × the × BASIC Module deletes the program that × is currently stored in RAM memory. In addition, all variables are set equal to ZERO, all strings and all BASIC evoked interrupts × are cleared. The REAL TIME CLOCK, × string allocation, and the internal stack pointer values are not affected. × In general, NEW (cr) is used to erase a program and × all × variables.

5.3.6 Command: NULL [integer]

Action taken: The NULL[integer] command × determines × how many NULL characters × (00H) the BASIC Module outputs after a carriage return. After initialization NULL 0. × Most printers × contain × a RAM buffer that eliminates the need to output NULL characters × after a carriage return.

5.3.7 Command: Control C

Action taken: This command stops × execution of the current program and returns the BASIC Module × to the COMMAND mode. In some cases you can continue × execution using a CONTinue. See × the explanation for CONTinue for more information.

5.3.7.1 Command: Disabling Control C

Action taken: This command disables × the Control C break × function. You can × do this × by setting × bit 48 (30H) to 1. Bit 48 is located in internal memory location 38 (26H). Set bit 48 by executing the following statement in a BASIC Module program or from the × command mode:

```
DBY(38) × DBY(38).OR.01H
```

When bit × 48 is set to 1, the Control C break × function for both LIST and RUN operations × is disabled. Cycling power returns Control C to normal operation × if it is disabled × from the command mode.

To re-enable the Control × C function, execute the following statement in a BASIC Module × program or × from the × command mode.

```
DBY(38) × DBY(38).AND.0FEH
```

CALL routines × do not check for this feature. If you enter a Control C while × using a CALL routine, the program stops if Control C is enabled or disabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.8 Command: Control S

Action › taken: This command interrupts › the scrolling › of a BASIC program › during the execution of a LIST command. › It also › stops output from the receiving › port if you are running a program. › In this case XOFF (Control S) operates as follows:

1. XOFF only operates on PRINT statements.
2. When received during a PRINT, › data › output › and program execution are suspended immediately.
3. When received at any other time, › program execution continues until a PRINT is › encountered. › At this time program execution › is suspended.
4. XON (Control › Q) is required › to resume program operation.

5.3.9 Command: Control Q

Action › taken: This command restarts a LIST command or PRINT output that › is interrupted by a Control S.

5.3.10 Overview of EPROM File Commands

Your BASIC Module › can › execute and SAVE up to 255 programs › in an EPROM. The › module › generates › all of the timing › signals needed to program most › EPROM devices. The programs › are › stored in sequence in the EPROM for › retrieval and execution. This sequential storing of programs is › called the EPROM FILE. The following commands allow › you to generate › and manipulate › the › EPROM FILE.

5.3.11 Commands: RAM and ROM [integer]

Action › taken: These two commands tell › the the BASIC Module interpreter whether to select the current program out of RAM or EPROM. The › current program is › displayed › during a LIST command and executed › when RUN is typed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.11 Commands: RAM and ROM [integer] (continued)

5.3.11.1 RAM

When you enter RAM(cr), the BASIC Module › selects › the current program from › RAM MEMORY.

Important: RAM › space › is limited to 13 K bytes. › Use the following formula › to calculate the available › user RAM space:

LEN system › control › value which contains › current › RAM program length

+# bytes allocated › for strings › (first › value in the STRING instruction)

+6 * each array › size › + 1 › (asterisk = › multiply)

+8 * each variable › used (including › each array name)

+1024 – number of › bytes reserved › for BASIC
V

Available › user › RAM= MTOP–V

5.3.11.2 ROM

When you enter ROM [integer], › the BASIC Module › selects the current program out › of EPROM memory. If › no integer › is typed after the ROM command (i.e. › ROM) the module defaults › to ROM 1. › Since the programs are › stored › in sequence › in EPROM, the integer › following the ROM command › selects › which program the user wants to › run or list. If you attempt › to select a program that does not exist › (e.g. you type in ROM 8 and only › 6 programs are stored › in the EPROM) the message ERROR: PROM MODE is › displayed.

The module › does not transfer › the program from EPROM to RAM when the › ROM mode › is › selected. › If you attempt to alter a program in the ROM mode, by typing in a line › number, the message ERROR: PROM MODE displays. › The XFER command allows › you to transfer a program from › EPROM to RAM for › editing › purposes. › You get no error message if › you attempt to edit a line of ROM program.

Important: When › you transfer programs from › EPROM to RAM you lose › the previous › RAM contents.

Since the ROM command does NOT transfer a program to RAM, it › is possible to have different › programs in › ROM and RAM simultaneously. You can › move back › and forth between the two modes when in command mode. If you are in run mode you can change › back › and forth using CALLS 70, 71 and 72. You can also use all of the RAM memory for variable storage if the program is stored in EPROM. The › system control value – MTOP always refers › to RAM. The system control value, LEN, refers › to the currently › selected program in RAM or › ROM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.12 Command: XFER

Action › taken: The XFER (transfer) › command transfers the current selected › program in › EPROM to › RAM and › then selects the RAM mode. If › you type › XFER while › the BASIC Module › is in the RAM mode, › the › program stored in RAM is › transferred back into › RAM and › the RAM mode › is › selected. After the XFER command executes, you can edit › the program in the same way you edit any RAM program.

Important: The XFER command › clears existing RAM programs.

5.3.13 Command: PROG

Important: Before › you attempt to program a › PROM, read the PROG, PROG1 › and › PROG2 sections of this chapter. Some PROG options exclude › the use of others.

Action › taken: The PROG command › programs › the resident EPROM with the current › program. › The current selected program may reside › in either RAM or EPROM. See Section 3.4, titled "Installing › the User Prom", for › additional information › concerning EPROM's.

Important: Be › sure › you have selected the program you want to save before › using the PROG command. Your › module does not automatically copy the RAM program to › ROM. You must also disable interrupts prior to the PROG, PROG1 › or PROG2 commands using › CALL 8 › and enable interrupts › after the PROM is › burned using › CALL 9. › If an error › occurs during EPROM programming, the › message ERROR PROGRAMMING is › displayed. When this › error occurs:

- previously › stored programs may or › may not › be accessible.
- you cannot › store › additional programs on this PROM.

After you type › PROG(cr), the BASIC Module displays › the number in the EPROM FILE › the › program occupies. Programming › can › take up to 12 minutes › to complete depending › on the length › of the program (51 seconds per K bytes of program).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.13 Command: PROG (continued)

Example:

```
> LIST
10 FOR I=1 TO 10
20 PRINT × I
30 NEXT I
40 END
READY
> CALL 8 :REM DISABLE INTERRUPTS
> PROG
12
> READY CALL 9 :REM ENABLE × INTERRUPTS
> ROM 12
READY
> LIST 10 FOR I=1 TO 10
20 PRINT × I
30 NEXT × I
40 END
READY
>
```

In this × example, the program just placed in the EPROM is × the 12th program stored.

Important: If you exceed × the available PROM space, × you cannot continue × programming until it is erased. In some cases you can alter × the previously stored programs. × Be sure × to use CALL 81 × to determine memory space prior to burning. See section × 5.3.13.1 × below.

5.3.13.1 User PROM Check × and × Description – CALL 81

Use CALL 81 × in command mode before burning a program into PROM × memory. This × CALL:

- determines × the number of × PROM programs.
- determines × the number of × bytes left × in PROM.
- determines × the number of × bytes in the × RAM program.
- prints × a message telling if enough space is available in × PROM for the RAM program.
- checks × for × a valid PROM if × it contains × no × program.
- prints × a good or × bad PROM message. A bad PROM message with an address of 00xx × indicates an incomplete program.

No × PUSHes or POPs are × needed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.14 Command: PROG1

Action taken: › You can use the PROG1 › command to program the resident EPROM with › baud rate information. › When the module is "powered-up" › the module reads this information and › initializes › the program port with the stored baud rate. The › "sign-on" message is sent to the › console immediately after › the module › completes its reset sequence. › If the baud rate on › the console device is changed you must program › a new EPROM to make › the module compatible with the new console.

If › you have › already used a PROG2 command, you cannot › use the PROG 1 command.

5.3.15 Command: PROG2

Action taken: › The PROG2 › command functions the same as the PROG1 command › except for the following. Instead of "signing-on" and entering the command › mode, the module immediately › begins executing the first program stored in the resident EPROM, if no program › is stored in RAM. Otherwise it › executes the RAM program. › You can use the PROG2 command to RUN a program on power-up without › connecting to a console. › Saving PROG2 information is the same as typing a ROM 1, RUN command sequence. This feature › also allows you to write a special initialization › sequence in BASIC and › generate a custom "sign-on" message for specific applications.

If › you have › already used a PROG1 command, › you cannot › use the PROG 2 command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3.15 Command: PROG2 (continued)

Figure 5.1
Flow Chart of Power-up Operation

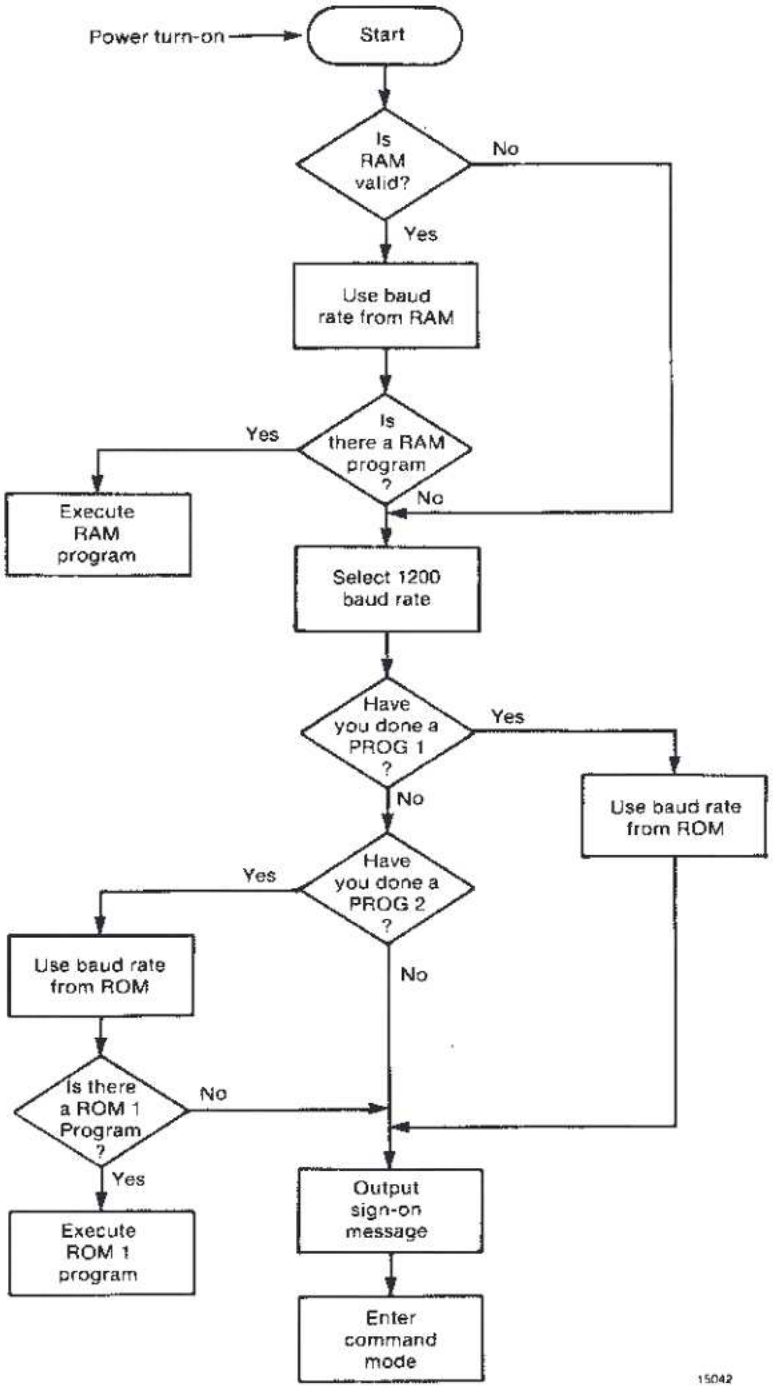


Figure 5.1 shows BASIC Module operation from a power-up condition using PROG1 or PROG2; or battery backed RAM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4 Description of Statements

The › following sections list and describe the statements › you can › use with the BASIC Module.

5.4.1 Statement: CALL [integer]

Mode: › COMMAND AND/ OR RUN
Type: › CONTROL

You › use the CALL [integer] statement to call specially › written BASIC Module application programs. Specific › call numbers are › defined later in this chapter.

5.4.2 Statement: CLEAR

Mode: › COMMAND AND/OR RUN
Type: › CONTROL

The › CLEAR statement › sets all variables equal to 0 and resets all BASIC › evoked interrupts and stacks. This means that after › the CLEAR statement is executed an ONTIME statement › must be executed before the module acknowledges the › internal timer interrupts. ERROR trapping using the ONE › RR statement also does not occur › until an ONERR[integer] › statement is executed. The CLEAR statement › does not affect › the real time clock › that is enabled by › the › CLOCK1 statement. › CLEAR also does not reset the memory that has been allocated › for strings, › so it is not necessary › to enter › the STRING [expr], › [expr] statement › to re-allocate › memory for strings after the CLEAR › statement › is executed. In general, › CLEAR is used › to "erase" › all variables.

5.4.3 Statement: CLEARI (clear interrupts)

Mode: › COMMAND AND/OR RUN
Type: › CONTROL

The › CLEARI statement clears all of the BASIC evoked interrupts. › The ONTIME interrupt › disables after the CLEARI statement › executes. CLEARI does not affect › the real time clock › enabled › by the CLOCK1 statement. You can use this statement › to selectively › DISABLE ONTIME interrupts during specific › sections of your BASIC › program. You must › execute the › ONTIME statement again before the specific interrupts enable.

Important: When the CLEARI › statement is LISTED it appears › as CLEAR › I.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.4 Statement: CLEARS

Mode: × COMMAND/RUN
Type: CONTROL

The CLEARS statement resets × all of the module's × stacks. × The control, × argument and internal × stacks all reset to their initialization values. You can use this × command to reset the stack if an error × occurs in a subroutine.

Important: When the × CLEARS statement is LISTed it appears as CLEAR S.

5.4.5 Statements: CLOCK1 and CLOCK0

Mode: COMMAND AND/OR × RUN
Type: CONTROL

CLOCK1

The CLOCK1 statement enables the real time clock × feature × resident on the BASIC Module. × The special × function operator TIME is incremented × once every × 5 milliseconds after the CLOCK1 statement is executed. The CLOCK1 STATEMENT uses an internal TIMER to generate an interrupt once every 5 milliseconds. × Because of this, the special × function operator TIME has a resolution × of 5 milliseconds. The special × function operator × TIME counts × from 0 to 65535.995 seconds. × After reaching × a count of 65535.995 seconds × TIME overflows × back to a count of zero. The interrupts associated × with the CLOCK1 statement × cause the module programs to × run at about 99.6% of normal speed. × That means that the interrupt handling for × the REAL TIME × CLOCK feature uses about .4% of × the total CPU time.

CLOCK0

The CLOCK0 (zero) × statement × disables × or "turns × off" the real time clock feature. After CLOCK0 is executed, × the special function operator TIME no longer × increments. CLOCK0 is × the only × module statement that can disable the real time clock. CLEAR and × CLEARI do NOT disable the real time clock, only × its associated ONTIME interrupt.

Important: CLOCK1 and CLOCK0 are independent of the × wall clock.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.6 Statements: DATA – READ – RESTORE

Mode: RUN
Type: Assignment

DATA

DATA specifies expressions that you can retrieve using a READ statement. If multiple expressions per line are used, you MUST separate them with a comma.

Example:

```
> 10 × DATA    10,ASC(A), ASC(B), 35.627
```

Important: You cannot × use the CHR() operator × in a DATA statement.

READ

READ retrieves the expressions × that are specified in the DATA statement and assigns × the value of the expression to the variable in the READ statement. × The READ statement is × always followed × by one or more variables. If more than one variable × follows × a READ statement, they are separated by a comma.

RESTORE

RESTORE "resets" × the internal read pointer × to the beginning of the data so that it may be read again.

Example:

```
> 10 FOR I=1 TO 3
> 20 READ A,B
> 30 PRINT A,B
> 40 NEXT I
> 50 RESTORE
> 60 READ A,B
> 70 PRINT A,B
> 80 DATA 10,20, × 10/2,20/2,SIN(PI),COS(PI)
> RUN
```

```
 10   20
  5   10
  0   –1
 10   20
```

Every time × a READ statement × is encountered the next consecutive expression × in the DATA statement × is evaluated and assigned × to the variable in the READ statement. × You can place DATA statements anywhere within a program. × They are not executed × and do not cause an error. DATA statements are considered to be chained × together and appear to be one large DATA statement. If at anytime all the data is read and another READ statement is executed, × the program terminates and the message ERROR: NO × DATA × – IN LINE XX × prints to the console device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.7 Statement: DIM

Mode: × COMMAND × AND/OR RUN
Type: × Assignment

DIM reserves storage for matrices. The storage × area is first assumed × to be zero. × Matrices × in the BASIC Module × may have only × one dimension and the size × of the dimensioned × array may not × exceed 254 elements. Once a variable is dimensioned in × a program it may not be re-dimensioned. × An attempt to re-dimension × an array causes an ARRAY SIZE × ERROR. If an arrayed × variable is used that is not dimensioned by the DIM statement, × BASIC assigns a default × value of 10 to the array size. All arrays are set equal × to zero × when the RUN command, NEW command × or the × CLEAR × statement is executed. The number of bytes allocated × for an array is 6 times × the array size × plus 1. The array × A(100) requires × 606 bytes of storage. Memory × size usually limits the size of a dimensioned × array.

Variations: More than one variable × can be dimensioned × by a single DIM statement.

Example:

```
> 10 DIM A(25), B(15), A1(20)
```

Example: Default error on attempt to re-dimension × array

```
> 10 A(5)10 –BASIC ASSIGNS DEFAULT OF 10 TO ARRAY SIZE HERE
> 20 DIM A(5) –ARRAY CANNOT BE RE–DIMENSIONED
> RUN

ERROR: ARRAY SIZE – IN LINE × 20
20      DIM × A(5)
------ X
```

5.4.8 Statements: DO – UNTIL [rel expr]

Mode: × RUN
Type: × CONTROL

The DO – UNTIL × [rel expr] instruction provides a means of "loop control" × within a module × program. × All statements between × the DO and × the UNTIL [rel expr] × are executed until the relational × expression following the UNTIL statement is TRUE. × You may nest × DO – UNTIL loops.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.8 Statements: DO – UNTIL [rel expr] (continued)

Examples:

SIMPLE DO-UNTIL

```
> 10 A=0
> 20 DO
> 30 A=A+1

> 40 PRINT A
> 50 UNTIL A=4
> 60 PRINT "DONE"
> 70 END
> RUN

  1
  2
  3
  4

DONE

READY
>
```

NESTED DO-UNTIL

```
> 10 DO
> 20 A=A+1
> 30 DO
> 40 B=B+1
> 50 PRINT A,B,A,*B
> 60 UNTIL B=3
> 70 B=0
> 80 UNTIL A=3
> 90 END
> RUN

  1 1 1
  1 2 2
  1 3 3
  2 1 2
  2 2 4
  2 3 6
  3 1 3
  3 2 6
  3 3 9

  READY
  >
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.9 Statements: DO – WHILE [rel expr] (continued)

Mode: RUN
Type: CONTROL

The × DO – WHILE [rel expr] instruction × provides a means of "loop control" within a module program. The operation of this statement is similar to the DO – × UNTIL [rel expr] except that all × statements between × the DO and the × WHILE [rel expr] are executed as long as the relational × expression following the WHILE statement × is true. You can nest × DO – WHILE and DO – × UNTIL statements.

Examples:

```
SIMPLE DO-WHILE

> 10 DO
> 20 A=A+1
> 30 PRINT A
> 40 WHILE At4
> 50 PRINT "DONE"
> 60 END
> RUN



1
2
3
4
DONE

READY
>
```

```
NESTED
DO-WHILE - DO-UNTIL

> 10 DO
> 20 PRINT A=A+1
> 25 DO
> 30 B=B+1
> 40 A,B,A*B
> 50 WHILE B< >3
> 60 B=0
> 70 UNTIL A=3
> 80 END
> RUN
1 1 1
1 2 2
1 3 3
2 1 2
2 2 4
2 3 6
3 1 3
3 2 6
3 3 9

READY
>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.10 Statement: END

Mode: RUN
Type: CONTROL

The END statement › terminates program execution. × The continue comm and, CONT does › not operate › if the END statement › is used to terminate execution. A CAN'T CONTINUE ERROR prints to the console. The last statement in a module program › automatically terminates program execution › if you do not use an end statement. › You should always › use an END statement › to terminate a program.

Examples:

END STATEMENT TERMINATION

```
> 10 FOR I=1 TO 4
> 20 GOSUB 100
> 30 NEXT I
> 40 END
> 100 PRINT › I
> 110 RETURN
> RUN

1
2
3
4

READY
>
```

Variations: › None

5.4.11 Statements: FOR – TO – (STEP) – NEXT

Mode: RUN
Type: CONTROL

Use the › FOR – TO – (STEP) – NEXT statements › to set up and control loops.

Example:

```
> 5 B=0: › C=10 : D=2
> 10 FOR A= › B TO C STEP D
> 20 PRINT › A
> 30 NEXT A
> 40 END
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.11 Statements: FOR – TO – (STEP) – NEXT (continued)

Since B=0, C=10 and D=2, › the PRINT statement › at line › 20 executes 6 times. The values of "A" printed are 0, 2, 4, 6, 8 and 10. "A" represents the name of › the index or loop counter. The value of "B" is the starting value of the index. The value › of "C" is the limit › value of the index and the › value of "D" is the increment › to the index. If the STEP statement and the value "D" are omitted, › the increment › value › defaults › to 1, therefore, STEP is an optional statement. › The NEXT statement › adds the value of "D" to the index. › The index is then compared to the value of "C", the limit. › If the index is less than or equal to the limit, control transfers › back to the statement › after › the FOR statement. › Stepping "backwards" › (FOR › I= 100 TO 1 STEP–1) is › permitted in the BASIC Module. You may not › omit the index from the › NEXT statement in the module (The NEXT statement › is always followed by the appropriate variable). You may nest FOR-NEXT loops up to 9 times.

Examples:

```
> 10 FOR I=1 TO 4
> 20 PRINT I,
> 30 NEXT I
> 40 END
> RUN
  1 2 3 4


READY
>
```

```
> 10 FOR I=O TO 8 STEP 2
> 20 PRINT I
> 30 NEXT I
> 40 END
> RUN
0
2
4
6
8

READY
>
```

5.4.12 Statements: GOSUB [ln num] – RETURN

Mode: RUN
Type: CONTROL

GOSUB

The GO SUB [ln › num] statement › causes › the BASIC Module › to transfer › control of the program directly to the line number ([ln num]) following the GOSUB statement. › In addition, › the GOSUB statement saves the location › of the statement following › GOSUB on the control stack so that you can perform a RETURN statement to return › control › to the statement › following › the most recently › executed GO SUB STATEMENT.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.12 Statements: GOSUB [ln num] – RETURN (continued)

RETURN

Use this statement × to "return" × control to the statement following × the most recently executed GO SUB STATEMENT. Use one return for each GOSUB to × avoid overflowing × the C-STACK. This × means that × a subroutine × called × by the GO SUB statement × can call another subroutine with another GOSUB statement.

Examples:

SIMPLE SUBROUTINE

```
> 10 FOR I=1 TO 5
> 20 GOSUB 100
> 30 NEXT I
> 40 END
> 100 PRINT I
> 110 RETURN
> RUN
  1
  2
  3
  4
  5

READY
>
```

NESTED SUBROUTINES

```
> 10 FOR I=1 TO 3
> 20 GOSUB 100
> 30 NEXT I
> 40 END
> 100 REM USER SUBROUTINE HERE
> 105 PRINT I,
> 110 GOSUB 200
> 200 PRINT I,I*I
> 210 RETURN
> RUN

  1  1
  2  4
  3  9

READY
>
```

5.4.13 Statement: GOTO [ln num]

Mode: COMMAND AND/OR × RUN
Type: CONTROL

The GOTO [ln num] statement × causes BASIC to transfer control directly to the line number ([ln × num]) following the GOTO statement.

Example:

```
> 50 GOTO 100
```

If × line 100 exists, × this statement causes × execution × of the program to resume at line 100. If line number 100 does not exist the message ERROR: INVALID × LINE × NUMBER × is printed to the console device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.13 Statement: GOTO [ln num] (continued)

Unlike the RUN command the GOTO statement, if executed in the COMMAND mode, does › not clear the variable storage space or interrupts. › However, if › the GOTO statement is executed in the COMMAND mode after a line › is edited the module clears › the variable storage › space and all BASIC evoked interrupts. This is necessary because › the variable storage and the BASIC program reside in the same RAM memory. Because › of this editing a program can destroy variables.

5.4.14 Statements: ON [expr] GOTO [ln num], [ln num],...[ln num], ON [expr] GOSUB[ln num], [ln num],...[ln num]

Mode: RUN
Type: › CONTROL

The value › of the expression › following › the ON statement › is the number in the line list that control is transferred to.

Example:

> 10 ON Q GOTO 100,200,300

If Q is equal to 0, control › is transferred › to line number 100. › If Q is equal to 1, control is transferred to line number 200. If Q is equal to 2, GOTO line 300, etc. All comments that apply to GOTO and › GOSUB apply › to the ON statement. If Q is less than › ZERO a BAD ARGUMENT ERROR › is › generated. › If Q is greater than the line number list › following › the GOTO or GOSUB statement, a BAD SYNTAX › ERROR › is generated. The ON statement › provides "conditional branching" › options within the constructs of a BASIC Module › program.

5.4.15 Statements: IF – THEN – ELSE

Mode: RUN
Type: › CONTROL

The IF statement sets up a conditional test. The general form of the IF – THEN – › ELSE statement follows:

[ln num] IF [rel expr] THEN valid › statement › ELSE valid statement

Example:

> 10 IF › A=100 THEN A=0 ELSE A=A+1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.15 Statements: IF – THEN – ELSE (continued)

Upon execution › of line 10 IF A is › equal to 100, THEN A is assigned a value of 0. IF A does not equal 100, A is › assigned a value of A+1. If you want to › transfer › control to different line numbers using the IF statement, you may omit › the GOTO statement. › The following examples give › the same results:

```
> 20 IF › INT(A) › < 10 THEN GOTO › 100 › ELSE GOTO 200
                              or
> 20 IF › INT(A) › < 10 THEN 100 ELSE 200
```

You can replace › the THEN statement › with any valid › BASIC Module statement, as shown below:

```
> 30 IF › A< > 10 THEN PRINT › A › ELSE 10

> 30 IF › A< > 10 PRINT › A › ELSE 10
```

You may execute › multiple › statements following › the THEN or ELSE if › you use a colon to separate them.

Example:

```
> 30 IF › A< > 10 THEN PRINT › A:GOTO › 150 › ELSE 10

> 30 IF › A< > 10 PRINT › A:GOTO 150 ELSE 10
```

In these examples, › if › A does not equal 10 then both PRINT A and GOTO 150 › are executed. › If A10, then control passes to 10.

You may omit the ELSE statement. If you omit the ELSE statement control passes to the next statement.

Example:

```
> 20 IF › A10 THEN 40

> 30 PRINT › A
```

In this example, if A equals › 10 then control › passes to line number 40. If A does not equal 10 line › number 30 › is executed.

5.4.16 Statement: INPUT

Mode: RUN
Type: › INPUT/OUTPUT

The INPUT statement allows you to enter data from the console during program › execution. You may assign data to one or more variables with a single › input statement. › You must separate the variables › with a comma.

Example:

```
> INPUT A,B
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.16 Statement: INPUT (continued)

Causes a question > mark (?) > to print on the console device. > This prompts > you to input > two numbers separated > by a comma. If you do not enter > enough data, the module prints > TRY AGAIN on the console device.

Example:

```
> 10 INPUT A,B
> 20 PRINT A,B
> RUN

?1

TRY AGAIN

?1,2
1    2

READY
```

You can > write the INPUT statement > so that > a descriptive prompt tells you what to > enter. The message to be printed > is placed in quotes after the INPUT statement. > If a comma appears > before the first variable on the input list, > the question mark prompt > character is not displayed.

Examples:

```
> 10 INPUT"ENTER A NUMBER" A
> 20 PRINT SQR(A)
> 30 END
> RUN

ENTER A NUMBER
?100
10
```

```
> 10 INPUT"ENTER A NUMBER–",A
> 20 PRINT SQR(A)
> 30 END
> RUN

ENTER A NUMBER–100
10
```

You can > also assign strings with an INPUT statement. Strings > are always terminated with a carriage > return (cr). If more than one string input is requested > with a single INPUT statement, the module prompts you > with > a question mark.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.16 Statement: INPUT (continued)

Examples:

```
> 10 STRING 110,10
> 20 INPUT "NAME:",$(1)
> 30 PRINT "HI",$(1)
> 40 END
> RUN

NAME: SUSAN
HI SUSAN


READY
```

```
> 10 STRING 110,10
> 20 INPUT "NAMES:",$(1),$(2)
> 30 PRINT "HI",$(1)," AND ",$(2)
> 40 END
> RUN

NAMES: BILL
?ANN
HI BILL AND ANN


READY
```

You can assign × strings and variables with a single × INPUT statement.

Example:

```
> 10 STRING 100,10
> 20 INPUT"NAME(CR), × AGE – ",$(1),A
> 30 PRINT "HELLO ",$(1),", × YOU ARE ",A,"YEARS × OLD"
> 40 END
> RUN

NAME(CR), AGE – × FRED
?15
HELLO FRED, × YOU × ARE × 15 × YEARS OLD

READY
>
```

5.4.17 Statement: LD@[expr]

Mode: COMMAND AND/OR × RUN

This statement, along with CALL 77, allows you to save/retrieve variables × to/from a protected × area of memory. This protected × area is not zeroed on power-up × or when the RUN command is issued. The LD@ statement takes × the variable stored at address × [expr] and moves it × to the top of the argument stack. × For more information on protecting variables, × see section 5.11.6, "Protected Variable Storage × – CALL 77".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.18 Statement: LET

Mode: COMMAND AND/OR › RUN
Type: › ASSIGNMENT

Use the › LET statement › to assign a variable to the value of an expression. › The general › form of › LET is:

LET [var] = [expr]

Examples:

LET A =10*SIN(B)/100 › or

LET A = A+1

Note that › the = sign › used in the LET statement › is not an equality operator. It is a "replacement" operator. The statement › should › be read A is › replaced by A plus one. The word LET is always optional, › (i.e. LET A =2 is the same as A › =2).

When LET is › omitted › the LET statement is called › an IMPLIED LET. We use › the word LET to refer › to both the LET statement › and the › IMPLIED LET statement.

Also use the LET statement to assign the string › variables:

LET $(1)="THIS › IS A STRING" or

LET $(2)=$(1)

Before › you can assign strings you must execute the STRING [expr], [expr] statement or else a MEMORY ALLOCATION ERROR occurs. See the following › section › 5.4.31 titled, "STRING".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.19 Statement: ONERR [ln num)

Mode: RUN
Type: › CONTROL

The ONERR[ln num] › statement lets you handle › arithmetic errors, if they occur, during program execution. › Only ARITH. OVERFLOW, ARITH. UNDERFLOW, › DIVIDE BY ZERO and BAD ARGUMENT errors › are "trapped" › by the ONE RR › statement. › All other errors are not "trapped". › If an arithmetic error › occurs after › the ONE RR statement › is executed, › the module interpreter passes control to the line number following the ONERR[ln num] statement. You handle the error condition in a manner suitable to the particular › application. The ONERR command › does not trap › bad data › entered during an input instruction. › This yields › a "TRY AGAIN" › message › or "EXTRA IGNORED" › message. › See Chapter › 9 for an explanation of errors.

With the › ONERR[ln num] statement, › you have the option of determining what › type of error › occurred. Do this › by examining external › memory location 257 (101H) › after the error › condition › is trapped.

The error codes are:

ERROR › CODE =10 › – DIVIDE BY ZERO

ERROR › CODE =20 › – ARITH. OVERFLOW

ERROR › CODE › =30 – ARITH. UNDERFLOW

ERROR CODE =40 – BAD ARGUMENT

You can examine this location › by using an XBY(257) statement.

Example:

```
PRINT XBY(257)
      or
E XBY(257)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.20 Statement: ONTIME [expr],[ln num]

Mode: RUN
Type: CONTROL

Your BASIC Module can process a line in milliseconds while the timer/counters on the microprocessor operate in microseconds. You must use the ONTIME [expr], [ln num] statement because of this incompatibility between the timer/counters on the microprocessor and the BASIC Module. The ONTIME statement generates an interrupt every time the special function operator, TIME, is equal to or greater than the expression following the ON TIME statement. Only the integer portion of TIME is compared to the integer portion of the expression. This comparison is performed at the end (CR or:) of each line of BASIC. The interrupt forces a GOSUB to the line number [ln num]) following the expression ([expr]) in the ONTIME statement.

Important: The ONTIME statement does not interrupt an input command or a CALL routine. Since the ONTIME statement uses the special function operator, TIME, you must execute the CLOCK1 statement for ONTIME to operate. If CLOCK1 is not executed the special function operator, TIME, does not increment.

You can generate periodic interrupts by executing the ONTIME statement again in the interrupt routine:

Example:

```
> 10 TIME=0 : CLOCK1 : ONTIME 2,100: DO
> 20 WHILE TIME < 10 : END
> 100 PRINT "TIMER INTERRUPT AT –",TIME,"SECONDS"
> 110 ONTIME TIME+2,100 : RETI
 > RUN

TIMER INTERRUPT AT – 2.045 SECONDS
TIMER INTERRUPT AT – 4.045 SECONDS
TIMER INTERRUPT AT – 6.045 SECONDS
TIMER INTERRUPT AT – 8.045 SECONDS
TIMER INTERRUPT AT – 10.045 SECONDS

READY
```

The terminal used in this example runs at 4800 baud. This baud rate allows about 45 milliseconds to print the message TIMER INTERRUPT AT –" ". The resulting printed time is 45 milliseconds greater than the planned time.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.20 Statement: ONTIME [expr], [ln num] (continued)

If you do not want this delay, you should assign a variable to the special function operator, TIME, at the beginning of the interrupt routine.

Example:

```
> 10 TIME=0 : CLOCK1 : ONTIME 2,100: DO
> 20 WHILE TIME < 10: END
> 100 A=TIME
> 110 PRINT "TIMER INTERRUPT AT –",A,"SECONDS"
> 120 ONTIME A+2,100 : RETI
> RUN

TIMER INTERRUPT AT – 2 SECONDS
TIMER INTERRUPT AT – 4 SECONDS
TIMER INTERRUPT AT – 6 SECONDS
TIMER INTERRUPT AT –8 SECONDS
TIMER INTERRUPT AT – 10 SECONDS

READY
```

Important: You must exit the ONTIME interrupt routine with a RET I statement. Failure to do this "locks-out" all future interrupts.

The ONTIME statement eliminates the need for you to "test" the value of the TIME operator periodically throughout the BASIC program.

5.4.21 Statement: PRINT or P.

Mode: COMMAND and/or RUN
Type: INPUT/OUTPUT

The PRINT statement directs the BASIC Module to output to the console device. You may print the value of expressions, strings, literal values, variables or text strings. You may combine the various forms in the print list by separating them with commas. If the list is terminated with a comma, the carriage return/line feed is suppressed. P. is a "shorthand" notation for PRINT.

Examples:

```
> PRINT 10*10,3*3  >  PRINT "MCS–51"  >  PRINT 5,1E3
  100   9              MCS–51             5  1000
```

Important: Values are printed next to one another with two intervening blanks. A PRINT statement with no arguments sends a carriage return/line feed sequence to the console device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.22 Special Print Formatting Statements

The following sections list and describe the special print formatting statements.

5.4.22.1 PRINT TAB([expr])

Use the TAB([expr]) function in the PRINT statement to cause data to print out in exact locations on the output device. TAB([expr]) tells the BASIC Module which position to begin printing the next value in the print list. If the printhead or cursor is on or beyond the specified TAB position, the module ignores the TAB function.

Example:

```
> PRINT TAB(5),"X",TAB(10),"Y"
      X       Y
```

5.4.22.2 PRINT SPC([expr])

Use the SPC([expr]) function in the PRINT statement to cause the BASIC Module to output the number of spaces in the SPC argument.

Example:

```
>  PRINT A,SPC(5),B
```

Use the above statement to place an additional 5 spaces between the A and B in addition to the two that would normally print.

5.4.22.3 PRINT CR

Use CR in a PRINT statement to force a carriage return, but no line feed. You can use CR to create one line on a CRT device that is repeatedly updated.

Example:

```
>  10 FOR I=1 TO 1000
> 20 PRINT I,CR,
> 30 NEXT I
```

The above example causes the output to remain on one line only. No line feed is ever sent to the console device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.22 Special Print Formatting Statements (continued)

5.4.22.4 PRINT USING (special characters)

Use the USING function to tell the BASIC Module what format to use when displaying printed values. The module "stores" the desired format after the USING statement is executed. All outputs following a USING statement are in the format evoked by the last USING statement executed. You do not need to execute the USING statement within every PRINT statement unless you want to change the format. U. is a "shorthand" notation for USING.

Important: The USING statement applies to numbers following it until another USING statement is encountered.

5.4.22.5 PRINT USING(Fx)

This forces the BASIC Module to output all numbers using the floating point format. The value of x determines how many significant digits are printed. If x equals 0, the module does not output any trailing zeros, so the number of digits varies depending upon the number. The module always outputs at least 3 significant digits even if x is 1 or 2. The maximum value for x is 8.

Example:

```
> 10 PRINT USING(F3),1,2,3
> 20 PRINT USING(F4),1,2,3
> 30 PRINT USING(F5),1,2,3
> 40 FOR I=10 TO 40 STEP 10
> 50 PRINT I
> 60 NEXT I
> RUN

 1.00 E 0     2.00 E 0     3.00 E 0
 1.000 E 0     2.000 E 0    3.000 E 0
 1.0000 E 0     2.0000 E 0   3.0000 E 0
 1.0000 E+1
 2.0000 E+1
 3.0000 E+1
 4.0000 E+1
 READY
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.22 Special Print Formatting Statements (continued)

5.4.22.6 PRINT USING(#.#)

This forces the module to output all numbers using an integer and/or fraction format. The number of "#"'s before the decimal point represents the number of significant integer digits that are printed and the number of "#"'s after the decimal point represents the number of digits that are printed in the fraction. Omit the decimal point to print integers only. You may use the abbreviation U for USING. USING(###.###), USING(######) and USING(######.##) are all valid in the BASIC Module. The maximum number of "#" characters is 8. If the module cannot output the value in the desired format (usually because the value is too large) a question mark (?) is printed to the console device. BASIC then outputs the number in the FREE FORMAT described below (refer to section 5.4.22.7).

Example:

```
> 10 PRINT USING(##.##),1,2,3
> 20 FOR I=1 TO 120 STEP 20
> 30 PRINT I
> 40 NEXT I
> 50 END
> RUN
  1.00     2.00     3.00
  1.00
  21.00
  41.00
  61.00
  81.00
? 101

READY
```

Important: The USING(Fx) and the USING(#.#) formats always "align" the decimal points when printing a number. This makes displayed columns of numbers easy to read.

5.4.22.7 PRINT USING(0)

This argument lets the BASIC Module determine what format to use. If the number is between +/–99999999 and +/–.1, BASIC displays integers and fractions. If it is out of this range, BASIC uses the USING (F0) format. Leading and trailing zeros are always suppressed. After reset, the module is placed in the USING(0) format.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.22 Special Print Formatting Statements (continued)

5.4.22.8 Reset Print Head Pointer – CALL 99

You can use CALL 99 when printing out wide forms to reset the internal print read/character counter and prevent the automatic CR/LF at character 79. You must keep track of the characters in each line.

You can solve this problem in revision A or B modules using DBY(1 6H) 0

Example:

```
> 10 REM THIS PRINTS TIME BEYOND 80TH COLUMN
> 20 PRINT TAB (79),
> 30 CALL 99 40 PRINT TAB (41), "TIME –",
> 50 PRINT H , ":" , M , ":" ,S
> 60 GOTO 20
> 70 END
```

5.4.23 Statement: PRINT# or P.#

Mode: COMMAND and/or RUN
Type: INPUT/OUTPUT

The PRINT# (PRINT @) or P.# statement functions the same as the PRINT or P. statement except that the output is directed to the list device instead of the console device. The baud rate and peripheral port parameters must match your device. See Chapter 3 Section 3.2.4 titled, "Configuration Plugs" and Chapter 5 Section 5.8.1 titled, "Peripheral Port Support – Parameter Set – CALL 30".

All comments that apply to the PRINT or P. statement apply to the PRINT# or P.# statement. P.# is a "shorthand" notation for PRINT#.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.24 Statements: PH0., PH1., PH0.#, PH1.#

Mode: COMMAND and/or RUN
Type: INPUT/OUTPUT

The PH0. and PH1. statements function the same as the PRINT statement except that the values are printed out in a hexadecimal format. The PH0. statement suppresses two leading zeros if the number printed is less than 255 (0FFH). The PH1. statement always prints out four hexadecimal digits. The character "H" is always printed after the number when PH0. or PH1. is used to direct an output. The values printed are always truncated integers. If the number printed is not within the range of valid integer (i.e. between 0 and 65535 (0FFFFH) inclusive), the BASIC Module defaults to the normal mode of print. If this happens no "H" prints out after the value. Since integers are entered in either decimal or hexadecimal form the statements PRINT, PH0., and PH1. are used to perform decimal to hexadecimal and hexadecimal to decimal conversion. All comments that apply to the PRINT statement apply to the PH0. and PH1. statements. PH0.# and PH 1.# function the same as PH0. and PH1. respectively, except that the output is directed to the list device instead of the console device. The baud rate and peripheral port parameters must match your device. See Chapter 3 Section 3.2.4 titled, "Configuration Plugs" and Chapter 5 Section 5.8.1 titled, "Peripheral Port Support – Parameter Set – CALL 30".

Examples:

> PH0.2*2	> PH1.2*2	> PRINT 99 H	> PH0.100
04H	0004H	153	64H
> PH0. 1000	> PH1. 1000	> P. 3E8H	> PH0.PI
3E8H	03E8H	1000	03H

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.25 Statement: PUSH[expr]

Mode: COMMAND AND/OR RUN
Type: ASSIGNMENT

The arithmetic expression, or expressions, following the PUSH statement are evaluated and then placed in sequence on the BASIC Module's ARGUMENT STACK. This statement, in conjunction with the POP statement, provides a simple means of passing parameters to assembly language routines. In addition, the PUSH and POP statements are used to pass parameters to BASIC subroutines and to "SWAP" variables. The last value PUSHed onto the ARGUMENT STACK is the first value POPed off the ARGUMENT STACK.

Variations: You can push more than one expression onto the ARGUMENT stack with a single PUSH statement. The expressions are followed by a comma: PUSH[expr],[expr],..... [expr]. The last value PUSHed onto the ARGUMENT STACK is the last expression [expr] encountered in the PUSH STATEMENT.

Examples:

```
SWAPPING        CALL
VARIABLES       ROUTINE
> 10 A=10       > 10 PUSH 3
> 20 B=20       > 20 CALL 10
> 30 PRINT A,B  > 30 POP W
> 40 PUSH A,B   > 40 REM PUSH VALUE ON STACK, CALL
> 50 POP A,B    > 50 REM CONVERSION, POP RESULT IN W
> 60 PRINT A,B  > 60 END
> 70 END
> RUN

10        20
20        10
READY
>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.26 Statement: POP[var]

Mode: COMMAND AND/OR RUN
TYPE: ASSIGNMENT

The value on top of the ARGUMENT STACK is assigned to the variable following the POP statement and the ARGUMENT STACK is "POPed" (i.e. incremented by 6). You can place values on the stack using either the PUSH statement or assembly language CALLS.

Important: If a POP statement executes and no number is on the ARGUMENT STACK, an A-STACK ERROR occurs.

Variations: You can pop more than one variable off the ARGUMENT stack with a single POP statement. The variables are followed by a comma (i.e. POP [var],[var], ... [var]).

Examples:

See PUSH statement above (section 5.4.25).

You can use the PUSH and POP statements to minimize GLOBAL variable problems. These are caused by the "main" program and all subroutines used by the main program using the same variable names (i.e. GLOBAL VARIABLES). If you cannot use the same variables in a subroutine as in the main program you can re–assign a number of variables (i.e. A=Q) before a GOSUB statement is executed. If you reserve some variable names just for subroutines (S1, S2) and pass variables on the stack as shown in the previous example, you can avoid any GLOBAL variable problems in the BASIC Module.

The PUSH and POP statements accept dimensioned variables A(4) and S1(12) as well as scalar variables. This is useful when large amounts of data must be PUSHed or POPed when using CALL routines.

Example:

```
> 40 FOR I=1 TO 64
> 50 PUSH I
> 60 CALL 10
> 70 POP A(I)
> 80 NEXT I
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.27 Statement: REM

Mode: COMMAND and/or RUN
Type: CONTROL – Performs no operation

REM is short for REMark. REM allows you to add comments to a program to make it easier to understand. If a REM statement appears on a line the entire line is used for the REM statement. You cannot terminate a REM statement using a colon (:), however, you can place a REM statement after a colon. This allows you to place a comment on each line.

Example:

```
> 10 REM INPUT ONE VARIABLE
> 20 INPUT A
> 30 REM INPUT ANOTHER VARIABLE
> 40 INPUT B
> 50 REM MULTIPLY THE TWO
> 60 Z=A*B
> 70 REM PRINT THE ANSWER
> 80 PRINT Z
> 90 END
> 10 INPUT A: REM INPUT ONE VARIABLE
> 20 INPUT B : REM INPUT ANOTHER VARIABLE
> 30 Z=A*B : REM MULTIPLY THE TWO
> 40 PRINT Z :REM PRINT THE ANSWER
> 50 END
```

The following example does NOT work because the entire line is interpreted as a REMark. The PRINT statement is not executed:

Example:

```
> 10 REM PRINT THE NUMBER: PRINT A
```

Important: REMark statements add time to program execution. Use them selectively or place them at the end of the program where they do not affect program execution speed. Do not use REMark statements in frequently called loops or subroutines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.28 Statement: RETI

Mode: RUN
Type: CONTROL

Use the RETI statement to exit from the ONTIME interrupts that are handled by a BASIC Module program. The RETI statement functions the same as the RETURN statement except that it also clears a software interrupt flag so interrupts can again be acknowledged. If you fail to execute the RETI statement in the interrupt procedure, all future interrupts are ignored.

5.4.19 Statement: ST@[expr]

Mode: COMMAND AND/OR RUN

This statement, along with CALL 77, allows you to save/retrieve variables to/from a protected area of memory. This protected area is not zeroed on power-up or when the RUN command is issued. The ST@ statement takes the argument on top of the argument stack and moves it to the address location specified by [expr]. For more information on protecting variables, see section 5.11.6, "Protected Variable Storage – CALL 77".

5.4.30 Statement: STOP

Mode: RUN
Type: CONTROL

The STOP statement allows you to break program execution at specific points in a program. After a program is STOPped you can display or modify variables. You can resume program execution with a CONTinue command. The purpose of the STOP statement is to allow for easy program "debugging" More details of the STOP-CONT sequence are covered in Section 5.3.2 titled, "Command: CONT".

Example:

```
> 10 FOR I=1 TO 100
> 20 PRINT I
> 30 STOP
> 40 NEXT I
> RUN

  1
  STOP – IN LINE 40

  READY
> CONT

  2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.4.30 Statement: STOP (continued)

Note that the line number printed out after execution of the STOP statement, is the line number following the STOP statement, not the line number that contains the STOP statement.

5.4.31 Statement: STRING

Mode: COMMAND and/or RUN
Type: CONTROL

The STRING [expr], [expr] statement allocates memory for strings. Initially, no memory is allocated for strings. If you attempt to define a string with a statement such as LET $(1)"HELLO" before memory is allocated for strings, a MEMORY ALLOCATION ERROR is generated. The first expression in the STRING [expr],[expr] statement is the total number of bytes you want to allocate for string storage. The second expression gives the maximum number of bytes in each string. These two numbers determine the total number of defined string variables.

The BASIC Module requires one additional byte for each string, plus one additional byte overall. The additional character for each string is allocated for the carriage return character that terminates the string. This means that the statement STRING 100,10 allocates enough memory for 9 string variables, ranging from $(0) to $(8) and all of the 100 allocated bytes are used. Note that $(0) is a valid string in the BASIC Module. Refer to the following Section 5.10 titled, "Description of String Operators" for further discussion of strings and example programs for string memory allocation.

Important: After memory is allocated for string storage, commands (e.g. NEW) and statements (e.g. CLEAR) cannot "de-allocate" this memory. Cycling power also cannot de-allocate this memory unless battery backup is disabled. You can de-allocate memory by executing a STRING 0,0 statement. STRING 0,0 allocates no memory to string variables.

Important: The BASIC Module executes the equivalent of a CLEAR statement every time the STRING [expr],[expr] statement executes. This is necessary because string variables and numeric variables occupy the same external memory space. After the STRING statement executes, all variables are "wiped-out". Because of this, you should perform string memory allocation early in a program (during the first statement if possible). If you re-allocate string memory you destroy all defined variables.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5 Description of Arithmetic, and Logical Operators and Expressions

The BASIC Module contains a complete set of arithmetic and logical operators. We divide the operators into two groups, dual operand (dyadic) operators and single operand (unary) operators.

5.5.1 Dual Operand (dyadic) Operators

The general form of all dual operand instructions is:

(expr) OP (expr), where OP is one of the following operators.

- + Addition Operator

  Example: PRINT 3+2
  5

- / Division Operator

  Example: PRINT 100/5
  20

- ** Exponentiation Operator

  The Exponentiation Operator raises the first expression to the power of the second expression. The maximum power to which you raise a number is 255.

  Example: PRINT 2**3
  8

- * Multiplication Operator

  Example: PRINT 3*3
  9

- – Subtraction Operator

  Example: PRINT 9–6
  3

- .AND. Logical AND Operator

  Example: PRINT 3.AND.2
  2

- .OR. Logical OR Operator

  Example: PRINT 1.OR.4
  5.

- .XOR. Logical EXCLUSIVE OR operator

  Example: PRINT 7.XOR.6
  1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5.1 Dual Operand (dyadic) Operators (continued)

5.5.1.1 Comments on logical operators .AND.,.OR. and .XOR.

These operators perform a BIT-WISE logical function on valid INTEGERS. This means both arguments for these operators are between 0 and 65535 (0FFFFH) inclusive. If they are not, the BASIC Module generates a BAD ARGUMENT ERROR. All non-integer values are truncated, not rounded. Use the following chart for bit manipulations on 16 bit values.

			Hex	Decimal
To set bit	0	.OR. VARIABLE WITH	0001H	1
	1		0002H	2
	2		0004H	4
	3		0008H	8
	4		0010H	16
	5		0020H	32
	6		0040H	64
	7		0080H	128
	10		0100H	256
	11		0200H	512
	12		0400H	1024
	13		0800H	2048
	14		1000H	4096
	15		2000H	8192
	16		4000H	16384
	17		8000H	32768
	Example: 10 A= A .OR. 0100H:REM SET BIT 10			
To clear bit	0	.AND. VARIABLE WITH	0FFFEH	65534
	1		0FFFDH	65533
	2		0FFFBH	65531
	3		0FFF7H	65527
	4		0FFEFH	65519
	5		0FFDFH	65503
	6		0FFBFH	65471
	7		0FF7FH	65407
	10		0FEFFH	65279
	11		0FDFFH	65023
	12		0FBFFH	64511
	13		0F7FFH	63487
	14		0EFFFH	61439
	15		0DFFFH	57343
	16		0BFFFH	49151
	17		7FFFH	32767
	Example: 10 A= A .AND. 0F7FFH: REM CLR BIT 13			
To clear all bits		.XOR. VARIABLE WITH ITSELF		
	Example: 10 A = A.XOR.A : REM CLR A			

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5.2 Unary Operators

We divide the unary operators into three groups; general purpose, log functions and trig functions.

5.5.2.1 General Purpose Operators

Following are the general purpose operators.

5.5.2.1.1 **ABS**([expr])

Returns the absolute value of the expression.

Examples:

```
> PRINT ABS(5)          > PRINT ABS(–5)
5                       5
```

5.5.2.1.2 **NOT**([expr])

Returns a 16 bit one's complement of the expression. The expression must be a valid integer (i.e. between 0 and 65535 (0FFFFH) inclusive). Non-integers are truncated, not rounded.

Examples:

```
> PRINT NOT(65000)      > PRINT NOT(0)
535                     65535
```

5.5.2.1.3 **INT**([expr])

Returns the integer portion of the expression.

Examples:

```
> PRINT INT(3.7)        > PRINT INT(100.876)
3                       100
```

5.5.2.1.4 **SGN**([expr])

Returns a value of +1 if the argument is greater than zero, zero if the argument is equal to zero and –1 if the argument is less than zero.

Examples:

```
> PRINT SGN(52)    > PRINT SGN(0)    > PRINT SGN(–8)
1                  0                 –1
```

5.5.2.1.5 **SQR**([expr])

Returns the square root of the argument. The argument may not be less than zero. The result returned is accurate to within +/–a value of 5 on the least significant digit.

Examples:

```
> PRINT SQR(9)     > PRINT SQR(45)     > PRINT SQR(100)
3                  6.7082035           10
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5.2 Unary Operators (continued)

5.5.2.1.6 **RND**

Returns a pseudo-random number in the range between 0 and 1 inclusive. The RND operator uses a 16-bit binary seed and generates 65536 pseudo-random numbers before repeating the sequence. The numbers generated are specifically between 0/65535 and 65535/65535 inclusive. Unlike most BASICS, the RND operator in the BASIC Module does not require an argument or a dummy argument. If an argument is placed after the RND operator, a BAD SYNTAX error occurs.

Examples:

```
> PRINT Rnd
.30278477
```

5.5.2.1.7 **PI**

PI is a stored constant. In the BASIC Module PI is stored as 3.1415926.

5.5.2.2 Log Functions

Following are the log functions.

5.5.2.2.1 **LOG**([expr])

Returns the natural logarithm of the argument. The argument must be greater than 0. This calculation is carried out to 7 significant digits.

Examples:

```
> PRINT LOG(12)              > PRINT LOG(EXP(1))
2.484906                     1
```

5.5.2.2.2 **EXP**([expr])

This function raises the number "e" (2.7182818) to the power of the argument.

Examples:

```
> PRINT EXP(1)               > PRINT EXP(LOG(2))
2.7182818                    2
```

5.5.2.3 Trig Functions

5.5.2.3.1 **SIN**([expr]

Returns the sine of the argument. The argument is expressed in radians. Calculations are carried out to 7 significant digits. The argument must be between ± 200000.

Examples:

```
> PRINT SIN(PI/4)            > PRINT SIN(0)
.7071067          0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5.2 Unary Operators (continued)

5.5.2.3.2 **COS**([expr])

Returns the cosine of the argument. The argument is expressed in radians. Calculations are carried out to 7 significant digits. The argument must be between ± 200000.

Examples:

```
> PRINT COS(PI/4)          > PRINT COS(0)
.7071067                   1
```

5.5.2.3.3 TAN([expr])

Returns the tangent of the argument. The argument is expressed in radians. The argument must be between ± 200000.

Examples:

```
> PRINT TAN(PI/4)          > PRINT TAN(0)
1                          0
```

5.5.2.3.4 ATN([expr])

Returns the arctangent of the argument. The result is in radians. Calculations are carried out to 7 significant digits. The ATN operator returns a result between –PI/2 (3.1415926/2) and PI/2.

Examples:

```
> PRINT ATN(PI)            > PRINT ATN(1)
1.2626272                  .78539804
```

5.5.2.4 Comments on Trig Functions

The SIN, COS and TAN operators use a Taylor series to calculate the function. These operators first reduce the argument to a value between 0 and PI/2. This reduction is accomplished by the following equation:

reduced argument = (user arg/PI – INT(user arg/PI)) * PI

The reduced argument, from the above equation, is between 0 and PI. The reduced argument is then tested to see if it is greater than PI/2. If it is, then it is subtracted from PI to yield the final value. If it is not, then the reduced argument is the final value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP