

# OMRON

## V+

---

### User's Manual

# Copyright Notice

---

The information contained herein is the property of Omron Robotics and Safety Technologies, Inc., and shall not be reproduced in whole or in part without prior written approval of Omron Robotics and Safety Technologies, Inc.. The information herein is subject to change without notice and should not be construed as a commitment by Omron Robotics and Safety Technologies, Inc. The documentation is periodically reviewed and revised.

Omron Robotics and Safety Technologies, Inc., assumes no responsibility for any errors or omissions in the documentation.

Copyright © 2022 by Omron Robotics and Safety Technologies, Inc. by OMRON Corporation. All rights reserved.

Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

DeviceNet is a trademark of ODVA.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Created in the United States of America.

# Table of Contents

---

<b>Chapter 1: Introduction</b>	<b>7</b>
1.1 Intended Audience	7
1.2 Related Manuals	7
1.3 System Overview	8
The V+ Control System	8
The V+ Language	8
Robot Integrated Control	8
Standard Control	9
Timing Considerations	9
1.4 V+ Keywords	12
1.5 Monitor Command Programs and V+ programs	13
1.6 V+ File Management	13
File and Folder Name Requirements	13
File Types	14
Programs and Subroutines	14
<b>Chapter 2: Keyword Usage</b>	<b>17</b>
2.1 Using Function Keywords	17
Function Keyword Syntax	17
Variable Assignments and Data Types	17
Using Functions as Keyword Parameters	17
Function Keywords Used Within Expressions	18
2.2 Using Monitor Command Keywords	18
Monitor Command Keyword Syntax	18
2.3 Using Program Command Keywords	19
Program Command Keyword Syntax	19
2.4 Using System Parameter Keywords	20
Available System Parameters	20
Viewing and Setting System Parameters	21
2.5 Using System Switch Keywords	22
Available System Switches	22
Viewing and Setting System Switches	23
<b>Chapter 3: V+ System Operations</b>	<b>25</b>
3.1 System Messages	25
Message Types	25
Message Handling	25
3.2 Digital I/O Control	26
I/O Usage Considerations	26
Basic I/O Control	26
Soft Signals	27

3.3 Disk I/O Operations .....	27
Logical Unit Numbers .....	27
I/O Operation Error Status .....	27
Attaching and Detaching Logical Units .....	28
Attaching and Detaching Disk Devices .....	28
Reading and Writing with I/O Devices .....	29
Input Wait Modes .....	30
Output Wait Modes .....	30
Disk File Operations .....	30
Advanced Disk Operations .....	33
<b>Chapter 4: Motion Control Operations .....</b>	<b>35</b>
4.1 Motion Control Overview .....	35
4.2 Joint-interpolated Motion vs. Straight-line Motion .....	35
4.3 Safe Approaches and Departures .....	35
4.4 Continuous-Path Trajectories .....	36
4.5 Breaking Continuous-Path Operation .....	37
4.6 Procedural Motion .....	38
4.7 Motion Control Timing Considerations .....	39
4.8 Robot Speed and Performance .....	39
Robot Speed .....	39
Robot End-effector / Tool Tip Speed Considerations .....	40
Robot Performance .....	40
4.9 Motion Modifiers .....	41
<b>Chapter 5: Variables and Data Types .....</b>	<b>43</b>
5.1 Variable Creation .....	43
5.2 Variable Classifications .....	43
Variable Classification Scope .....	43
External Variables .....	45
Global Variables .....	46
Local Variables .....	47
Automatic Variables .....	47
5.3 Variable Name Requirements .....	48
5.4 Variable Initialization .....	48
5.5 Variable Operators .....	49
Assignment Operators .....	49
Mathematical Operators .....	49
Relational Operators .....	50
Logical Operators .....	51
String Operators .....	51
Order of Evaluation .....	51
5.6 Numeric Representation .....	52
Numeric Expressions .....	52
5.7 String Data Types .....	53
ASCII Values .....	53

5.8 Real Integer Data Types .....	53
Real and Integer Value Ranges .....	54
5.9 Logical Constants .....	54
5.10 Location Data Types .....	54
Creating Location Variables .....	55
Modifying Location Variables .....	55
5.11 Arrays .....	57
Global Array Access Restriction .....	58
5.12 Variable Context .....	58
Interpretation of Context Specification for Variables .....	60
<b>Chapter 6: Monitor Command Programs .....</b>	<b>61</b>
6.1 Monitor Command Program Overview .....	61
6.2 Monitor Command Program Names .....	61
6.3 Monitor Command Program Contents .....	61
6.4 Executing Monitor Command Programs .....	62
Executing Monitor Command Programs from the Monitor Window .....	62
Autostarting a Monitor Command Program .....	62
Monitor Command Program Processing .....	63
<b>Chapter 7: V+ programs .....</b>	<b>65</b>
7.1 V+ program Format .....	65
Program Lines .....	65
Program Structure .....	66
7.2 V+ Program Execution .....	66
Program Tasks .....	67
V+ program Processing .....	68
7.3 Program Control .....	69
Unconditional Branching .....	69
Conditional Branching .....	70
Looping Structures .....	71
Loading Programs and Variables to System Memory .....	73
Removing Programs and Variables from System Memory .....	74
7.4 V+ Program Interrupts .....	74
WAIT Program Interrupts .....	75
WAIT.EVENT Program Interrupts .....	75
REACT and REACTI Program Interrupts .....	75
REACTE Program Interrupts .....	76
HALT, STOP, and PAUSE Program Interrupts .....	77
Additional Program Interrupt Keywords .....	77
Program Interrupt Example .....	77
7.5 V+ program Stacks .....	79
V+ Program Stack Requirements .....	79
7.6 Exchanging Information Between V+ programs .....	80
Exchanging Information Using Global Variables .....	80

Exchanging Information Using Soft-signals .....	80
Exchanging Information Using the Program Argument List .....	81
7.7 Reentrant and Recursive Programs .....	83
Reentrant Programs .....	83
Recursive Programs .....	84
7.8 Asynchronous Processing .....	84
Error Trapping .....	85
<b>Chapter 8: T20 Pendant Programming .....</b>	<b>87</b>
8.1 Attaching and Detaching the T20 Pendant .....	87
8.2 Writing to the T20 Pendant Display .....	87
8.3 Detecting User Input .....	88
Detecting Pendant Key Presses .....	88
8.4 Other Pendant Programming Functions .....	89
<b>Appendix .....</b>	<b>91</b>
A.1 Warning, Information, and Error Messages .....	91
System Warning Messages .....	91
System Information Messages .....	93
System Error Messages .....	94

# Chapter 1: Introduction

This manual contains information that is necessary to use V+. Please read this manual and make sure you understand the functionality of V+ before attempting to use it.

The information in this document applies to systems running versions 4 and higher. Systems prior to version 4 are considered as legacy systems.

## 1.1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of common programming practices and robotic control methods.

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of installing and maintaining FA systems.
- Personnel in charge of managing FA systems and facilities.

## 1.2 Related Manuals

Use the following related manuals for reference.

Table 1-1. Related Manuals

Manual	Description
V+ Keyword Reference Manual (Cat. No. I672)	Provides references to V+ Keyword use and functionality.
Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)	Contains information that is necessary to use Automation Control Environment (ACE) software.
Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595)	Learning about the operating procedures and functions of the Sysmac Studio to configure Robot Integrated System using Robot Integrated CPU Unit.
Robot User Manuals	User Manual for the robot in use.
T20 Pendant User's Manual (Cat. No. I601)	Describes the use of the optional T20 manual control pendant.
NJ-series Robot Integrated CPU Unit User's Manual (Cat. No. 0037)	Describes the settings and operation of the CPU Unit and programming concepts for OMRON robot control.
Robot Safety Guide (Cat. No. I590)	Contains safety information for OMRON industrial robots.

## 1.3 System Overview

V+ is an interpreted programming language and control system for OMRON industrial robots. This system offers a variety of keywords supporting robot motion, I/O, vision, file operations, and communication with other devices. Real-time interpretation and forward processing of programs in parallel tasks provides continuous path trajectory generation, as well as on-line program generation, debugging, and modification.

### The V+ Control System

The V+ control system is automatically launched when the robot controller and NJ-series Robot Integrated CPU Unit (if present) are powered ON. The control system accepts instructions from application programs, input from workcell peripheral devices, and operator input from the pendant. The tasks performed by the control system include the following.

- Managing the execution of application programs.
- Managing the flow of information to and from storage devices.
- Monitoring external devices attached to the controller.
- Reporting errors generated during processing.

### The V+ Language

The V+ language is comprised of V+ keywords that provide control, data manipulation, and other application-specific functionality. These keywords are used in statements with specific syntax detailed in this document with examples. Statements are executed in a consecutive manner until the .END statement is reached. Every statement between the .PROGRAM *some\_name*() line and the .END line are executed as a program.. Keywords must be used to create V+ programs or issue individual commands in the Monitor Window. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for other syntax and statement examples.

**Additional Information:** Refer to V+ Program Execution on page 66

### Robot Integrated Control

When the robot is used with the NJ-series Robot Integrated CPU Unit, I/O, logic, safety, and motion control functions can be created using IEC 61131-3 programming language specifications. With the addition of shared variables, shared signals, and function blocks for interacting with V+ program execution, a user can develop an entire control solution from the Sysmac Studio software.

The V+ control system runs on OMRON robots and the Robot Integrated CPU Unit in a distributed control configuration, while communicating with other system objects to facilitate programming and runtime functionality. The general system overview is shown below.

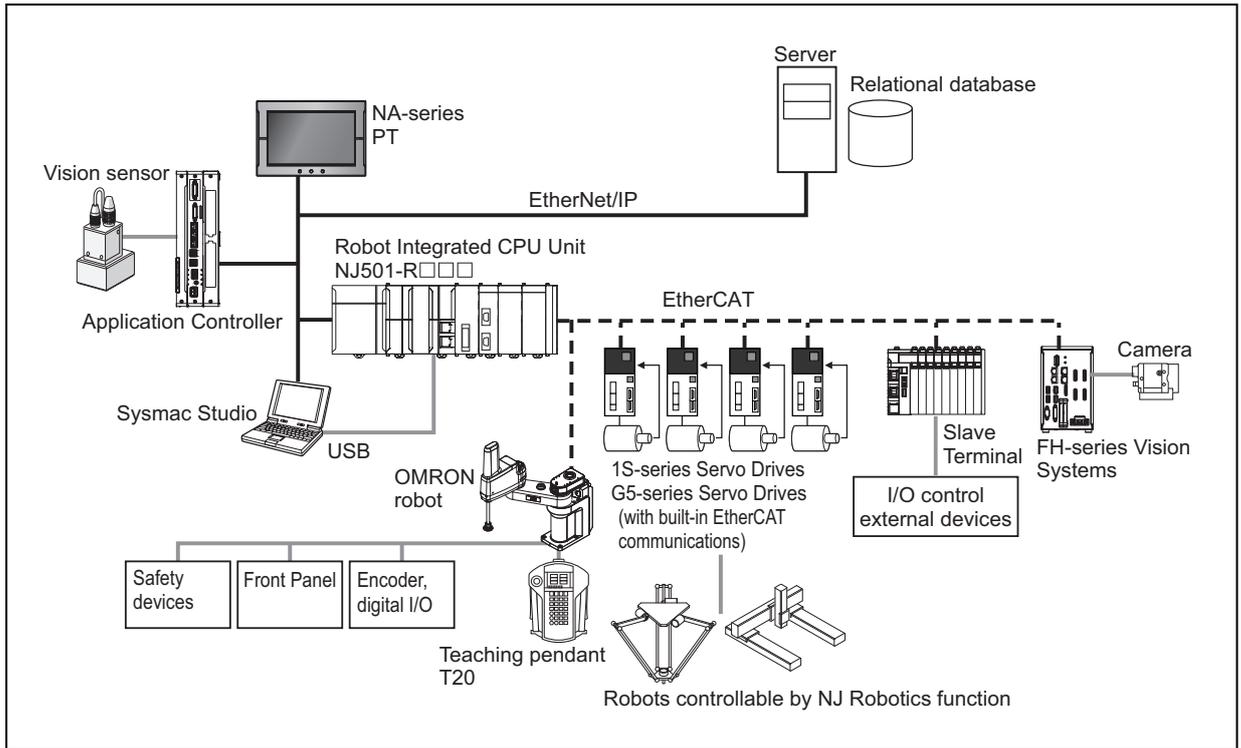


Figure 1-1. Robot Integrated Control System Overview

## Standard Control

When the robot is used with the Standard Control method, I/O, logic, safety, and motion control functions can be created using ACE software. Standard Control uses a centralized control configuration where V+ keyword execution occurs within the robot controller.

The general system overview is shown below.

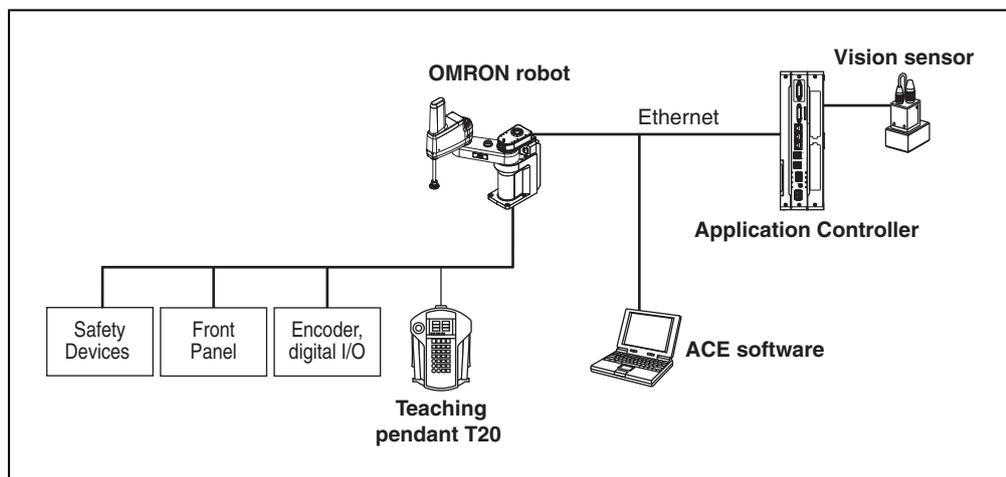


Figure 1-2. Standard Control System Overview

## Timing Considerations

Execution and response timing for some keyword execution needs to be considered for demanding applications where high-speed motion coordination is required. This timing may

vary between Robot Integrated Control and Standard Control systems as described below.

**Robot Integrated Control Timing**

Robot Integrated Control systems have a distributed control configuration. As a result, the execution times of certain keywords take longer than execution times in Standard Control systems.

The keywords described in this section are executed by the system services, robot control processing service, and robot controller. EtherCAT communications between the Robot Integrated CPU Unit and the robot controller occur during these operations.

Keywords described below execute in approximately 8 ms or 16 ms, depending on the Host system setting. Other keywords will execute within approximately 2 ms or 4 ms, depending on the Host system setting. Refer to the *NJ-series Robot Integrated CPU Unit User's Manual (Cat. No. 0037)* for more information about settings, tasks, and services for Robot Integrated CPU Unit.

Use the following diagram to understand the basic operations that occur during these keyword executions.

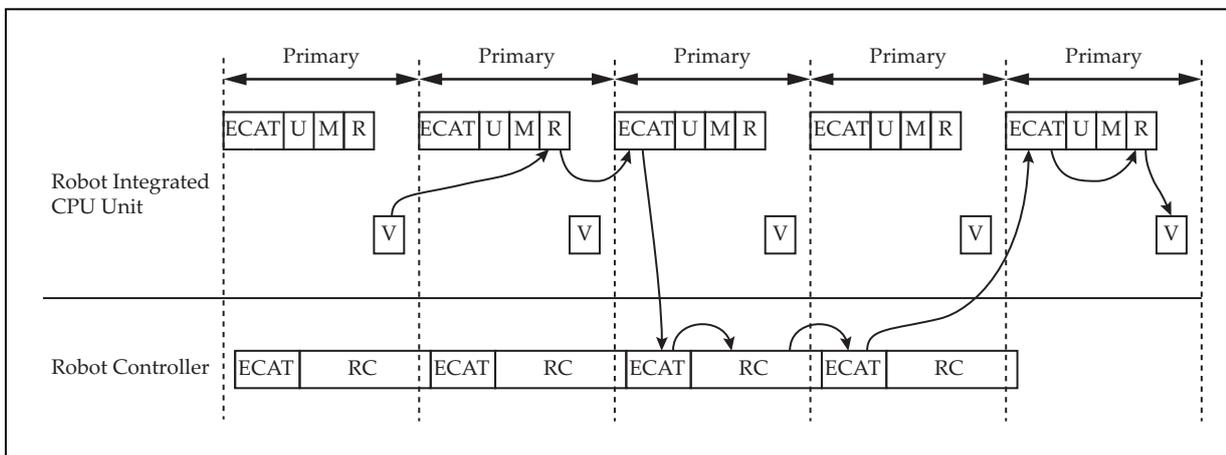


Figure 1-3. Distributed Control Timing

Item	Description
ECAT	EtherCAT Communications
U	User Program Execution
M	Motion Control
R	Robot Control
V	V+ Program Execution (Robot Integrated CPU Unit System Services)
RC	Robot Control

Keywords that have longer execution times are provided below.

Category	Examples	Description
Motion Initiation	MOVE, MOVES, MOVEC, READY, APPRO, DEPART, ALIGN, DRIVE, JMOVE	Keywords that create robot motion, excluding jogging.
Motion Termination	BREAK, BRAKE, ESTOP	Keywords that stop ongoing

Category	Examples	Description
		motion.
Kinematics Calculations	SOLVE.TRANS, SOLVE.ANGLES, SOLVE.FLAGS, TOOL, BASE, INRANGE	Keywords that output a transformation or location based on input parameters.
Robot/Servo Parameter Access	DEVICES (type 3, 5 or 6), PAYLOAD, HOUR.METER	Keywords that read or write robot or servo parameters.
Belt Parameter Access	DEFBELT, SETBELT, BELT, BELT.MODE, DEVICE (type 0)	Keywords that define, executes, or configure a belt.
Latch	LATCH, LATCHED, DEVICE (type 0), ENCLATCH	Keywords that read or write latched position values.
Permanent Motion Parameter Access	FINE, COARSE, NULL, SPEED, ACCEL	Keywords that determine how robot motions are executed or nulled.
Robot Attach/Detach	ATTACH and DETACH	Keywords that implicitly attach or detach the robot to a specific task.
Robot Power	ENABLE POWER, DISABLE POWER	Keywords that affect the robot power, excluding interrupted motion.
Calibration	CALIBRATE	Keywords that execute runtime calibration for joint offsets.
Jogging	JOG	Keywords associated with robot jogging in auto or manual mode.
Teach Pendant	PENDANT, PDNT.CLEAR, PDNT.NOTIFY, KEYMODE	Keywords that are associated with reading from and writing to the pendant.

### Standard Control Timing

Standard Control systems do not require communications between devices and all keywords can be executed within one 4 ms cycle of the robot controller.

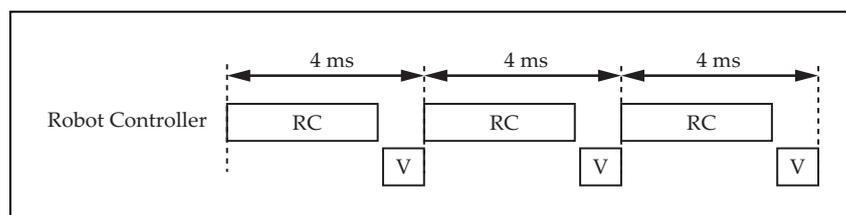


Figure 1-4. Standard Control Timing

Item	Description
RC	Robot Control
V	V+ Program Execution

## 1.4 V+ Keywords

V+ keywords are a set of instructions that are used to perform various operations for a robot-based application. Keywords are used to create statements arranged in steps for the following programmatic functionality.

- Robot motion control
- Error processing and handling
- Arithmetic functions
- System parameter and switch manipulation
- Data management
- I/O control and external device communication
- Logic and other conditional evaluations
- Subroutine execution and control
- Executing various system functions
- T20 Pendant functions

Depending on the type of keyword, they can be used to build a program or they can be issued individually using the Monitor Window in the Sysmac Studio or ACE software.

Keywords are categorized into six types as described below.

Table 1-2. V+ Keyword Types

Keyword Type	Usage
Function keywords	Used to return values from the V+ control system. Refer to Using Function Keywords on page 17 for more information.
Monitor command keywords	Used to issue individual operations in the Monitor Window or to create Monitor Command Programs. Refer to Using Monitor Command Keywords on page 18 for more information.
Other keywords	Used to specify units when using the SPEED program command keyword. Refer to the <i>V+ Keyword Reference Manual (Cat. No. I672)</i> for more information.
Program command keywords	Used to command operations in V+ programs. Refer to Using Program Command Keywords on page 19 for more information.
System parameter keywords	Used to manipulate system parameters in V+ programs or with the Monitor Window. Refer to Using System Parameter Keywords for more information.
System switch keywords	Used to manipulate system switches in V+ programs or with the Monitor Window. Refer to Using System Switch Keywords on page 22 for more information.

## Keywords and Syntax

In order for keywords to work with the V+ control system, they must be issued with a specific syntax. This syntax generally consists of a keyword and associated parameters in a specific order, separated by characters such as commas, brackets, and parentheses. This syntax is detailed in this manual and also in the *V+ Keyword Reference Manual (Cat. No. I672)*. An example of keyword syntax is shown below.

```
SET loc_name = SHIFT(loc_value BY 5, 5, 5)
```

When creating V+ programs with the Sysmac Studio or ACE software, syntax is checked and formatted as the characters are input. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)* for more information.

## 1.5 Monitor Command Programs and V+ programs

There are two types of programs that the V+ control system can execute: Monitor Command Programs and V+ programs.

V+ programs contain the logic, motion control, and vision keywords that control a robot during run-time. Refer to V+ programs on page 65 for more information.

Monitor Command Programs are used to perform system-level functions such as loading files, changing the default directory path, and executing main V+ programs after system boot up. Refer to Monitor Command Programs on page 61 for more information.

## 1.6 V+ File Management

The V+ control system has a file management system very similar to other operating systems. Each file within a subdirectory must have a unique name. There are several file extensions that are used for different types of files in the V+ system as described in File Types on page 14.

New files can be created with the Sysmac Studio or ACE software, or by issuing specific keywords. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)* and the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

### File and Folder Name Requirements

The V+ file and folder name requirements are described below.

- Names can have a maximum of eight characters.
- File names must have an extension (file type) designation that is 1 to 3 characters in length. Refer to File Types on page 14 for more information.
- Names can only include alphanumeric characters and the underscore ( `_` ) character. No other special characters are permitted.
- Names cannot contain blank spaces.
- Keywords cannot be used as file names. Do not name files the same as any keywords.
- Names are case-sensitive and follow POSIX rules.

**NOTE:** File names are not case-sensitive.

## File Types

There are several file types that are used by the V+ control system. File types that are used during system development, commissioning, debugging, and other typical operations are listed below.

**IMPORTANT:** Direct access and editing of these files is not necessary under normal use.

Table 1-3. V+ File Type Descriptions

File Type	Description
.v2	<p>V+ programs</p> <p>Global variables referenced by the programs and subroutines can be stored in the file.</p> <p>All of the subroutines referenced (directly or indirectly) by the specified program can be stored in the file.</p> <p>Refer to the <i>V+ Keyword Reference Manual (Cat. No. I672)</i> for information about storing programs and variables in a disk file using the STORE keyword.</p> <p>Refer to the <i>Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)</i> for information about saving programs and variables to the controller.</p>
.cal	System calibration data files.
.xml	Robot specification and other parameter files.
.rtf	Text files (Readme).
.pg	<p>V+ program or group of programs (module)</p> <p>In addition to the programs specified to store in the file, any subroutines referenced by those programs are also stored in the file.</p> <p>Refer to the <i>V+ Keyword Reference Manual (Cat. No. I672)</i> for information about storing programs and variables in a disk file using the STOREP and STOREM keywords.</p>

## Programs and Subroutines

V+ is an interpreted language, therefore linking and compiling are not required. Main programs and subroutines always exist as separate programs. The V+ file structure allows you to keep a main program and all the subroutines it calls or executes together in a single file so that when a main program is loaded, all the subroutines it calls are also loaded. A single file that contains a program and subroutines is also referred to as a module.

If a program calls a subroutine that is not resident in system memory, the error \*Undefined program or variable name\* will result.

**Additional Information:** Refer to the STORE\_ and MODULE keyword descriptions in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)* for more information about creating a program file.



# Chapter 2: Keyword Usage

This chapter describes how to use the different types of keywords with the V+ system.

## 2.1 Using Function Keywords

Function keywords are issued within V+ Program statements and can be used in several different ways as described below.

**Additional Information:** Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about specific function keywords.

### Function Keyword Syntax

V+ provides you with a wide variety of predefined function keywords for performing string, mathematical, and other data manipulation. In most cases, you must provide the data that is input to a function keyword. The keyword then returns a value based on a specific operation on that data. Function keywords can be used anywhere that a value or expression would be used.

Correct syntax for function keywords must be observed when creating statements in V+ programs. If incorrect syntax is used, V+ will return an *\*Illegal monitor command\** error.

### Variable Assignments and Data Types

When a function keyword is issued, a value is returned by the V+ control system. The data type of the value that is returned must match the data type of the variable that is assigned to the function. For example, if the \$TIME function keyword is issued, V+ will return a string data type value. The examples below demonstrate various data types used with variable assignments.

```
$current_time = $MID($TIME(,time),11,8)
SET #pos1 = #PHERE
SETBELT %main.belt = BELT(%main.belt)
```

**Additional Information:** Refer to Variables and Data Types on page 43 for more information

### Using Functions as Keyword Parameters

A function keyword can be used as a parameter as long as the data type returned by that function is the correct type. For example, consider the SQR function statement shown below. The SQR function keyword uses the syntax "SQR(value)" for reference.

```
i = SQR(SQR(x))
```

The statement above returns the absolute value of variable "x" and assigns it to variable "i". Any function that returns a numerical value can be substituted for "SQR(x)" in the statement above.

## Function Keywords Used Within Expressions

A function keyword can be used as an expression as long as the data type returned by that function is the correct type. For example, consider the IF ... THEN conditional statement shown below. The IF ... THEN keyword uses the syntax "IF logical\_exp THEN" for reference.

```
IF LEN($string_variable) > 12 THEN
```

The statement above evaluates the expression "LEN(\$string\_variable) > 12". This expression includes the LEN function keyword that returns the number of characters in the variable "\$string\_variable". Any function that returns a numerical value can be substituted for LEN (\$string\_variable) in the statement above.

## 2.2 Using Monitor Command Keywords

Monitor command keywords can be issued in the Monitor Window or with the use of a Command Program. Most monitor command keywords include at least one parameter that controls how the V+ system executes the operation. This additional information is specified when the keyword is issued in the Monitor Window. Parameters must be entered in the order they are listed and they must use correct syntax.

**Additional Information:** Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about specific monitor command keywords.

Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)* for more information about the Monitor Window.

## Monitor Command Keyword Syntax

Correct syntax for monitor command keywords must be observed when creating statements in V+ programs. If incorrect syntax is used, V+ will return an \*Illegal monitor command\* error.

In general, monitor command keywords have the following formats. The keyword is shown in uppercase and the arguments are shown in lowercase. Required keywords, parameters, and symbols such as equal signs and parentheses are shown in bold text. Optional keywords, parameters, and symbols are shown in regular text.

```
KEYWORD
```

```
KEYWORD req_param
```

```
KEYWORD (req_param)
```

```
KEYWORD opt_param
```

```
KEYWORD opt_param1, opt_param2, ... , opt_paramX
```

```
KEYWORD req_param, opt_param = value
```

```
KEYWORD @task:program_step(expression)
```

```
KEYWORD req_param1 = req_param2
```

```
KEYWORD req_param1 = req_param2 KEYWORD opt_param
```

**Additional Information:** All required or optional monitor command keyword parameters and associated syntax is described for each keyword in the

*V+ Keyword Reference Manual (Cat. No. I672)*. Refer to specific keyword details in that manual for more information.

### Monitor Command Keyword Parameter Entry

Monitor command keyword parameters can be optional or required. If a parameter is required, a value must be entered on the command line or the command will not execute correctly. A space is required between the keyword and the parameters that follow it. A comma is typically used to separate parameters but some monitor command keywords may require an equal sign (=) assignment operator.

If a parameter is optional, its value can be omitted and the system will substitute a default value. For example, the STATUS monitor command keyword can be issued as shown below. The STATUS syntax is "STATUS select" for reference, where "select" is an optional parameter.

```
STATUS
```

Issuing the STATUS monitor command keyword as shown above will return status information for all the program tasks.

Issuing the STATUS monitor command keyword with a parameter value as shown below will return status information for only system task number 1.

```
STATUS 1
```

Spaces before and after parameter separators are optional. If one or more parameters follow an omitted parameter, the parameter separator(s) must be typed. If all the parameters following an omitted parameter are optional, and those parameters are also omitted, the separators do not need to be typed.

## 2.3 Using Program Command Keywords

Program command keywords are used to create V+ programs for robot control and other functions such as I/O control, file operations, error handling, and data management. This section provides details about the usage of program command keywords when creating V+ programs.

### Program Command Keyword Syntax

Correct syntax for program command keywords must be observed when creating statements in V+ programs. If incorrect syntax is used, the statement will cause errors in the V+ Editor and the program cannot be executed.

Symbols such as commas, brackets, and parentheses may be required for program command keyword syntax when parameters are present. program command keywords may contain parameters that define or specify information needed for the operation. Some program command keywords do not require any parameters while other program command keywords have required or optional parameters.

In general, program command keywords have the following formats (not an exhaustive listing). The keyword is shown in uppercase and the arguments are shown in lowercase. Required keywords, parameters, and symbols such as equal signs and parentheses are shown in **bold** text. Optional keywords, parameters, and symbols are shown in regular text.

```
KEYWORD
```

```
KEYWORD req_param
```

```
KEYWORD opt_param
```

```
KEYWORD (parameter)opt_param
```

```
KEYWORD opt_param1, opt_param2, ... , opt_paramX
```

```
KEYWORD req_param, opt_param = value
KEYWORD req_param1 = req_param2
KEYWORD req_param1 = req_param2 KEYWORD opt_param
```

**Additional Information:** All required or optional program command keyword parameters and associated syntax is described for each keyword in the *V+ Keyword Reference Manual (Cat. No. I672)*. Refer to a specific keyword details in that manual for more information.

### No Required Parameters

Some program command keywords do not require any parameters such as LEFTY, RIGHTY, ABOVE, and BELOW. Keywords that do not require parameters can be used as shown below.

```
ABOVE
MOVE point1
BREAK
```

### Required or Optional Parameters

Some program command keywords have parameters that are required or optional. When using a keyword that uses multiple parameters, symbols such as commas, brackets, and parentheses may be required. If required parameters or syntax is incorrect, an error will occur. Refer to Warning, Information, and Error Messages on page 91 for error information and the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about the use of symbols to complete the correct syntax for V+.

**NOTE:** When program lines are entered, extra blank spaces can be used between any elements in the line. The V+ editor adds or deletes spaces in program lines to make them conform with the standard spacing. The editors also automatically format the lines to uppercase for all keywords and lowercase for all user-defined names.

## 2.4 Using System Parameter Keywords

System parameters determine certain operating characteristics of the V+ system. These parameters have numeric values that can be changed from the Monitor Window or from within a program to suit particular system configurations and needs. The various parameters are described in this section along with the operations for displaying and changing their values.

### Available System Parameters

Use the following table to understand the available system parameters and their basic functions. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for setting details, examples, and other information.

Table 2-1. System Parameter Details

Parameter	Description
BELT.MODE	Set characteristics of the conveyor tracking feature of the V+ system.
DEVIATION	Adds a path deviation from 1 to 100% to the motion in the singularity region when a robot is in singularity.
NOT.CALIBRATED	Represents the calibration status of the robot(s) controlled by the V+

Parameter	Description
	system.
VTIMEOUT	Sets a timeout value so that an error message is returned if no response is received following a vision command.
JOG.TIME	Sets the keep-alive time of a jog operation. Each time a jog operation is executed, this parameter setting specifies the time the axis or joint moves.

## Viewing and Setting System Parameters

System parameters can be viewed and set in the Monitor Window. They can also be controlled by V+ programs. Use this section to understand how to view and set system parameters.

### Viewing and Setting System Parameters with the Monitor Window

The PARAMETER monitor command keyword is used to view and set parameter values as shown in the examples below.

#### Viewing Parameter Values

The following example will display all parameters and their current values in the Monitor Window.

```
PARAMETER
```

The following example will display the BELT.MODE parameter current value in the Monitor Window.

```
PARAMETER BELT.MODE
```

#### Setting Parameter Values

The following example will set the BELT.MODE parameter to 4.

```
PARAMETER BELT.MODE = 4
```

### Reading and Writing System Parameters with V+ Programs

Parameters can be set during V+ program execution by using the PARAMETER program command keyword.

**NOTE:** It is common practice to use a Monitor Command Program to set parameters. Refer to Monitor Command Programs on page 61 for more information.

The following program statement will set the BELT.MODE system parameter to have bits 1 and 3 set to 1 (mask values 1 + 4) using the PARAMETER program command keyword.

```
PARAMETER BELT.MODE = 5
```

A parameter value can be returned during V+ program execution or using the Monitor Window.

The following program statement will return the current setting of the BELT.MODE system parameter in the Monitor Window.

TYPE "The BELT.MODE parameter is set to", PARAMETER(BELT.MODE)

## 2.5 Using System Switch Keywords

System switches determine certain operating characteristics of the V+ system. These switches can be turned ON or OFF from the Monitor Window or from within a program.. The various system switches are described in this section along with the operations for displaying and controlling them.

### Available System Switches

Use the following table to understand the available system switches and their basic functions. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for settings, details, examples, and other information.

Switch	Description
AUTO.POWER.OFF	<p>When this switch is ON, errors will disable high power. When this switch is OFF, these errors stop the robot and signal the V+ system, but do not cause high power to be turned OFF.</p> <p>The errors are defined as follows.</p> <ul style="list-style-type: none"> <li>• (-624) *force protect limit exceeded*</li> <li>• (-1003) *Time-out nulling errors* Mtr</li> <li>• (-1006) *Soft envelope error* Mtr</li> </ul>
CP	Enables or disables continuous-path motion processing. Refer to Continuous-Path Trajectories on page 36 for more information.
DECEL.100	Enables or disables the use of 100 percent as the maximum deceleration for the ACCEL program command keyword.
DELAY.IN.TOL	Controls the timing of coarse or fine nulling after V+ completes a motion segment (positioning tolerance). Refer to COARSE and FINE program command keyword details.
DRY.RUN	<p>This switch enables or disables the transmission of motion commands to the robot. Turn this switch ON to test programs for proper logical flow and external communication functionality to prevent collisions.</p> <p><b>NOTE:</b> The T20 pendant can still be used to move the robot when DRY.RUN is enabled if there is no task attached to the robot. Otherwise, a Robot Interlocked error (-621) will occur.</p>
MESSAGES	<p>Setting this switch to ON will allow messages to be displayed on the Monitor Window while using the TYPE program command keyword.</p> <p>Setting this switch OFF will prevent messages from being displayed on the Monitor Window with the TYPE program command keyword.</p>
OBSTACLE	Enables or disables up to 4 different obstacles that are defined for a robot.
POWER	<p>This switch controls and displays the robot high power. This switch is automatically enabled whenever robot high power is turned ON.</p> <p>If the robot timing specifications in the system configuration is non-zero, enabling high power is a two-step process. In this case, after</p>

Switch	Description
	<p>enabling high power from the T20 Pendant, Monitor Window, or software, the system flashes the high power indicator for the duration of the timing specification setting (in seconds). If a timeout occurs, the message *HIGH POWER button not pressed* will appear.</p> <p>Refer to Warning, Information, and Error Messages on page 91 for more information.</p> <p>Refer to the <i>Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No I633)</i> for more information about robot timing specification settings.</p>
ROBOT	Enables or disables one robot or all robots.
SCALE.ACCEL	Enables or disables the scaling of acceleration and deceleration as a function of program speed as long as the program speed is below a preset threshold.
SCALE.ACCEL.ROT	Specifies whether or not the SCALE.ACCEL system switch accounts for the cartesian rotational speed during straight-line motions.
UPPER	Determines whether comparisons of string values will consider lowercase letters the same as uppercase letters. When this switch is ON, all lowercase letters are considered as though they are uppercase.

## Viewing and Setting System Switches

System switches can be viewed and set in the Monitor Window. They can also be controlled by V+ programs. Use this section to understand how to view and set system switches.

### Viewing and Setting System Switches from the Monitor Window

The SWITCH monitor command keyword is used to view and set switches as shown in the examples below.

#### Viewing System Switches

The following example will display all switches and their current values in the Monitor Window.

```
SWITCH
```

The following example will display the POWER switch current state in the Monitor Window.

```
SWITCH POWER
```

#### Setting System Switches

The following example will turn ON the AUTO.POWER.OFF switch.

```
ENABLE AUTO.POWER.OFF
```

The following example will turn OFF the AUTO.POWER.OFF switch.

```
DISABLE AUTO.POWER.OFF
```



# Chapter 3: V+ System Operations

---

The following sections describe various V+ system operations.

## 3.1 System Messages

System messages have designated numbers and strings for identification. These can be used to detect and recover from various V+ conditions with V+ programs or using the Monitor Window.

**Additional Information:** Refer to Warning, Information, and Error Messages on page 91

### Message Types

There are three different message types:

- Informational messages (message numbers 0 to 49)
- Warning messages (message numbers 50 to 299)
- Error messages (messages with negative values)

### System Behavior

The V+ system will behave differently depending on the type of message and V+ state when the message occurs.

Informational and warning messages do not impact the system operation and can be used for observational purposes.

Errors messages have an impact on system operation. They will not completely stop the V+ system, but individual tasks associated with the error will stop running.

System behavior after errors occur depend on the running task, robot status, and type of error. If the robot is running and on the same task where an error occurs, the robot will be stopped with maximum deceleration and an error message(s) will be generated. If an error occurs on a task that is not running a robot, only this task will be stopped and the robot will not be affected.

### Message Handling

V+ will internally store system message information in order to be accessed with the ERROR and REACTE keyword operations.

The following information is stored for access using the ERROR keyword operation.

- The last system message of a task (current or selected)
- The last system message of each attached robot
- The last message of the V+ system
- A category, source, and name for each system message.

**Additional Information:** Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about the ERROR and REACTE keyword operations.

## 3.2 Digital I/O Control

When V+ starts, blocks of system memory are assigned to internal and external digital I/O detected by the system.

Several program command keywords are available for digital I/O control purposes. Digital input signals can be monitored for changes and outputs can be turned ON and OFF conditionally.

Refer to the robot user guide for more information about default signal allocation.

### I/O Usage Considerations

The RESET and DEF.DIO program command keywords have no effect on Host I/O signal numbers 4001 to 4999.

The IO monitor command keyword will return the value of signals that are mapped in the host and returns "-" for signals that are not mapped.

The SIG.INS function keyword will return TRUE if the host signal is available and FALSE if it is not available (not mapped).

Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

### Basic I/O Control

Use the following examples to understand basic I/O control functionality with program command keywords.

#### Input Signal Examples

The following example halts program execution until a switching device attached to digital input channel 1001 is closed.

```
WAIT SIG(1001)
```

The following example evaluates input signal 1002 and calls a program "service.feeder" when the signal turns ON.

```
IF SIG(1002) THEN
    CALL service.feeder()
END
```

**NOTE:** The example above will call "service.feeder" if signal 1002 is ON only when the condition is evaluated.

#### Output Signal Examples

The following example turns OFF digital output signal 33.

```
SIGNAL -33
```

The following example turns ON digital output signal 33.

```
SIGNAL 33
```

The following example controls multiple digital output signals with a single statement.

```
SIGNAL 33, -34, 35
```

## Soft Signals

Soft signals are provided in the range of 2001 to 2999. These signals are accessible with all V+ programs and can be used with the SIG and SIGNAL keywords for interaction between V+ programs on different tasks or interaction between V+ programs and functionality in an Application Manager..

## 3.3 Disk I/O Operations

The following sections describe the basic procedures and functions for disk I/O operations.

### Logical Unit Numbers

All V+ disk I/O operations reference an integer value called a Logical Unit Number (LUN). The LUN provides a method of identifying which device or file is being referenced by an I/O operation. A LUN device can refer to a robot, a disk, a TCP protocol device driver, a TFTP server, or a UDP protocol device driver.

The LUN specifier is a parameter used in several keywords listed below. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

The following program command keywords use the LUN parameter.

- ATTACH
- DETACH
- FCMND
- FOPEN
- FSET
- READ
- WRITE

### I/O Operation Error Status

Unlike most other V+ keywords, I/O operations are expected to fail under certain circumstances. For example, when reading a file, an error status is returned to the program to indicate when the end of the file is reached. The program is expected to manage this error and continue execution.

For these reasons, V+ I/O keywords normally do not stop program execution when an error occurs. Instead, the success or failure of the operation is saved internally for access by the IOSTAT real-valued function. For example, a reference to IOSTAT(5) returns a value indicating the status of the last I/O operation performed on LUN 5. The values returned by IOSTAT fall into one of following three categories.

Table 3-1. IOSTAT Keyword Return Values

Value	Description
1	The I/O operation completed successfully.
0	The I/O operation has not yet completed. This value appears only if a pre-read or no-wait I/O operation is being performed.
<0	The I/O operation completed with an error. The error code indicates what type of error occurred.

**Additional Information:** Refer to Warning, Information, and Error Messages on page 91 The \$ERROR string function keyword can be used in a program (or

with the LISTS monitor command keyword) to generate the text associated with most I/O errors.

**NOTE:** It is not necessary to use IOSTAT after use of a GETC keyword since errors are returned directly by the GETC keyword.

#### **Attaching and Detaching Logical Units**

An I/O device must be attached using the ATTACH program command keyword before it can be accessed by a program. Once a specific device (such as the teach pendant) is attached by one program task, it cannot be used by another program task.

Most I/O requests fail if the device associated with the referenced Logical Unit Number is not attached. When a program is finished with a device, it detaches the device with the DETACH program command keyword. This allows other programs to process any pending I/O operations.

A physical device type can be specified when the logical unit is attached. If a device type is specified, it supersedes the default, but only for the logical unit attached. The specified device type remains selected until the logical unit is detached.

When a control program completes execution normally, all I/O devices attached by it are automatically detached. If a program stops abnormally, most device attachments are preserved. If the control program task is resumed and attempts to reattach these logical units, it may fail because of the attachments still in effect.

**Additional Information:** The KILL monitor command forces a program to detach all the devices that it has attached.

If attached by a program, the Monitor Window and teach pendant are detached whenever the program halts or pauses for any reason, including error conditions and single-step mode. If the program is resumed, the Monitor Window and the teach pendant are automatically reattached if they were attached before the termination.

#### **Attach Request Response Mode**

An attach request can optionally specify immediate mode. Normally, an attach request is queued and the calling program is suspended if another control program task is attached to the device. When the device is detached, the next attachment in the queue will be processed. In immediate mode, the ATTACH program command keyword completes immediately with an error if the requested device is already attached by another control program task.

Attach requests can also specify no-wait mode. This mode allows an attach request to be queued without forcing the program to wait for it to complete. The IOSTAT function must then be used to determine when the attach has completed.

If a task is already attached to a logical unit, it will get an error immediately if it attempts to attach again without detaching, regardless of the type of wait mode specified.

#### **Attaching and Detaching Disk Devices**

Use the sections below to understand how to attach and detach disk devices.

##### **Attaching Disk Devices**

The type of device to be accessed is determined by the DEFAULT monitor command keyword or the ATTACH program command keyword. If the default device type set by the DEFAULT

keyword is not appropriate at a particular time, the ATTACH keyword can be used to override the default.

The following example attaches to an available disk logical unit and returns the number of the logical unit in the variable "dlun", which can then be used in other disk I/O operations.

```
ATTACH (dlun, 4) "DISK"
```

If the device name is omitted from the keyword syntax, the default device for the specified LUN is used. It is recommended that you always specify a device name with the ATTACH keyword. The device SYSTEM refers to the device specified with the DEFAULT monitor command. Once the attachment is made, the device cannot be changed until the logical unit is detached. However, any of the units available on the device can be specified when opening a file. For example, the V+ DISK units are A, C and D. After attaching a DISK device LUN, a program can open and close files on any of these disk units before detaching the LUN.

### Detaching Disk Devices

When a disk logical unit is detached, any disk file that was open on that unit is automatically closed. However, error conditions detected by the close operation may not be reported. Therefore, it is good practice to use the FCLOSE keyword to close files and to check the error status afterwards. FCLOSE ensures that all buffered data for the file is written to the disk, and updates the disk directory to reflect any changes made to the file. The DETACH keyword frees up the logical unit.

The following example will close a file and detach a disk LUN.

```
FCLOSE (dlun)
  IF IOSTAT(dlun) THEN
    TYPE $ERROR(IOSTAT(dlun))
  END
DETACH (dlun)
```

### Reading and Writing with I/O Devices

The READ program command keyword processes input from all devices. The basic READ keyword issues a request to the device attached on the indicated LUN and waits until a complete data record is received before program execution continues. The length of the last record read can be obtained with the IOSTAT function with its second argument set to a value of 2.

The GETC real-valued function returns the next data byte from an I/O device without waiting for a complete data record. It is commonly used to read data from the Monitor Window. It also can be used to read disk files in a byte-by-byte manner. Special mode bits to allow reading with no echo are supported for Monitor Window read operations.

Monitor Window input can be performed using the PROMPT program command keyword. The GET.EVENT real-valued function can be used to read input from the Monitor Window. This may be useful in writing programs that operate on both graphics and non-graphics-based systems. To read data from a disk device, a file must be open on the corresponding logical unit. The FOPEN\_ program command keywords open disk files.

The WRITE program command keyword processes output to disk devices and to the Monitor Window. The basic WRITE keyword issues a request to the device attached on the indicated LUN, and waits until the complete data record is output before program execution continues. WRITE keywords accept format control specifiers that determine how output data is formatted, and whether or not an end of record mark should be written at the end of the record.

Monitor Window output can be performed using the PROMPT or TYPE keywords. A file must be opened using the FOPENW or FOPENA keywords before data can be written to a

disk device. FOPENW opens a new file. FOPENA opens an existing file and appends data to that file.

**Additional Information:** Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for usage considerations, details, and examples of the keywords described above.

#### Input Wait Modes

Normally, V+ waits until the data from an input keyword is available before continuing with program execution. However, the READ keyword and GETC keyword accept an optional parameter that specifies no-wait mode.

In no-wait mode, these keywords return immediately with the error status -526 (No data received) if there is no data available. A program can loop and use these operations repeatedly until a successful read is completed or until some other error is received. The disk devices do not recognize no-wait mode on input and treat such requests as normal input-with-wait requests.

#### Output Wait Modes

Normally, V+ waits for each I/O operation to be completed before continuing to the next program statement.

The following example causes V+ to wait for the entire record of 50 spaces to be transmitted (about 50 ms for 9600 baud rate) before continuing to the next program statement.

```
TYPE /X50
```

Similarly, WRITE program command keywords to disk files will wait for any required physical output to complete before continuing.

This waiting is not performed if the /N (no wait) format control specifier is used in an output keyword. Instead, V+ immediately executes the next program statement. The IOSTAT function checks whether or not the output has completed. It returns a value of zero if the previous I/O is not complete. If a second output keyword for a particular Logical Unit Number is encountered before the first no-wait operation has completed, the second statement automatically waits until the first is done. This scheme means the no-wait output is effectively double-buffered. If an error occurs in the first operation, the second operation is canceled, and the IOSTAT value is correct for the first operation. The IOSTAT function can be used with a second parameter of 3 to explicitly check for the completion of a no-wait write.

#### Disk File Operations

Disk file operations that are available with the V+ system are described below.

The FDIRECTORY monitor command keyword is used to display the names of the files in a directory. A disk file can be accessed either sequentially, where data records are accessed from the beginning of the file to its end, or randomly, where data records are accessed in any order. Refer to Advanced Disk Operations on page 33 for more information about sequential and random access for disk files.

The FCMND program command keyword is used to perform the following operations.

- Rename a file
- Create a subdirectory
- Delete a subdirectory

**Additional Information:** The FCMND keyword is similar to other disk I/O keywords in that a logical unit must be attached and the success or failure of the

command is returned via the IOSTAT real-valued function. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about the IOSTAT function.

### Accessing the Disk Directories

The V+ directory structure is identical to that used by the IBM PC DOS operating system. For each file, the directory structure contains the file name, attributes, creation time and date, and file size. Directory entries may be read after successfully executing an FOPEN keyword.

Each directory record returned by a READ keyword operation contains an ASCII string with the information shown in the following table.

Table 3-2. Disk Directory Format Description

Byte	Size (Bytes)	Description
1 to 7	7	Attribute codes that are padded with blanks on the right side. The attribute field is blank if no special attributes are indicated.  The following characters are possible. <ul style="list-style-type: none"> <li>• D: Subdirectory</li> <li>• P: File is protected and cannot be read or modified</li> <li>• R: File is read-only and cannot be modified</li> <li>• S: File is system file</li> </ul>
9	1	ASCII tab character (9 decimal).
10 to 19	10	ASCII file size in sectors, right justified.
20	1	ASCII tab character (9 decimal).
20 to 28	9	File revision date with the format of dd-mm-yy.  <b>NOTE:</b> The file revision date is blank if the system date and time had not been set when the file was created or last modified.
29	1	ASCII tab character (9 decimal).
30 to 38	9	File revision time with the format hh:mm:ss.  <b>NOTE:</b> The file revision time is blank if the system date and time had not been set when the file was created or last modified.
39	1	ASCII tab character (9 decimal).
40+	---	ASCII file name and extension (size depends on file name length).

### Opening a Disk File

Before a disk file can be opened, the disk the file is on must be attached. The FOPEN\_ keywords open disk files (and file directories). These keywords associate a LUN with a disk file. Once a file is open, the READ, GETC, and WRITE keywords can be used to access the file. These keywords use the assigned LUN to access the file so that multiple files may be open on the same disk and the I/O operations for the different disk files will not affect each other.

**NOTE:** While a file is open for write or append access, another control program task cannot access that file. However, multiple control program tasks can access a file simultaneously in read-only mode.

#### **Writing to a Disk**

The following example writes the string stored in "\$in.string" to the disk file open on "dlun".

```
WRITE (dlun) $in.string
```

The following example returns any error generated during the write operation example above to the "error" variable.

```
error = IOSTAT(dlun)
```

#### **Reading from a Disk**

The following example reads from the open file on "dlun" up to the first CR/LF or end of file if it is encountered, and stores the result in "\$in.string". When the end of file is reached, V+ error number -504 (Unexpected end of file) is generated. The IOSTAT function must be used to recognize this error and halt reading of the file.

```
DO
  READ (dlun) $in.string
  TYPE $in.string
UNTIL IOSTAT(dlun) == -504
```

**Additional Information:** The GETC function reads the file byte-by-byte if you want to examine individual bytes from the file or if the file is not delimited by CR/LFs.

#### **Multi-functional Example**

The following example creates a disk file, writes to the file, closes the file, reopens the file, and reads back its contents.

```
AUTO dlun, i
AUTO $file.name
$file.name = "data.tst"

ATTACH (dlun, 4) "DISK"
IF IOSTAT(dlun) < 0 GOTO 100

FOPENW (dlun) $file.name
IF IOSTAT(dlun) < 0 GOTO 100

FOR i = 1 TO 10
  WRITE (dlun) "Line "+$ENCODE(i)
  IF IOSTAT(dlun) < 0 GOTO 100
END

FCLOSE (dlun)
IF IOSTAT(dlun) < 0 GOTO 100

FOPENR (dlun) $file.name
IF IOSTAT(dlun) < 0 GOTO 100
READ (dlun) $txt
WHILE IOSTAT(dlun) > 0 DO
```

```

        TYPE $txt
        READ (dlun) $txt
    END
    IF (IOSTAT(dlun) < 0) AND (IOSTAT(dlun) <> -504) THEN
100 TYPE $ERROR(IOSTAT(dlun))
    END
    FCLOSE (dlun)
    IF IOSTAT(dlun) < 0 THEN
        TYPE $ERROR(IOSTAT(dlun))
    END
    DETACH (dlun)

```

## Advanced Disk Operations

This section provides additional information for using the FOPEN and FOPENR program command keywords for use with advanced disk operations in the V+ system.

### Variable Length Records

The default disk file access mode is variable-length record mode. In this mode, records can have any length up to a maximum of 512 bytes and can cross the threshold of 512-byte sectors. The end of a record is indicated by a Line-Feed character (10 decimal). The end of the file is indicated by the presence of a Ctrl+Z character (26 decimal) in the file.

Variable-length records should not contain any internal Line-Feed or Ctrl+Z characters as data. This format is used for loading and storing V+ programs, and is compatible with the standard ASCII file format. Variable-length record mode is selected by setting the record length parameter with the FOPEN\_ keyword to zero, or by omitting the parameter completely. In this mode, WRITE keywords automatically append Return (13 decimal) and Line-Feed characters to the output data, which makes it a complete record. If the /S format control specifier is used in an output specification, no Return/Line-Feed is appended. Then, any subsequent WRITE will have its data concatenated to the current data as part of the same record. If the /Cn format control specifier is used, n Return/Line-Feeds are written, creating multiple records with a single WRITE.

When a variable-length record is read using a READ keyword, the Return/Line-Feed sequence at the end is removed before returning the data to the V+ program. If the GETC keyword is used to read from a disk file, all characters are returned as they appear in the file including Return, Line-Feed, and Ctrl+Z characters.

### Fixed Length Records

In fixed-length record mode, all records in the disk file have the same specific length. Then there are no special characters embedded in the file to indicate where records begin or end. Records are contiguous and may freely cross the threshold of 512-byte sectors.

Fixed-length record mode is selected by setting the record length parameter in the FOPEN\_ keyword to the size of the record, in bytes. WRITE keywords pad data records with zero bytes or truncate records as necessary to make the record length the size specified. No other data bytes are appended and the /S format control specifier has no effect.

In fixed-length mode, READ keywords always return records of the specified length. If the length of the file is such that it cannot be divided into an even number of records, a READ of the last record will be padded with zero bytes to make it the correct length.

### **Sequential Access Files**

Normally, the records within a disk file are accessed in order from the beginning to the end without skipping any records. Such files are called sequential files. Sequential-access files may contain either variable-length or fixed-length records.

### **Random Access Files**

V+ supports random access only for files with fixed-length records. Random access is selected by setting the random-access bit in the mode parameter of the FOPEN\_ keyword. A nonzero record length must also be specified.

A specific record is accessed by specifying the record number in a READ or WRITE keyword. If the record number is omitted, or is zero, the record following the one last accessed is used.

Records are numbered starting with 1. The position of the first byte in the random-access record can be computed with the statement below.

```
byte_position = 1 + (record_number - 1) * record_length
```

### **Buffering and I/O Overlapping**

All physical disk I/O occurs as 512-byte sector reads and writes. Records are unpacked from the sector buffer on input and additional sectors are read as needed to complete a record. To speed up read operations, V+ automatically issues a read request for the next sector while it is processing the current sector. This request is called a pre-read.

Pre-read is selected by default for both sequential-access and random-access modes. It can be disabled by setting a bit in the mode parameter of the FOPEN\_ keyword. If pre-reads are enabled, opening a file for read access immediately issues a read for the first sector in the file.

Pre-read operations may actually degrade system performance if records are accessed in truly random order, since sectors would be read that would never be used. In this case, pre-reads should be disabled and the FSEEK keyword should be used to initiate a pre-read of the next record to be used.

**Additional Information:** The function IOSTAT(lun, 1) returns the completion status for a pending pre-read or FSEEK operation.

On output, records are packed into sector buffers and written after the buffers are filled. If no-wait mode is selected for a write operation by using the /N format control specifier, the WRITE keyword does not wait for a sector to be written before allowing program execution to continue.

In random-access mode, a sector buffer is not normally written to disk until a record not contained in that buffer is accessed. The FEMPTY keyword empties the current sector buffer by immediately writing it to the disk.

A file may be opened in non-buffered mode, which is much slower than normal buffered mode, but it guarantees that information that is written will not be lost due to a system crash or power failure. This mode was intended primarily for use with log files that are left opened over an extended period of time and intermittently updated. For these types of files, the additional significant overhead of this mode is not as important as the benefit.

When a file is being created, information about the file size is not stored in the disk directory until the file is closed. Closing a file also forces any partial sector buffers to be written to the disk. Aborting a program does not force files associated with it to be closed. The files are not closed and the directory is not updated until a KILL keyword is executed or until the aborted program is executed again.

# Chapter 4: Motion Control Operations

---

This section describes motion control operations with the V+ system.

## 4.1 Motion Control Overview

V+ executes motion control statements immediately and does not wait for motion to complete before continuing to the next statement. This forward processing is the normal method for basic motion control operations.

**Additional Information:** The forward processing behavior can be changed by breaking continuous-path operation. Refer to Continuous-Path Trajectories on page 36 for more information.

In the example below, if the CP system switch is ON, the output signal 1 will turn ON immediately after the robot begins motion commanded by the first MOVE statement. The second MOVE statement will not be executed until the first MOVE to location "part.1" is complete. Signal 2 will turn ON immediately after the robot begins moving to location "part.2".

```
MOVE part.1
SIGNAL 1
BREAK
MOVE part.2
SIGNAL 2
```

## 4.2 Joint-interpolated Motion vs. Straight-line Motion

The path a robot takes when moving from one location to another can be either a joint-interpolated motion or a straight-line motion. Joint-interpolated motions move each joint at a constant velocity (except during the acceleration / deceleration phases) and are limited by the slowest joint.

Straight-line motions ensure that the robot tool tip traces a straight line from location-to-location in applications that are path sensitive.

The following statement will cause the robot to move to the location "pick" using joint-interpolated motion.

```
MOVE pick
```

The following statement will cause the robot to move to the location "pick" using straight-line motion.

```
MOVES pick
```

## 4.3 Safe Approaches and Departures

In many cases it is necessary to approach a location from a distance offset along the tool Z-axis or depart from a location along the tool Z-axis before moving to the next location.

The example below will cause the robot to safely approach and depart the "place" location with a 50 mm Z-axis distance. The robot will move with joint-interpolated motion with the following sequence.

1. Move to a location 50 mm above the "place" location.
2. Move down in the Z direction to the "place" location.
3. Move up in the Z direction 50 mm above the "place" location.

```
APPRO place, 50
MOVE place
DEPART 50
```

**IMPORTANT:** The APPRO operation is relative to the location specified and the DEPART operation is relative to the current tool orientation. For robots with 4 degrees of freedom using tool offsets of format X, Y, Z, 0, 180, Roll, the behavior of these keywords will be similar with motion aligned with the robot World Z-axis. For robots with 6 degrees of freedom or robots with 4 degrees of freedom using tool offsets with a different orientation in yaw and pitch, the behavior of these keywords will be different, and motion may not be aligned with the robot World Z-axis.

**NOTE:** Using the keywords APPROX, DEPARTS, and MOVES in the example above will cause the same sequence but with straight-line motion instead.

## 4.4 Continuous-Path Trajectories

Making smooth transitions between motion segments without stopping the robot motion is called continuous-path operation. This is the normal method V+ uses to perform robot motions. If desired, continuous-path operation can be disabled with the CP system switch keyword. When the CP switch is disabled, the robot decelerates and stops at the end of each motion segment before beginning to move to the next location.

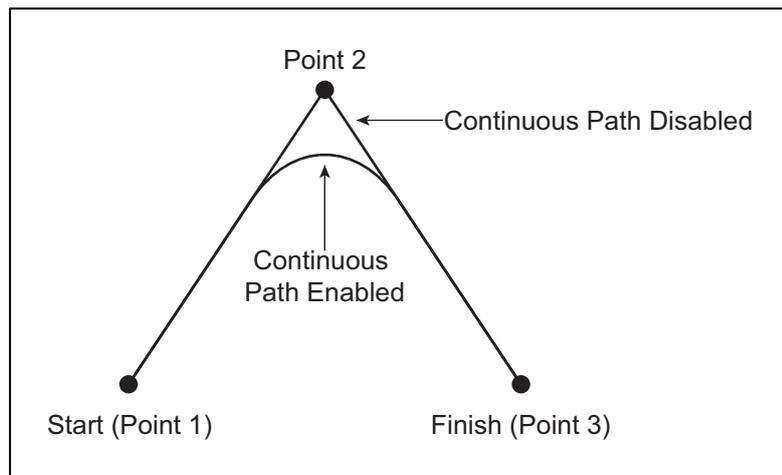


Figure 4-1. Continuous Path Enabled / Disabled

**NOTE:** Disabling continuous-path operation does not affect forward processing.

When a single motion keyword such as MOVE is executed by V+, the robot begins moving toward the location by accelerating smoothly to the commanded speed. When the robot is close to the destination, the robot decelerates smoothly to a stop at the location specified. This motion is referred to as a single motion segment. The following statement will produce a single motion segment to a location defined as "pick".

```
MOVE pick
```

When a sequence of motion keywords are executed by V+, the robot begins moving towards the first location by accelerating smoothly to the commanded speed. When the robot is close to the first destination, it does not decelerate to a stop. It will seamlessly transition with a trajectory towards the next location with a behavior referred to as blending. When the first motion segment reaches the deceleration portion of the motion, the robot will begin the acceleration portion of the next motion segment. The shape of the blended motion is determined by many parameters including the length, acceleration profiles, acceleration, deceleration, and speeds of both motion segments. The robot will decelerate to a stop when it reaches the last location in the sequence. The following statement demonstrates this and will produce a continuous-path trajectory with motion blending past the first location and then decelerating to a stop at the second location.

```
MOVE loc.1
MOVE loc.2
```

**Additional Information:** Continuous-path transitions can occur between any combination of straight-line and joint-interpolated motions. For example, a continuous motion could consist of a straight-line motion with the DEPARTS keyword followed by a joint-interpolated motion with the APPRO keyword and a final straight-line motion with the MOVES keyword. Any number of motion segments can be combined this way.

**NOTE:** The example above will decelerate to a stop between moves if the CP system switch is disabled. Refer to Using System Switch Keywords on page 22 for more information.

## 4.5 Breaking Continuous-Path Operation

Certain program command keywords cause program execution to be suspended until the current robot motion reaches its destination location and comes to a stop. This is called breaking continuous path. This functionality is useful when the robot must be stopped while some operation is performed (for example, closing the end-effector).

The following example demonstrates breaking the continuous path.

```
MOVE loc.1
BREAK
SIGNAL 1
```

In the example above, the MOVE keyword makes the robot move to "loc.1". Program execution continues and the BREAK operation is executed. This causes the program execution to temporarily stop forward processing of the program until the move to "loc.1" is completed. After the move is completed, the SIGNAL 1 operation is executed and the signal is turned ON.

**Additional Information:** The keywords BREAK, CPOFF, DETACH(0), HALT, OPENI, PAUSE, and TOOL cause V+ to suspend program execution until the robot stops.

The robot decelerates to a stop when the BRAKE keyword is executed or when the REACTI operation is executed. These events could happen at any point within a motion segment.

The robot decelerates to a stop if no new motion control operation is executed before the current motion completes. This situation can occur with the following conditions.

- A WAIT or WAIT.EVENT keyword is executed if the condition is not satisfied before the robot motion completes.

- A PROMPT keyword is executed and no response is entered before the motion completes.
- Keyword execution between the motion operations takes longer to execute than the robot takes to perform its commanded motion.

## 4.6 Procedural Motion

The ability to move in straight lines and joint-interpolated arcs is built into the basic operation of V+. The robot tool can also move along a path that is prerecorded, or described by a mathematical formula. These motions are performed with procedural motion.

Procedural motion utilizes a program loop that computes many short motions and issues the appropriate motion requests. The path is typically computed and stored in a array before starting the loop. The parallel execution of robot motions and non-motion keywords allows each successive motion to be defined without stopping the robot. The continuous-path feature of V+ automatically smooths the transitions between the computed motion segments.

The following example demonstrates procedural motion where the robot tool is moved along a trajectory described by locations stored in the array path[]. The LAST keyword is used to determine the size of the array. The robot tool moves at the constant speed of 0.75 inch per second through each location defined in the array path[].

```
SPEED 0.75 IPS ALWAYS
FOR index = 0 TO LAST(path[])
  MOVES path[index]
END
```

Alternatively, it is common to space the locations equidistant along the path, increase the acceleration / deceleration, and use the DURATION keyword to control the time the motion will take to complete the move to the next point. This will effectively define the velocity based on the distance between any two points in the path.

The following example demonstrates procedural motion where the robot tool is moved along a circular arc. The path is not prerecorded and it is described mathematically, based on the radius and center of the arc to be followed.

The program segment below assumes that a real variable radius has already been assigned the radius of the desired arc, and "x.center" and "y.center" have been assigned the respective coordinates of the center of curvature.

The variables start and last are assumed to have been defined to describe the portion of the circle to be traced. Finally, the variable "angle.step" is assumed to have been defined to specify the angular increment to be traversed in each incremental motion. Because the DURATION keyword is used, the program moves the robot tool "angle.step" degrees around the arc every 0.5 seconds.

When this program segment is executed, the X and Y coordinates of points on the arc are repeatedly computed. They are then used to create a transformation that defines the destination for the next robot motion segment.

```
DURATION 0.5 ALWAYS
FOR angle = start TO last STEP angle.step
  x = radius*COS(angle)+x.center
  y = radius*SIN(angle)+y.center
  MOVE TRANS(x, y, 0, 0, 180, 0)
END
DURATION 0 ALWAYS
```

**Additional Information:** It may be useful to increase acceleration during the procedural motion so that the program acceleration does not constrain the motion.

## 4.7 Motion Control Timing Considerations

Because of the computation time required by V+ to perform the transitions between motion segments, there is a limit on how closely spaced commanded locations can be. When locations are too close together, there is not enough time for V+ to compute and perform the transition from one motion to the next and there will be a break in the continuous-path motion. This means that the robot stops momentarily at intermediate locations.

The minimum spacing that can be used between locations before this effect occurs is determined by the time required to complete the motion from one location to the next. Straight-line motions can be used if the motion segments take more than twice the trajectory period each. Joint-interpolated motions can be used with motion segments as short as about one single trajectory period each.

## 4.8 Robot Speed and Performance

Robot speed is referred to in this manual as how fast the robot moves between the acceleration and deceleration phases of a motion. It can also be considered as the magnitude of the constant velocity portion of the velocity profile. Robot speed is sometimes more generally referred to as how fast the robot gets from one position to another (referred to in this manual as robot performance).

### Robot Speed

The speed of a robot move based on full speed is determined by program speed and monitor speed as described below. With monitor speed and program speed set to 100, the robot moves at its full speed. With monitor speed set to 50 and program speed set to 50, the robot moves at 25% of its full speed.

- The program speed set with the SPEED program command keyword. This speed is set to 100 when program execution begins.
- The monitor speed set with the SPEED monitor command keyword or a SPEED program command keyword that specifies "MONITOR" in the units parameter. This speed is normally set to 50 at system startup. Monitor speed can also be set with the Sysmac Studio or ACE software.

**Additional Information:** The effects of the two SPEED operations above are different. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information. Refer to *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for information about setting an initial monitor speed that takes effect on boot up.

A robot move has the following three phases.

1. Acceleration phase where the robot accelerates to the maximum speed specified for the move.
2. Velocity phase where the robot moves at a rate not exceeding the specified maximum speed. The robot may not reach the specified maximum speed based on the acceleration / deceleration values and the distance of the move.

3. Deceleration phase where the robot decelerates to a stop (or transitions to the next motion).

The robot speed between the acceleration and deceleration phases is specified as either a percentage of full speed or an absolute rate of travel of the robot tool tip. Speed set as a percentage of full speed is the default specification.

### Robot End-effector / Tool Tip Speed Considerations

To move the robot tool tip at an absolute rate of speed, a speed rate in inches per second or millimeters per second is specified in the SPEED program command keyword as shown in the example below. This example specifies an absolute tool tip speed of 25 millimeters per second for all robot motions until the next SPEED keyword is executed.

```
SPEED 25 MMPS ALWAYS
```

In order for the tool tip to move at the specified speed, the following must be considered.

- The monitor speed must be 100.
- The locations must be far enough apart so that the robot can accelerate to the desired speed and decelerate to a stop at the end of the motion.

### Robot Performance

Robot performance is a function of the SPEED keyword settings and the following factors.

- The robot acceleration profile and ACCEL settings must be considered. The ACCEL keyword can scale down these maximum rates so that the robot acceleration and / or deceleration takes more time.
- SCALE.ACCEL and SCALE.ACCEL.ROT system switch states have an impact on robot performance. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.
- You can also define optional acceleration profiles that alter the rate of change for acceleration and deceleration.
- The location tolerance settings (COARSE / FINE, NULL / NONULL) for the move must be considered. The more accurately a robot must get to the actual location, the more time the move will take.
- Any DURATION setting must be considered. DURATION forces a robot move to take a minimum time to complete regardless of the SPEED settings.
- The maximum allowable velocity must be considered. Maximum velocity is factory set.
- The inertial loading of the robot and the tuning of the robot must be considered.
- Straight-line vs. joint-interpolated motions must be considered. Straight-line and joint-interpolated paths produce different dynamic responses and therefore different motion times.

**Additional Information:** Robot performance for a given application can be greatly enhanced or severely degraded by these settings. For example, a heavily loaded robot may actually show better performance with slower SPEED and ACCEL settings which lessens overshoot at the end of a move and allows the robot to settle more quickly. Applications such as picking up bags of product with a vacuum gripper do not require high accuracy and can generally run faster with a COARSE tolerance.

## 4.9 Motion Modifiers

The following keywords modify the characteristics of individual motions.

- ABOVE / BELOW
- ACCEL
- BREAK
- COARSE / FINE
- CPON / CPOFF
- DURATION
- FLIP / NOFLIP
- LEFTY / RIGHTY
- NOOVERLAP / OVERLAP
- NULL / NONULL
- BRAKE
- SINGLE / MULTIPLE
- SPEED

**Additional Information:** Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.



# Chapter 5: Variables and Data Types

---

This section describes variable creation and use for the V+ system.

## 5.1 Variable Creation

Variables and their data types can be declared explicitly or with their first use in a program. The first use of a variable determines its data type and allocates storage for that variable. This method is referred to as dynamic allocation. Numeric data types, string data types, and transformation arrays up to three dimensions can be created with dynamic allocation.

To explicitly create a new variable without relying on dynamic allocation, simple statements can be used as shown below. If the variable has already been defined, the following statements will only assign new values to them.

```
real_var = 13.65
$string1 = "ABC123"
```

## 5.2 Variable Classifications

Variables can have a classification of External, Global, Local, or Automatic. This classification determines how a variable can be used and altered by different calling instances of a program.

Variable definition statements must occur at the top of a program before other keyword statements. Only the .PROGRAM statement, comments, and blank lines may appear before a variable definition statement.

**Additional Information:** Once a variable has been assigned to a classification, an attempt to assign the variable to a different classification will result in the error \*Attempt to redefine variable class\*.

**NOTE:** Variable declaration statements with the AUTO keyword cannot be added to a program that is currently running as a task on the stack. A program must be stopped and removed from the stack before adding additional AUTO variables is possible.

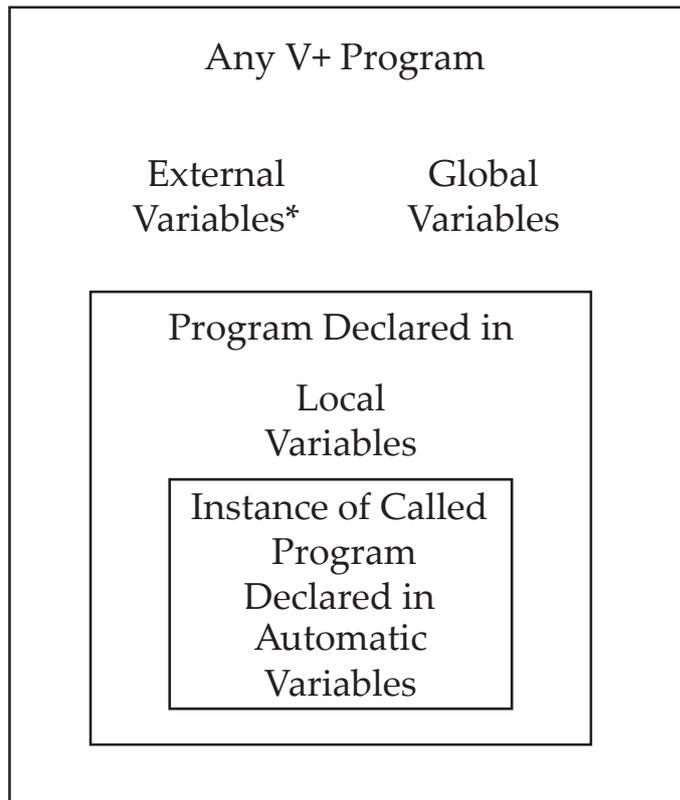
Use the information in this section to understand variable classification details.

### Variable Classification Scope

The scope of a variable refers to the range of programs that can access that variable. The following figure shows the scope of the different variable classifications.

A variable can be altered by the program(s) indicated in the area of the box it is in plus any programs that are in smaller boxes.

**NOTE:** When a program defines an External, Automatic, or Local variable, any Global variables of the same name created in other programs are not accessible.



*Figure 5-1. Variable Scoping*

**NOTE:** External variables can only be used in a V+ Program when defined in the NJ-series Robot Integrated CPU Unit. Refer to External Variables on page 45 for more information.

Use the figure below to understand Global, Local, and Auto variable definitions when various program calls are made.

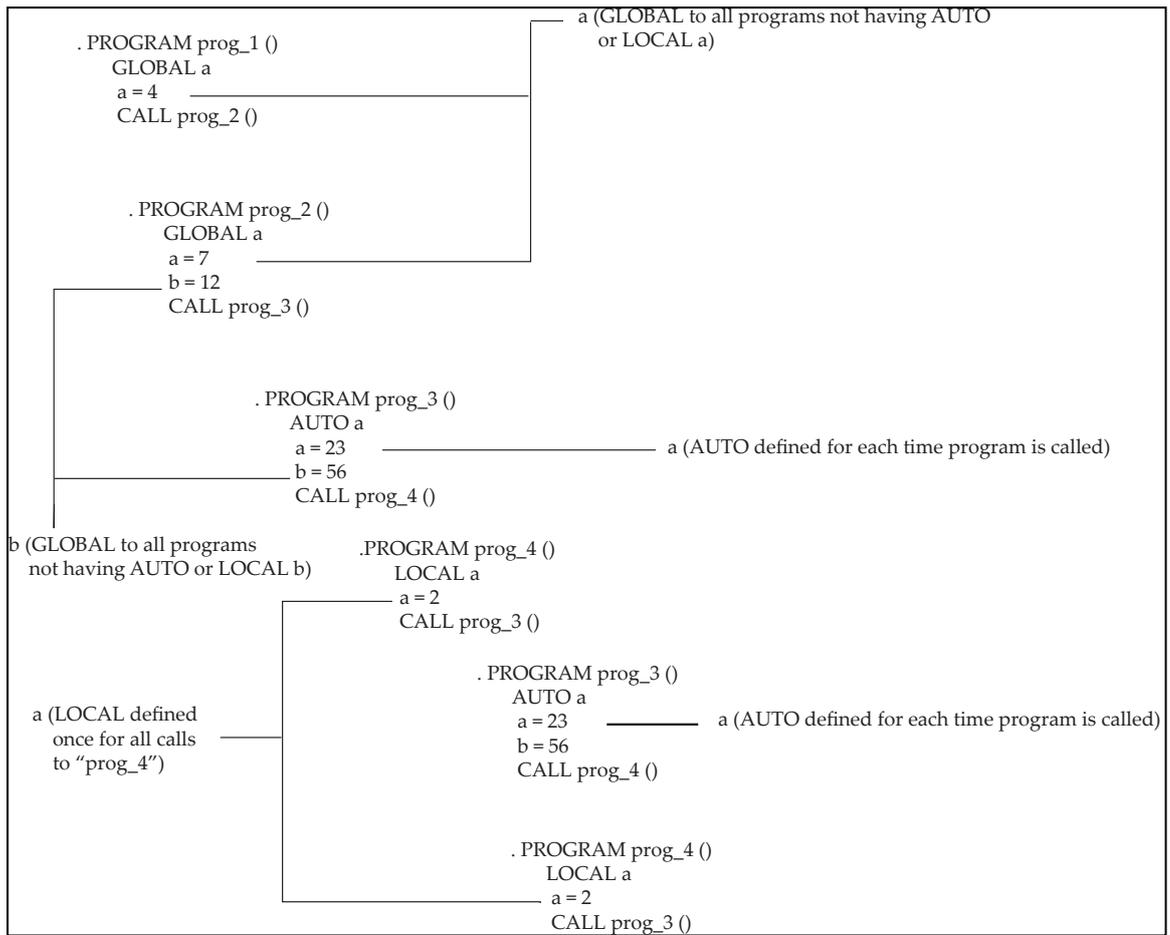


Figure 5-2. Variable Scope Example

## External Variables

External variables are shared between the V+ system and the NJ-series Robot Integrated CPU Unit. To use External variables, they must be defined in the NJ-series Robot Integrated CPU Unit and the V+ system.

The EXTERNAL program command keyword is used to define External variables in the V+ system. After the External variables are defined, they can be used in any V+ program executed in any task.

Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about use of the EXTERNAL program command keyword. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for more information about External variable use with the NJ-series Robot Integrated CPU Unit.

Use the following considerations when creating External variables.

- A previously defined External variable will become unavailable if it is deleted or all V+ programs that reference it are removed from system memory (using a DELETE or ZERO keyword).
- External variables can be accessed from V+ programs and with the Monitor Window in the same way as Global variables.
- The name of the variables must be the same in the V+ system and in the NJ-series Robot Integrated CPU Unit.

- Variable names must begin with a letter.
- Variable names can only use 0-9, a-z, period, or underscore character types.
- The maximum length of a variable name is 15 bytes. Only the first 15 characters in a variable name are significant.
- External variables cannot be defined with the same name of other variable classifications (GLOBAL, AUTO, LOCAL) in the same program. In the event an External and Global variable with the same name are defined in the V+ system, an error will occur when the duplicated variable name is defined.
- When defining a new External variable within a V+ program, the EXTERNAL statement must precede any executable program statements other than comment lines or other variable type definition statements.
- External variables defined as an array must have an array size of 0 to 99. The array size is defined in the NJ-series Robot Integrated CPU Unit. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for more information.

### External Variable Data Types

The External variable type and array size is defined in the NJ-series Robot Integrated CPU Unit. The V+ system will automatically use the External variable as a double-precision data type. String, structure, variable length arrays, and other data types are not supported.

### Errors Associated with External Variables

An error will occur during V+ program execution if any of the following conditions are present.

- An *\*Undefined program or variable name\** (-406) error will occur if the External variable does not exist in the NJ-series Robot Integrated CPU Unit.
- An *\*Illegal array index\** (-404) will occur if the array limits are exceeded [0 to 99].
- An *\*Illegal array index\** (-404) will occur if the variable types differ between the V+ definition and the NJ-series Robot Integrated CPU Unit.
- A *\*Variable type mismatch\** (-465) will occur if the External variable types are not supported.

### Global Variables

Global variables are available for use across all executing programs. Unless a variable has been defined as auto, local, or external, it will be considered global with the following considerations.

- Defining the same variable as local or auto in a program will override the variable's global classification.
- The global variable will persist in memory and will be available to any executing program until the variable is deleted or all programs that reference the variable are removed from system memory using the DELETE or ZERO keyword operations.

A global variable can be explicitly defined using the GLOBAL program command keyword as shown in the example below.

```

GLOBAL LOC my.loc
GLOBAL REAL my.real
GLOBAL DOUBLE my.double
GLOBAL $my.string
GLOBAL %my.beltvar

```

**IMPORTANT:** Global variables are very powerful and should be used selectively. Good programming practice suggests using local and automatic variables instead of global variables whenever possible.

## Local Variables

Local variables are available only within an executing program that defines it. Local variables have the following properties.

- Local variables are local to all copies (calling instances) of a program, not just a particular calling instance of that program.
- A local variable can be referenced only within its own program.
- If a variable is listed in a LOCAL statement, any global variable with the same name cannot be accessed directly by that program.
- The names of local variables can be selected without regard for the names of local variables defined in other programs.
- Local variables can be changed only by the program in which they are defined.
- Local variables are allocated only once during program execution and their values are preserved between successive subroutine calls. These values are also shared if the same program is executed by multiple program tasks.
- The values of local variables are not saved or restored by the STORE or LOAD monitor command keywords.

**IMPORTANT:** If a program that uses local variables is called by several different program tasks or called recursively by a single task, the values of those variables can be modified by the different program instances and cause unpredictable program errors. Therefore, automatic variables should be used for all temporary local variables to minimize the chance of errors. Refer to Automatic Variables on page 47 for more information.

**NOTE:** Do not confuse a location variable (LOC) with a local variable (LOCAL).

A local variable can be explicitly defined using the LOCAL program command keyword as shown in the example below.

```

LOCAL LOC my.loc
LOCAL REAL my.real
LOCAL DOUBLE my.double
LOCAL $my.string
LOCAL %my.beltvar

```

## Automatic Variables

An automatic variable is automatically created when the program that defines it is executed. It is defined only within the current program. Automatic variables have the following properties.

- An automatic variable can be referenced only by the specific calling instance of a program.

- An automatic variable has an undetermined value when a program is initially executed and has no value after the program exits.
- If a variable is listed in an AUTO statement, any global variables with the same name cannot be accessed directly by the program.
- The values of automatic variables are not saved or restored by the STORE or LOAD monitor command keywords.
- The names of automatic variables can be selected without regard for the names of variables defined in any other programs.
- Automatic variables are allocated each time the program is called and their values are not preserved between successive subroutine calls.
- A separate copy of an automatic variable is created each time a program is called, even if it is called simultaneously by several different program tasks or called recursively by a single task.
- The storage space for automatic variables is allocated on the program execution stack. If the stack is too small for the number of automatic variables defined, the task execution will stop.
- Automatic variables cannot be deleted with the DELETE\_ program command keywords.

An automatic variable can be explicitly defined using the AUTO program command keyword as shown in the example below.

```
AUTO LOC my.loc
AUTO REAL my.real
AUTO DOUBLE my.double
AUTO $my.string
AUTO %my.beltvar
```

## 5.3 Variable Name Requirements

Use the following requirements when creating new variable names.

- Keywords cannot be used as variable names. Do not name variables the same as any keywords.
- Variable names must begin with a letter.
- Only letters, numbers, periods, and the underscore character (\_) may be used in variable names.
- Variable names are limited to 15 characters in length. Variable name designations over 15 characters will automatically truncate without warning. For example, the variable name "variable\_name\_too\_long" will automatically truncate to "variable\_name\_t".
- Because the V+ system automatically creates default system variable names, avoid creating variable names that begin with two or three letters followed by a period to prevent coincidental variable name duplications. For example, sv.error, tsk.idx, and tp.pos1 are variable names that should be avoided.

## 5.4 Variable Initialization

Before a variable can be used it must be initialized. String and numeric variables can be initialized by placing them on the left side of an assignment statement. Use the following considerations when initializing variables.

- A variable can never be initialized on the right side of a statement.
- Strings, numeric variables, and location variables can be initialized by being loaded from a disk file.
- Strings and numeric variables can be initialized with the PROMPT keyword.
- Transformations and precision points can be initialized with the SET or HERE program command keywords. They can also be initialized with the HERE monitor command keyword or with the teach pendant.

The following statements initialize the variables "var\_one" and "\$var\_two".

```
var_one = 36
$var_two = "two"
```

The following statement initializes "var\_one" if "var\_two" has already been initialized. If this statement is executed when "var\_two" has not been initialized, an *\*Undefined value\** error will be returned.

```
var_one = var_two
```

The following statement is valid only if "var\_one" has been initialized in a previous statement.

```
var_one = var_one + 10
```

## 5.5 Variable Operators

The information in this section provides variable operator details.

### Assignment Operators

The equal sign (=) is used to assign a value to a numeric or string variable. The variable being assigned a value must appear by itself on the left side of the equal sign. The right side of the equal sign can contain any initialized variable or value of the same data type as the left side, or any expression that resolves to the same data type as the left side.

Any variables used on the right side of an assignment operator must have been previously initialized. Location variables require the use of the SET keyword for a valid assignment statement.

The following statement is unacceptable for location and precision-point variables.

```
loc_var1 = loc_var2
```

V+ can resolve an unknown transformation used in a compound transformation on the left side of an assignment operator. This is useful when initializing location variables relative to other frames. In the following example, "slot.offset" will be initialized as TRANS (200,200,200,0,0,0).

```
SET pallet_frame = TRANS(100,100,100,0,0,0)
SET taught_loc = TRANS(300,300,300,0,0,0)
SET pallet_frame:slot.offset = taught_loc
```

### Mathematical Operators

The following mathematical operators are available for use.

Table 5-1. Mathematical Operators

Operator	Function
+	Addition
-	Subtraction or unary minus
*	Multiplication
/	Division
MOD	Modular (remainder) division

## Relational Operators

Relational operators are used in expressions that yield a boolean value. The resolution of an expression containing a relational operator is always -1 (true) or 0 (false) and indicates if the specific relation stated in the expression is true or false.

**NOTE:** A relational operator never changes the value of the variables on either side of the relational operator.

The most common use of relational expressions is with the control structures. V+ uses the standard relational operators shown in the following table.

Table 5-2. Relational Operators

Operator	Function
==	Equal to
<	Less than
>	Greater than
<= or =<	Less than or equal to
>= or =>	Greater than or equal to
<>	Not equal to

The following boolean statements resolve to -1 (true) when "x" has a value of 6 and "y" has a value of 10.

```
x < y
y >= x
y <> x
```

The following boolean statements resolve to 0 (false) when "x" has a value of 6 and "y" has a value of 10.

```
x > y
x <> 6
x == y
```

Observe the difference between the assignment operator "=" and the relational operator "==" in the example below. In this statement, "z" is assigned a value of 0 since the boolean expression "x==y" is false and resolves to 0.

```
z = x == y
```

## Logical Operators

Logical operators affect the resolution of a boolean variable or expression and combine several boolean expressions so they resolve to a single Boolean value. V+ uses the standard logical operators shown in the following table.

Table 5-3. Logical Operators

Operator	Function
NOT	Complement the expression or value. Makes a true expression or value false. Makes a false expression or value true.
AND	Both expressions must be true before the entire expression is true.
OR	Either expression must be true before the entire expression is true.
XOR	One expression must be true and one must be false before the entire expression is true.

The following statements resolve to -1 (true) when "x" has a value 6 and "y" has a value of 10.

```
NOT (x == 7)
(x > 2) AND (y =< 10)
```

The following statements resolve to 0 (false) when "x" has a value 6 and "y" has a value of 10.

```
NOT (x == 6)
(x < 2) OR (y > 10)
```

## String Operators

Strings can be concatenated (joined) using the plus sign.

The following statement results in the variable "\$full\_name" having the value "Omron Robotics and Safety Technologies, Inc."

```
$name = "Omron Robotics "
$incorp = ", Inc."
$full_name = $name + "and Safety Technologies" + $incorp
```

## Order of Evaluation

Expressions containing more than one operator are not evaluated in a simple left to right manner. The following table lists the order in which operators are evaluated.

Within an expression, functions are evaluated first with operator priority according to the table. The order of evaluation can be changed using parentheses. Operators within each pair of parentheses, starting with the most deeply nested pair, are completely evaluated according to the rules in the following table before any operators outside the parentheses are evaluated. Operators on the same row of the table are evaluated strictly left-to-right.

Table 5-4. Operator Priority Levels

Priority Level	Operator
Highest	NOT, COM

Priority Level	Operator
...	- (Unary minus)
	*, /, MOD, AND, BAND
	+, -, OR, BOR, XOR, BXOR
Lowest	==, <=, >=, <, >, <>

### Operator Evaluation Order Examples

Use the examples below to understand the operator order of evaluation in expressions.

The following example demonstrates evaluation priority for operators on different priority levels. Execution of this statement results in "1 + 1 x 3 = 4" displayed in the Monitor Window.

```
TYPE "1 + 1 x 3 = ", 1 + 1 * 3
```

The following example demonstrates the use of parentheses to change evaluation priority for operators on different priority levels. Execution of this statement results in "(1 + 1) x 3 = 6" displayed in the Monitor Window.

```
TYPE "(1 + 1) x 3 = ", (1 + 1) * 3
```

The following example demonstrates simple left-to-right evaluation priority for operators with the same priority level. The statement is evaluated from left-to-right and results with "10 MOD 3 AND 10 MOD 3 = 2" displayed in the Monitor Window.

```
TYPE "10 MOD 3 AND 10 MOD 3 = ", 10 MOD 3 AND 10 MOD 3
```

## 5.6 Numeric Representation

Numeric values can be represented in the standard decimal notation or in scientific notation, as described in the previous section. Numeric values can also be represented in octal, binary, and hexadecimal form. The following table shows the required form for each integer representation.

Prefix	Representation	Example
None	Decimal	-193
^B	Binary with a maximum of 8 bits	^B1001
^	Octal	^346
^H	Hexadecimal	^H23FF
^D	Double-precision	^D20000000

**NOTE:** V+ does not have a specific logical (Boolean) data type. Any numeric value, variable, or expression can be used as a logical data type. V+ considers 0 to be false and any other value to be true. Refer to Logical Constants on page 54 for more information.

### Numeric Expressions

In almost all situations where a numeric value of a variable can be used, a numeric expression can also be used. The following examples result in variable "x" having the same value of 3.

```
x = 3
x = 6/2
```

```
x = SQRT(9)
x = SQR(2) - 1
x = 9 MOD 6
```

## 5.7 String Data Types

Variables that include string data are preceded by a dollar sign (\$). Values assigned to string variables must be enclosed with quotes. The following limitations exist with string variable values.

- A string variable can be assigned any ASCII value ranging from 32 (space) to 126 (~) with the exception of 34 (").
- Strings can contain up to 128 characters.
- String variables can contain values from 0 to 255.

**NOTE:** The dollar sign is not included in the 15 character limit for variable names.

The following statement will create a new variable with the name "string\_variable" if the variable does not already exist, and a value of "ABC123" will be assigned to it.

```
$string_variable = "ABC123"
```

The following statement will return an error because an attempt to assign a real value to a string variable was made (refer to the \$ENCODE keyword in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information).

```
$string_variable = 123.0
```

### ASCII Values

An ASCII value is the numeric representation of a single ASCII character. An ASCII value is specified by prefixing a character with an apostrophe ('). Any ASCII character from the space character (decimal value 32) to the tilde character (7e, decimal value 126) can be used as an ASCII constant. The following are valid ASCII constants.

```
'A'   '1'   'v'   '%'
```

**Additional Information:** The ASCII value '1 (decimal value 49) is not the same as the integer value 1 (decimal value 1) and not the same as the string value "1".

## 5.8 Real Integer Data Types

Numbers that have a whole number and a fractional part (or mantissa and exponent if the value is expressed in scientific notation) belong to the data type real. Numeric values having only a whole number belong to the data type integer. In general, V+ does not require differentiation between these two data types.

If an integer is required and a real value is supplied, V+ converts the real to an integer by rounding (not truncation). Where real values are required, V+ considers an integer a special case of a real that does not have a fractional part.

The default real type is a signed, 32-bit IEEE single-precision number. Real values can also be stored as 64-bit IEEE double-precision numbers if they are specifically typed using the DOUBLE keyword.

## Real and Integer Value Ranges

Use the ranges below when assigning values to real and integer variables.

Table 5-5. Real and Integer Value Ranges

Data Type	Range
Integer	-16,777,216 to 16,777,215
Single-precision Real	-1E+38 to 1E+38 with 24 bits of precision
Double-precision Real	-1E+307 to 1E+307 with 52 bits of precision

## 5.9 Logical Constants

The following logical constants are available with V+. These constants can be used anywhere that a Boolean expression is expected.

- TRUE (resolves to -1)
- ON (resolves to -1)
- FALSE (resolves to 0)
- OFF (resolves to 0)

A logical value, variable, or expression can be used anywhere that an evaluation is required. In the following example, an input signal is evaluated. If the signal is ON (high) the variable 'dio.sample' is given the value true and the IF clause executes. Otherwise, the ELSE clause will execute.

```

dio.sample = SIG(1001)
IF dio.sample THEN
TYPE "The signal is ON"
ELSE
TYPE "The signal is OFF"
END

```

In the above example, the first two lines could be combined and replaced with the following statement.

```

IF SIG(1001) THEN

```

## 5.10 Location Data Types

Locations can be specified as transformations or precision points.

A transformation is a set of six components that identifies a location in cartesian space and the orientation of the motion device end-of-arm tooling at that location. A transformation can also represent the location of an arbitrary local reference frame.

Relative transformations allow you to make one location relative to another and to build local reference frames to which that transformation can be relative. For example, you may be building an assembly whose location in the workcell changes periodically. If all the locations on the assembly are taught relative to the world coordinate frame, each time the assembly is located differently in the workcell, all the locations must be retaught. If, however, you create a frame based on identifiable features of the assembly, you will have to reteach only the points that define the frame. Refer to Modifying Location Variables on page 55 for more information.

A precision point uniquely defines a location in space by including an element for each joint position of the mechanism's kinematics. A mechanism with four degrees of freedom will

require precision points with four elements, while a mechanism with six degrees of freedom will require 6 elements. Rotational joint values are measured in degrees and translational joint values are measured in millimeters. These values are absolute with respect to the joint zero positions determined by the robot hardware calibration.

## Creating Location Variables

The simplest method of creating a location variable is to place the robot at a location and then execute the following statement.

```
HERE loc_var
```

A location can be specified using either the six components or by specifying the state that the robot joints would be in when a location is reached. The former method results in a transformation variable. Transformations are the most flexible and efficient location variables because they can be used in various arm configurations. They can also be used in different contexts, for example, tool offsets or slot-offsets could be used in different arm positions or while accessing different pallet frames with the same pallet layout.

Precision points record the joint values of each joint of the robot. Precision points may be more accurate and they are the only way of extracting joint information that will allow you to move an individual joint. Precision points are identified by a leading pound sign (#).

The following statement will create the precision point "pick" that is equal to the current robot joint values.

```
HERE #pick
```

## Modifying Location Variables

Use the following examples to understand the different methods used to modify location variables.

**Additional Information:** Transformations and precision points can be initialized with the SET or HERE Program Command Keywords

### *Duplicate a Location Variable*

The following example will duplicate an existing location variable and give the value of "loc\_value" to variable "loc\_name".

```
SET loc_name = loc_value
```

### *Create Location Variables Based on Modifications to Existing Variables*

The following example will create a location variable "loc\_name" and shift the location 5 mm in the positive X, Y, and Z directions from the variable "loc\_value".

```
SET loc_name = SHIFT(loc_value BY 5, 5, 5)
```

### *Using Relative Transformations*

Use the statements below to generate locations shown in Figure 5-3. . In this figure, the transformation "loc\_b" defines the transformation needed to get from the local reference frame defined by "loc\_a" to the local reference frame defined by "loc\_c".

The colon (:) operator used in the examples below represents a compound transformation operation. For example, loc\_a:loc\_b is the transformation loc\_b relative to loc\_a. The loc\_b is a completely unique and independent location transformation as shown in Figure 5-3. , but when used relative to loc\_a (whose pitch is 180 degrees), is the transformation to get to loc\_c.

The following statements will define a simple transformation and then move the robot to that location.

```
SET loc_a = TRANS (300, 50, 350, 0, 180, 0)
MOVE loc_a
BREAK
```

The following statements will move the robot to a location offset -50 mm in X, 20 mm in Y, and 30 mm in Z relative to "loc\_a" position and orientation.

```
MOVE loc_a:TRANS (-50, 20, 30)
BREAK
```

The following statements will define "loc\_b" to be the current location relative to "loc\_a" where "loc\_b" = -50, 20, 30, 0, 0, 0.

```
HERE loc_a:loc_b
BREAK
```

The following statements will define "loc\_c" as the vector of "loc\_a" and "loc\_b" where "loc\_c" = 350, 70, 320, 0, 180, 0.

```
SET loc_c = loc_a:loc_b
```

After the statements above are executed, "loc\_b" exists as a transformation that is completely independent of "loc\_a". The following statement moves the robot another -50 mm in the X direction, 20 mm in the Y direction, and 30 mm in the Z direction relative to "loc\_c".

```
MOVE loc_c:loc_b
```

Multiple transformations can be associated. If we define "loc\_d" to have the value of 0, 50, 0, 0, 0, 0, the following statements will move the robot to a position of X = -50 mm, Y = 70 mm, and Z = 30 mm relative to "loc\_a".

```
SET loc_d = TRANS (0, 50)
MOVE loc_a:loc_b:loc_d
```

If the transformation is required to go in the opposite direction (from "loc\_c" to "loc\_a"), it can be calculated with the following statement.

```
INVERSE (loc_b)
```

The statement below will move the robot back to "loc\_a".

```
MOVE loc_c:INVERSE (loc_b)
```



The following statement allocates space for a two-dimensional array named "array.two" and places "row 4, col 5" in row four, column five of the array.

```
$array.two[4,5] = "row 4, col 5"
```

The following statement allocates space for a three-dimensional array named "array.three" and places the value 10.5 in row two, column two, range four.

```
array.three[2,2,4] = 10.5
```

If any of the above statements were executed and the array had already been declared, the statement would merely place the value in the appropriate location. An error will occur if the data type of the value is different than the data type of the array.

The following statement results in allocation of array elements 0 to 15.

```
any_array[2] = 50
```

The following statements result in allocation of array elements 0 to 31.

```
any_array[2] = 50
any_array[20] = 75
```

## Global Array Access Restriction

V+ has a feature where global and local arrays are automatically extended as they are used. For efficiency, there is no interlocking of the array extension process between multiple tasks. A crash can occur if one task is extending or deleting an array while another is trying to access it. Custom V+ programs must be coded to avoid this problem using one of the following methods.

### Method 1

If there is a known reasonable upper-bound on the array dimensions, define by assigning an arbitrary value to it the highest element of the array. For multi-dimensional arrays, assign the highest element of each possible sub-array. This assignment prevents the arrays from extending.

### Method 2

Use the TAS function keyword to interlock access to the array. In this case, access to the array is handled exclusively from one or two subroutines that include the TAS to control access to the array.

## 5.12 Variable Context

When keywords are used to access local or automatic variables, the V+ system must know which program's variables are to be accessed. You must specify the program context to be used. The general syntax for a command that can access local and automatic variables is shown below where "command" represents the keyword name and "parameters" represents the keyword parameters.

```
command @task:program parameters
```

The optional element "@task:program" specifies the program context for any variables referenced in the keyword parameters. The "task" portion is an integer that specifies one of the system program tasks. The "program" portion of the context is the name of a program in the system memory.

The context can be specified in any of the formats described below. Refer to Interpretation of Context Specification for Variables on page 60 for a summary of the various context specifications that can be used.

1. "task:program" is blank ("@" is used as syntax). The context for referenced variables will be one of the following.
  - A blank context specification in an MCS keyword. The following example indicates that the variables are treated as though they are referenced within the program containing the MCS keyword. A keyword used as in the example below can be used to delete local variables after they are used by a routine regardless of which task is assigned to the executing program.
 

```
MCS "DELETEL @ parts[]"
```
  - Variables will be treated as though they are referenced within the program on the top of the stack for the robot control task (the last program listed by the STATUS keyword). For example, "STATUS 0".
2. "Task" is specified as a task number (0, 1, 2, etc.) and the portion ":program" is omitted. For example, "@1" is input. In this case, variables are treated as though they are referenced within the program on the top of the stack for the specified program task (the last program shown for that task by the STATUS keyword). For example, "STATUS 1".
3. "Task:" is omitted and "program" is specified (for example, "@sample" is used as syntax). In this case, variables are treated as though they are referenced within the most recent occurrence of the specified program on the task stack that is determined as follows.
  - If the program debugger is not in use in the Sysmac Studio or ACE software, the stack for the main control task (task 0) is used.
  - If the program debugger is in use in the Sysmac Studio or ACE software, the stack for the task being accessed by the debugger is used.
4. "Task:program" are both specified, where "task" is a task number (0, 1, 2, etc.), and "program" is a program name (for example, "@1:sample" is used as syntax). In this case the variables will be treated as though they are referenced within the most recent occurrence of the program on the stack for the specified program task.

When a context is specified, all the variables referenced in the statement will be considered as local or automatic in that program if applicable. Any variable referenced in the statement that is not found within that program will be considered a global variable. Because automatic variables and subroutine arguments exist only within a particular program instance and can vanish whenever that program exits, access to their values by monitor command keywords is limited by the following conditions.

- The program must be on a task stack. That is, it must appear in a program list shown by the STATUS keyword.
- The program task for the stack being referenced must not be active. This is to guarantee that the variable does not vanish while being accessed.

**NOTE:** Failure to meet the conditions above results in the error message:  
\*Undefined value in this context\*.

If automatic variables or subroutine arguments need to be accessed during debugging, the desired program task can be paused, the variables accessed, and execution of the task resumed.

## Interpretation of Context Specification for Variables

Use the table below to understand context specification for variables.

- "T" indicates the task that will be accessed.
- "P" indicates the program to be accessed.
- "None" indicates no specific task or program (global variables are accessed).
- "Current debug" indicates the task or program being accessed with the debugger in the Sysmac Studio or ACE software.
- "Top of stack" indicates the program at the top of the execution stack for the indicated task.

Table 5-6. Context Specification for Variables

Context Specification	Interpretation When Not Using Debugger	Interpretation When Using Debugger
None	T = None	T = Current debug
	P = None	P = Current debug
@ (see note)	T = Task 0	T = Current debug
	P = Top of stack	P = Current debug
@number	T = Task number	T = Task number
	P = Top of stack	P = Top of stack
@program	T = Task 0	T = Current debug
	P = Program	P = Program
@number:program	T = Task number	T = Task number
	P = Program	P = Program

**NOTE:** If "@" is specified in an MCS keyword in a program, T = Current task P = Current program (top of stack).

# Chapter 6: Monitor Command Programs

---

This section describes the functions and use of Monitor Command Programs.

## 6.1 Monitor Command Program Overview

Monitor Command Programs are special programs that typically consist of a series of monitor command keywords. These are used to automate system startup and other frequently executed keyword sequences. Monitor Command Programs can be started from the V+ Monitor Window or from the teach pendant. They execute in a top-down manner from the first line to the last.

**Additional Information:** Program command keywords can be used in Monitor Command Programs. Refer to Monitor Command Program Contents on page 61 for more information.

With the autostart feature, a Monitor Command Program can be started automatically when the system is turned ON. Monitor Command Programs are most commonly used in V+ based applications, as described in the following cases.

- At system startup, the controller autostart switch will launch the Monitor Command Program "auto" located in D:\AUTO.V2. Refer to Autostarting a Monitor Command Program on page 62 for more information.
- Monitor Command Programs can be used to save a V+ module and application variables. A Monitor Command Program will define the programs in a module and then perform a backup / save operation.

When the Sysmac Studio or ACE software is used, this is not necessary because the V+ Module object saves the programs in the project. The same applies for V+ application variables contained in the V+ Global Variable Collection object.

**Additional Information:** Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595)* / *Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for more information about V+ modules and V+ global variables.

## 6.2 Monitor Command Program Names

Monitor Command Program names can have up to 15 characters, must begin with a character, and can contain letters, numbers, periods, and underscore characters.

Monitor Command Programs are stored as objects in the V+ disk file. The disk file name, not the command program name, must conform to V+ disk file name conventions. Refer to File and Folder Name Requirements on page 13 for more information.

## 6.3 Monitor Command Program Contents

Normally, a Monitor Command Program contains only monitor command keywords. Program command keywords can be included by using the DO monitor command keyword if necessary. To include a program command keyword in a Monitor Command Program, use the following statement syntax.

MC DO keyword parameter, ...

Make the following considerations when creating a Monitor Command Program.

- Monitor Command Programs can contain all monitor command keywords except the DONE, TEACH and ZERO keywords.
- Control keywords (for example, GOTO and IF) are ignored in a Monitor Command Program.
- Every non-blank line of a Monitor Command Program must contain either a monitor command keyword, or a comment.

**IMPORTANT:** Ensure Monitor Command Program can execute correctly without operator response or intervention.

## 6.4 Executing Monitor Command Programs

Monitor Command Programs can be executed with the methods described below.

### Executing Monitor Command Programs from the Monitor Window

Use the COMMANDS monitor command keyword to execute a Monitor Command Program from the Monitor Window.

Use the following statement to execute a Monitor Command Program named "my.cmd.program" with the file name "my\_cmd.pg".

```
LOAD my_cmd.pg
COMMANDS my.cmd.program
```

**NOTE:** The EXECUTE keyword expects programs that contain program command keywords and not monitor command keywords. If you use the EXECUTE keyword to execute a program with lines that begin with "MC", the program will abort and a message indicating that you cannot mix program command keywords and monitor command keywords will be returned.

### Autostarting a Monitor Command Program

A Monitor Command Program will be loaded and started automatically when the controller is turned ON if the following conditions have been met.

- The autostart option must be enabled in the controller configuration settings.
- A disk file with the name "AUTO.V2" must reside on the default disk.
- The AUTO.V2 disk file must contain a program named "auto" and this program must be a Monitor Command Program. Other programs, and data, can be stored in the AUTO.V2 disk file.

If the conditions above are met, the following functions occur when the controller is powered ON.

- The V+ control system is loaded.
- The default disk specification is set.
- The following statements are issued automatically.

```
LOAD auto.v2
COMMANDS auto
```

V+ does not wait for high power to be turned ON or for any other event or condition. The autostart Monitor Command Program must explicitly invoke any such functions or they must be performed by programs invoked by the Monitor Command Program. For example, if you want to require that the operator press the program start button on the controller when the Monitor Command Program is processed, you must include a WAIT.START statement in the Monitor Command Program.

**Additional Information:** A Monitor Command Program that is autostarted should load minimal information before executing a V+ program that can process more complicated logic and error handling.

## Monitor Command Program Processing

When a Monitor Command Program is executed, it will process all of the statements until one of the following events occurs.

- The end of the Monitor Command Program is reached.
- A CYCLE.END statement is processed in the Monitor Command Program and the referenced program task is executing. This will suspend processing of the Monitor Command Program until the referenced program task finishes executing.

**NOTE:** A Monitor Command Program will not be suspended when an EXECUTE statement is processed. The Monitor Command Program will initiate execution of the specified application program and then immediately continue with the next statement.

- CTRL+C is pressed. This will terminate processing of the Monitor Command Program. The statement being processed or a program invoked by the Monitor Command Program will continue to completion.
- Another Monitor Command Program is invoked from within the active command program. Unlike starting an executable program, control will not return to the first Monitor Command Program when the second Monitor Command Program completes.
- An error condition results when a statement is processing in the Monitor Command Program.



# Chapter 7: V+ programs

This section describes the creation and use of V+ programs.

## 7.1 V+ program Format

V+ programs are used to generate robot motion and contain logic and functionality for other operations such as I/O control, data management, and various run-time tasks. V+ programs are created with program command and function keywords.

This section details the format that V+ programs must follow. The format of the individual lines is described, followed by the overall organization of programs. This information applies to all V+ programs regardless of their intended use.

### Program Lines

Each line or step of a program is interpreted by the V+ system as a program statement. The general format of an V+ program step has the syntax shown below.

```
step_number step_label operation ;Comment
```

The items in the syntax above are optional and described below.

Table 7-1. Program Line Item Description

Item	Description
step_number	Each step within a program is automatically assigned a step number. Steps are numbered consecutively and the numbers are automatically adjusted whenever steps are inserted or deleted. Step numbers are automatically present in V+ programs. These are used to reference V+ program errors and other program references. Step numbers are also referenced as line numbers.
step_label	Because step numbers change as a program is created, they are not useful for identifying steps for program-controlled branching. Therefore, program steps can contain a step label. A step label is a programmer-specified integer (0 to 65535) that is placed at the start of a program line to be referenced elsewhere in the program. These are typically used with GOTO statements.
operation	The operation portion of each step must be a valid statement and may contain parameters and additional keywords. Refer to the <i>V+ Keyword Reference Manual (Cat. No. I672)</i> for more information.
comment	The semicolon character (;) is used to indicate that the remainder of a program line is comment information to be ignored by V+. When all the elements of a program step are omitted, a blank line results.  When only the comment element of a program step is present, the step is called a comment line. Use comments to describe and explain the intent of the sections of the programs. These program notations will make it easier to modify and debug programs.

Item	Description
	<p><b>NOTE:</b> Blank program lines are acceptable in V+ programs. Blank lines are often useful to separate program steps to make a program easier to read.</p>

### Blank Spaces and Case

When program lines are entered, extra blank spaces can be used between any elements in the line. The V+ editor adds or deletes spaces in program lines to make them conform with the standard spacing. The editors also automatically format the lines to uppercase for all keywords and lowercase for all user-defined names.

**Additional Information:** When program line is completed by entering a carriage return, moving off a line, or exiting the editor, the editor checks the syntax of the line. If the line cannot be executed, it is displayed in red within the Sysmac Studio or ACE software V+ editor. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for more information.

### Program Structure

The first step of every V+ program must be a .PROGRAM keyword. This keyword names the program, defines any arguments it receives or returns, and has the following format.

```
.PROGRAM program_name (parameter_list) ;Comment
```

The program\_name item is required but the parameter list and comment are optional.

After the .PROGRAM statement on the first line, there are only two restrictions that apply to the order of other statements that follow.

1. AUTO, EXTERNAL, LOCAL, or GLOBAL keyword statements must precede any other normal keyword statements at the top of the program. Only comment lines, blank lines, and other AUTO, LOCAL, or GLOBAL keywords are permitted between the .PROGRAM step and an AUTO, LOCAL, or GLOBAL keyword statement.
2. The end of a program is marked by a line beginning with .END. The V+ editors automatically add this line at the end of a program.

**NOTE:** The .PROGRAM and .END statements are automatically inserted by the Sysmac Studio or ACE software program editor. If a different program editor is used, you must manually enter these two lines. In general, any editor that produces unformatted ASCII files can be used for programming.

## 7.2 V+ Program Execution

The V+ system provides for simultaneous execution of up to 64 different programs. Execution of each program is administered as a separate program task by the system. The way program execution is started depends upon the program task to be used and the type of program to be executed.

The following sections describe program execution in detail.

## Program Tasks

V+ programs are executed on one of the 64 available Tasks. Task number 0 has the highest priority and is typically used for the primary application program. Task 0 is normally used to execute the primary robot control program.

When execution of Task #0 begins, the Task automatically selects robot 1 and attaches the robot as soon as a motion related keyword is encountered. Execution of Task 0 is typically started by issuing the EXECUTE monitor command keyword. The ABORT monitor command or program command keyword stops Task #0 after the current robot motion completes. The CYCLE.END monitor command or program command keyword can be used to stop the program at the end of its current execution cycle.

For normal execution of a robot control program, the system switch keyword DRY.RUN must be disabled and the robot must be attached by the robot control program. Then, any robot related error will stop execution of the program unless an error-recovery program has been established. Refer to REACTE in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

If program execution stops because of an error, a PAUSE statement, an ABORT statement, the monitor command keywords PROCEED or RETRY can be used to resume execution.

**Additional Information:** While execution is stopped, the DO monitor command keyword can be used to execute a single program statement manually as though it were the next statement in the program that is stopped.

Execution of program Tasks other than 0 is generally the same as for Task 0. The following differences apply.

- The Task number must be explicitly included in all the statements that affect program execution, including EXECUTE, ABORT, PROCEED and RETRY.
- If the program controls the robot, it must explicitly SELECT and ATTACH the robot before executing any statements that control the robot.

### Exclusive Control of a Robot

Use the following considerations to understand the restrictions for exclusive control of a robot with Tasks.

- Whenever a robot is attached by an active Task, no other Task can attach that robot or execute statements that affect it, except for the REACTI and BRAKE keywords. Refer to V+ Program Interrupts on page 74 for more information.
- When the robot control Task stops execution for any reason, the robot is detached until the Task resumes at which time the Task automatically attempts to reattach the robot. If another Task has attached the robot in the meantime, the first Task cannot be resumed.
- Task 0 always attempts to attach robot 1 when program execution begins. No other Tasks can successfully attach any robot unless explicit SELECT and ATTACH statements are executed.
- Since Task 0 attempts to attach robot 1, that Task cannot be executed after another Task has attached that robot. If another Task needs to control the robot and you want to execute task 0, you must follow this sequence of events:
  1. Start Task 0.
  2. Make Task 0 DETACH robot 1.
  3. Start the Task that will control robot 1. The program executing as Task 0 can

start up another Task.

4. Make that other Task ATTACH the robot.

**NOTE:** Robots are attached even in DRY.RUN mode. In this case, motion statements issued by the Task are ignored and no other Task can access the robot.

### **EtherCAT Communication Errors**

EtherCAT communication errors affect Task and robot behavior as described below.

#### **EtherCAT Network Communication Error**

An EtherCAT network communication error will cause the following system behavior.

- All running tasks are stopped and it is not possible to execute any Task that requires communication with the host system. All robots will become detached.
- All robots in the network will stop with maximum deceleration and high power is disabled.

#### **EtherCAT Node Communication Error**

If network communication stops for individual nodes, the following events occur.

- Running tasks associated with the problematic node are stopped, associated robots will become detached, and it is not possible to execute any Task that requires communication with the host system.
- The robot(s) nodes that have lost communications will stop running Tasks that are attached.

### **V+ program Processing**

Program statements are normally executed sequentially from the beginning of a program to its end. This sequential flow may be changed when a GOTO or IF...GOTO statement or a control structure is encountered.

A CALL statement causes another program to be executed, but it does not change the sequential flow through the calling program because execution of the calling program resumes where it left off when a RETURN or RETURNE statement is executed by the called program.

A WAIT statement suspends execution of the current program until a condition is satisfied. The WAIT.EVENT statement suspends execution of the current program until a specified event occurs or until a specified time elapses.

The PAUSE and HALT statements both terminate execution of the current program. After a PAUSE statement, program execution can be resumed with a PROCEED monitor command keyword. The HALT program command causes a break and then terminates execution of the application program regardless of any program loops remaining to be completed. After termination by a HALT command, program execution cannot be resumed with a PROCEED or RETRY command.

A STOP statement may or may not terminate program execution. If there are more program execution cycles to perform, the STOP statement causes the main program to be restarted at its first step even if the STOP statement occurs in a subroutine. If no execution loops remain, STOP terminates the current program. Refer to V+ Program Interrupts on page 74 for more information.

## 7.3 Program Control

This section introduces the structures available in V+ and other methods used to control program execution. V+ supports looping branching structures common to many high-level languages as well as some keywords specific to V+.

### Unconditional Branching

The following keywords are described in this section for use with unconditional branching.

- GOTO
- CALL
- CALLS

#### *GOTO Unconditional Branching*

The GOTO keyword causes program execution to branch immediately to a label somewhere else in the program. The syntax for a GOTO statement is shown below.

```
GOTO label
```

The label item above is an integer entered at the beginning of a line of program code. Label is not the same as the program step numbers. Step numbers are assigned by the system and labels are entered by the programmer as the opening to a line of code.

The following example will go to label 100 and execute the TYPE statement "The GOTO keyword got me here."

```
GOTO 100
100 TYPE "The GOTO Keyword got me here."
```

A GOTO statement can branch to a label that is before or after the GOTO keyword.

**IMPORTANT:** Use GOTO keywords with care. GOTO statements can make program logic difficult to follow and debug, especially in a long or complicated program with many subroutine calls. A common use of GOTO is as an exit routine or exit on error statement.

#### *CALL Unconditional Branching*

The CALL keyword causes program execution to be suspended and execution of a new program to begin. When the new program has completed execution, execution of the original program resumes at the statement after the CALL keyword.

The simplified syntax for a CALL statement is shown below.

```
CALL program_1(arg_list)
```

The "program\_1" item above is the name of the program to be called. The program name must be specified exactly and the program being called must be resident in system memory.

The arg\_list item above is the list of arguments being passed to the subroutine. These arguments can be passed either by value or by reference and must agree with the arguments expected by the program being called. Refer to Exchanging Information Using the Program Argument List on page 81 for more information.

The following example suspends execution of the calling program, passes the arguments "locx", "locy", and "length" to program "check\_data", executes "check\_data", and after "check\_data" has completed execution, resumes execution of the calling program at the statement that follows the CALL operation.

```
CALL check_data(locx, locy, length)
```

### **CALLS Unconditional Branching**

The CALLS keyword is identical to the CALL keyword except for the specification of program. For a CALLS keyword, the string parameter is a string value, variable, or expression. This allows you to call different subroutines under different conditions using the same line of code. These different subroutines must have the same arg\_list.

The simplified syntax for a CALLS statement is shown below.

```
CALLS string(arg_list)
```

The following example suspends execution of the calling program, passes the variables "length" and "width" to the program specified by array index "program\_select" from the array "\$program\_list", executes the specified program, and resume execution of the calling program at the next step.

```
$program_name = $program_list[program_select]
CALLS $program_name(length, width)
```

### **Conditional Branching**

The following sections discuss program control structures whose execution depend on an expression or variable with a boolean value (a variable that is either true or false or an expression that resolves to true or false). An expression can take into account any number of variables or digital input signals as long as the final resolution of the expression is a boolean value.

Conditional branching keywords allow you to execute blocks of code based on the current values of program variables or expressions. V+ has three conditional branch keywords.

1. IF...GOTO
2. IF...THEN...ELSE
3. CASE value OF

Any number (real or integer) can satisfy this requirement. Zero is considered false and any nonzero number is considered true. There are four system constants: TRUE and ON that resolve to -1 and FALSE and OFF that resolve to 0.

#### **IF...GOTO Conditional Branching**

IF...GOTO behaves similarly to GOTO, but a condition is attached to the branch.

The following example will branch to label 100 if the "logical\_expression" is true.

```
IF logical_expression GOTO 100
```

#### **IF...THEN...ELSE Conditional Branching**

IF...THEN...ELSE will conditionally execute one of two different actions depending on the result of a logical expression.

This keyword has two forms shown below.

```
IF expression THEN
    statement
END
```

```
IF expression THEN
    statement
```

```

ELSE
    statement
END

```

In the following example, if "num\_parts" is greater than 75, CALL check\_num(num\_parts) is executed. If "num\_parts" is not greater than 75, program execution continues with the statement following END..

```

IF num_parts > 75 THEN
    CALL check_num(num_parts)
END

```

In the following example, if signal 1033 is ON and "need\_part" is true, the program executes the MOVE and DEPART statements. When the IF statement is false, the TYPE statement is executed and program execution continues with the statement following END.

```

IF SIG(1033) AND need_part THEN
    MOVE loc1
    DEPART 50
ELSE
    TYPE "Part not picked up"
END

```

### **CASE value OF Conditional Branching**

The CASE structure will allow a program to take one of many different actions based on the value of a variable. The variable used must be a real or an integer.

The form of the CASE structure is shown below. If the "list\_of\_values" listing of real values (separated by commas) equals "target", the statements following that value statement is executed. Statements after the ANY step will be executed when "target" is not in any "list\_of\_values" listing.

```

CASE target OF
    VALUE list_of_values
        statements
    VALUE list_of_values
        statements
ANY
    statements
END

```

### **Looping Structures**

In many cases, you will want the program to execute statements more than once. V+ has three looping structures that allow you to execute statements a variable number of times as described below.

#### **FOR Looping Structure**

A FOR keyword creates an execution loop that will execute a given block of statements a specified number of times.

The basic form of a FOR loop is shown below. This loop will start by setting "index" to the value of "start\_val". The loop checks that the value of "index" is within the range of "start\_val" to "end\_val". If this is true, it executes any statements within the FOR loop. After a loop is executed, the value of "index" will be incremented by the value of "incr". This continues until the value of "index" is no longer in the range of "start\_val" to "end\_val".

```
FOR index = start_val TO end_val STEP incr
    statement
    ...
    statement
END
```

The following example will output even elements of array "\$names" up to index 32.

```
FOR i = 2 TO 32 STEP 2
    TYPE $names[i]
END
```

The following example will output the values of the 2-dimensional array "values" in column and row form (10 rows by 10 columns).

```
FOR i = 1 TO 10
    FOR j = 1 to 10
        TYPE values[i,j], /S
    END
    TYPE " ", /C1
END
```

The following example will count backward from 10 to 1 by entering a negative value for the step increment.

```
FOR i = 10 TO 1 STEP -1
    TYPE i
END
```

Changing the value of index inside a FOR loop will cause the loop to behave improperly. To avoid problems with the index, make the index variable an auto variable and do not change the index from inside the FOR loop. Changes to the starting and ending variables do not affect the FOR loop once it is executing.

**NOTE:** FOR loops can consume unnecessary processing time by executing more frequently than necessary. To avoid this, insert a WAIT or WAIT.EVENT statement within the FOR loop.

### **DO...UNTIL Looping Structure**

DO...UNTIL is a looping structure that executes a group of statements an indeterminate number of times. Termination of the loop occurs when the boolean expression or variable that controls the loop becomes true. The boolean value is evaluated after each execution of group of statements and if the expression becomes true, the loop is not executed again. Since the expression is not evaluated until after the statements have been executed, the code block will always execute at least once.

**NOTE:** The expression is any boolean statement and must eventually become true or the loop executes indefinitely.

The form for this looping structure is shown below.

```
DO
    Statement
    ...
    Statement
UNTIL expression
```

The following example will output the numbers 1 to 100 to the Monitor Window.

```

x = 1
DO
    TYPE x
    x = x + 1
UNTIL x > 100

```

The following example will echo up to 15 characters to the Monitor Window. The loop will stop when 15 characters or the "#" character has been entered.

```

x = 1
DO
    PROMPT "Enter a character: ", $ans
    TYPE $ans
    x = x + 1
UNTIL (x > 15) or ($ans == "#")

```

### WHILE...DO Looping Structure

WHILE...DO is a looping structure similar to DO...UNTIL except the boolean expression is evaluated at the beginning of the loop instead of at the end. This means that if the condition indicated by the expression is true when the WHILE...DO statement is encountered, the code within the loop will be executed. WHILE...DO loops are susceptible to infinite looping just as DO...UNTIL loops are. The expression controlling the loop must eventually evaluate to false for the loop to terminate.

The form of the WHILE...DO looping structure is shown below.

```

WHILE expression DO
    statement(s)
END

```

The following example shows a WHILE...DO loop being used to validate input. Since the boolean expression is tested before the loop is executed, the code within the loop will be executed only when the operator inputs an unacceptable value.

```

PROMPT "Enter a number in the range 32 to 64.", ans
WHILE (ans < 32) OR (ans > 64) DO
    PROMPT "Number must be in the range 32-64.", ans
END

```

In the above code, an operator could enter a nonnumeric value in which case the program execution would stop. A more robust strategy would be to use a string variable in the PROMPT statement and then use the \$DECODE and VAL keywords to evaluate the input.

In the following example, if digital signal 1033 is ON when the WHILE statement is reached, the loop does not execute and the program continues after the END keyword. If digital signal 1033 is OFF, the loop executes continually until the signal comes ON.

```

WHILE NOT SIG(1033) DO
    ...
END

```

### Loading Programs and Variables to System Memory

V+ cannot use programs and information stored in disk files until they are loaded into system memory (RAM). A file can be loaded with the Sysmac Studio or ACE software, or with the LOAD monitor command keyword. Refer to the *Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual (Cat. No. W595) / Automation Control Environment (ACE) Version 4 User's Manual (Cat. No. I633)* for more information.

The LOAD keyword does not actually start a program executing. It simply places a copy of the disk file contents into system RAM so additional commands can start the program, modify the program, or modify the values of the program variables.

The LOAD keyword has the following syntax.

```
LOAD my_file
```

This statement places the contents of "my\_file" into RAM.

**NOTE:** A single disk file can include multiple programs and variable information. Refer to V+ File Management on page 13 for more information.

**Additional Information:** To view all programs that have been loaded into memory, use the DIRECTORY keyword.

To view all files on the SD card, use the FDIRECTORY keyword.

### Removing Programs and Variables from System Memory

When a program completes, it is not automatically removed from system memory (RAM). If an \*Out of memory\* error message appears while loading a disk file, the only way to complete the load operation is to delete objects from system memory. To make system memory available for use by other programs and data, objects in memory must be specifically removed with a DELETE keyword.

**NOTE:** The LOAD keyword will not overwrite programs that reside in system memory. Therefore, if you want to load new programs with identical names, you must delete the programs currently in system memory.

Objects do not have to be removed from system memory before different disk files are loaded and other programs are executed, unless you are loading a program with the same name as one already in system memory.

The DELETE keyword does not delete a program that is being executed or is present on the stack for any task. The DELETE keyword does not affect the disk files from which the objects were loaded. After objects have been removed from system memory, they can be reloaded using the LOAD command.

## 7.4 V+ Program Interrupts

V+ provides several ways of suspending or terminating program execution.

- A program can be put on hold until a specific condition becomes TRUE using the WAIT keyword. This may not be the most efficient use of processor time. Refer to WAIT.EVENT Program Interrupts on page 75 for more information.
- A program can be put on hold for a specified time period or until an event is generated in another task by the WAIT.EVENT keyword.
- A program can be interrupted based on a state transition of a digital signal with the REACT and REACTI keywords. Refer to REACT and REACTI Program Interrupts on page 75 for more information.
- Program errors can be intercepted and handled with a REACTE keyword.
- Program execution can be terminated with the HALT, STOP, and PAUSE keywords.

These keywords interrupt the program in which they are contained. Any programs running as other tasks are not affected. Robot motion can be controlled with the BRAKE, BREAK, and

DELAY keywords. The ABORT and PROCEED monitor command keywords can also be used to suspend and proceed programs.

Use the information below to understand different program interrupt methods.

### WAIT Program Interrupts

The WAIT keyword suspends program execution until a condition(s) becomes true.

The syntax for the WAIT keyword is shown below.

```
WAIT condition
```

The following example suspends execution until digital input signal 1032 is ON and 1028 is OFF.

```
WAIT SIG(1032, -1028)
```

The following example suspends program execution until timer 1 returns a value greater than 10.

```
WAIT TIMER(1) > 10
```

### WAIT.EVENT Program Interrupts

The WAIT.EVENT keyword suspends program execution until a specified event has occurred or until a specified amount of time has elapsed.

The syntax for the WAIT.EVENT keyword is shown below.

```
WAIT.EVENT mask, timeout
```

**Additional Information:** This keyword is more efficient than waiting for a timer (WAIT TIMER(1) > 10) because the task does not have to loop continually to check the timer value.

The following example suspends program execution for 3.7 seconds.

```
WAIT.EVENT, 3.7
```

The following example suspends program execution until another task issues a SET.EVENT statement to the waiting task.

```
WAIT.EVENT
```

**NOTE:** If the SET.EVENT operation does not occur, the task waits indefinitely.

### REACT and REACTI Program Interrupts

When a REACT or REACTI keyword is encountered, the program begins monitoring a digital input signal specified in the REACT statement. This signal is monitored in the background with program execution continuing normally until the specified signal transitions. When a transition is detected, the program suspends execution at the currently executing step and a call to a specified subroutine is made.

Additionally, REACTI issues a BRAKE operation to immediately stop the current robot motion. Both keywords specify a subroutine to be run when the digital transition is detected. After the specified subroutine has completed, program execution resumes at the step executing when the digital transition was detected.

Digital signals can be used with the REACT keyword. The signal monitoring initiated by REACT / REACTI is in effect until another REACT / REACTI or IGNORE keyword is

encountered. If the specified signal transition is not detected before an IGNORE or second REACT / REACTI keyword is encountered, the REACT / REACTI keyword has no effect on program execution. The following signals can be used.

Signal Type	Range	Additional Informaiton
Digital Input	1001 to 1999	Signal numbers are available for use only if they also physically exist and are configured.
Software	2001 to 2999	
External	4001 to 4999	Signal numbers are available if they are mapped.

**Additional Information:** The LOCK keyword can be used to control execution of a program after a REACT or REACTI subroutine has completed.

The syntax for the REACT and REACTI keywords are shown below.

```
REACT signal_number, program, priority
REACTI signal_number, program, priority
```

The following example implements a REACT routine where the signal 1001 is evaluated for a change in state from ON to OFF. If a change is detected, a call to the subroutine "alarm" occurs with a priority setting of 10 and execution resumes at the next step. The main program has a default priority of 0.

If signal 1001 transitions during execution of the MOVE c statement MOVE c, MOVE d, and LOCK 0 complete since the program had been given a higher priority than REACT. Subroutine "alarm" is called and execution resumes at the MOVE e statement. The LOCK statement sets the program reaction lock-out priority to a value of 20.

```
REACT -1001, alarm, 10

MOVE a
MOVE b
LOCK 20
MOVE c
MOVE d
LOCK 0
MOVE e

IGNORE -1001
```

## REACTE Program Interrupts

REACTE enables a reaction program that is run whenever a system error that causes program execution to terminate is encountered. This includes all robot errors, hardware errors, and most system errors (it does not include I/O errors). Unlike REACT and REACTI, REACTE cannot be deferred based on priority considerations.

The syntax for the REACTE keyword is shown below.

```
REACTE program_name
```

The following example enables monitoring of system errors and executes the program "trouble" whenever a system error is generated.

```
REACTE trouble
```

## HALT, STOP, and PAUSE Program Interrupts

When a HALT keyword is encountered, program execution is terminated and any open disk units are detached and closed with the FCLOSE operation. PROCEED or RETRY will not resume execution.

When a STOP keyword is encountered, execution of the current program cycle is terminated and the next execution cycle resumes at the first step of the program. If the STOP keyword is encountered on the last execution cycle, program execution is terminated and any open disk units are detached and closed with the FCLOSE operation. PROCEED or RETRY will not resume execution.

When a PAUSE keyword is encountered, execution is suspended. After a PAUSE, the system prompt appears and monitor command keywords can be executed. This allows you to verify the values of program variables and set system parameters. This is useful during program debugging. The monitor command keyword PROCEED resumes execution of a program interrupted with the PAUSE keyword.

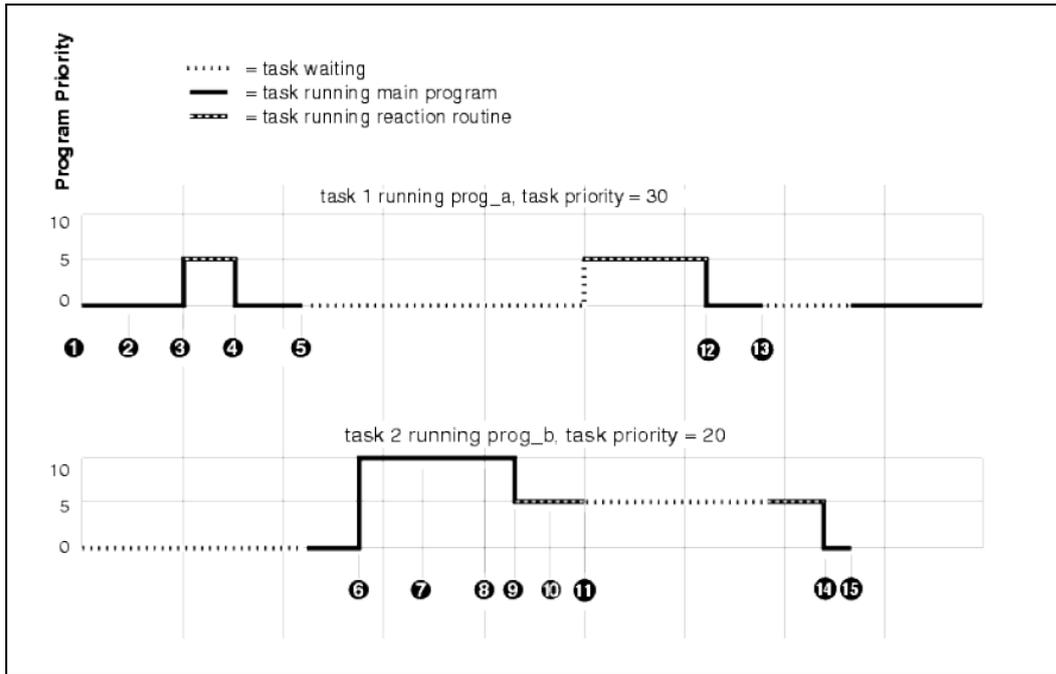
## Additional Program Interrupt Keywords

You can specify a parameter in the I/O keywords ATTACH, READ, GETC, and WRITE that causes the program to suspend until the I/O request has been successfully completed. Refer to the *V+ Keyword Reference Manual (Cat. No. 1672)* for more information.

## Program Interrupt Example

The following figure shows how the task and program priority scheme works. It also shows how the asynchronous and program interrupt statements work within the priority scheme. The example makes the following assumptions.

- Task 1 runs in all time slices at priority 30.
- Task 2 runs in all time slices at priority 20.
- All system tasks are ignored.
- All system interrupts are ignored.



The illustration shows the timing of executing programs. A solid line indicates a program is running and a dotted line indicates a program is waiting. The Y axis shows the program priority. The X axis is divided into 16-millisecond major cycles.

The example shows two tasks executing concurrently with REACT routines enabled for each task. The LOCK keyword and triggering of the REACT routines change the program priority. The sequence of events for the example is described below.

1. Task 1 is running program "prog\_a" at program priority 0. A reaction program based on signal 1003 is enabled at priority 5.
2. Signal 1003 is asserted externally. The signal transition is not detected until the next major cycle.
3. The signal 1003 transition is detected. The task 1 reaction program begins execution, interrupting "prog\_a".
4. The task 1 reaction program reenables itself and completes by issuing a RETURN operation. "prog\_a" resumes execution in task 1.
5. Task 1 "prog\_a" issues a CLEAR.EVENT operation followed by a WAIT.EVENT operation to wait for its event flag to be set. Task 1 is suspended and task 2 resumes execution of "prog\_b". Task 2 has a reaction program based on signal 1010 enabled at priority 5.
6. Task 2 "prog\_b" issues a LOCK 10 statement to raise its program priority to level 10.
7. Signal 1010 is asserted externally. The signal transition is not detected until the next major cycle.
8. The signal 1010 transition is detected and the task 2 reaction is triggered. However, since the reaction is at level 5 and the current program priority is 10, the reaction execution is deferred.
9. Task 2 "prog\_b" issues a LOCK 0 statement to lower its program priority to level 0. Since a level 5 reaction program is pending, it begins execution immediately and sets the program priority to 5.
10. Signal 1003 is asserted externally. The signal transition is not detected until the next major cycle.

11. The signal 1003 transition is detected which triggers the task 1 reaction routine and also sets the task 1 event flag. Since task 1 has a higher priority (30) than task 2 (20), task 1 begins executing its REACT routine and task 2 is suspended.
12. The task 1 reaction routine completes by issuing a RETURN keyword. Control returns to "prog\_a" in task 1.
13. Task 1 "prog\_a" issues a CLEAR.EVENT operation followed by a WAIT.EVENT operation to wait for its event flag to be set. Task 1 is suspended and task 2 resumes execution of its REACT routine. The task 2 REACT routine completes by issuing a RETURN keyword. Control returns to "prog\_b" in task 2.
14. Task 2 "prog\_b" issues a SET.EVENT 1 statement, setting the event flag for task 1. Task 2 now issues a RELEASE statement to yield control of the CPU.
15. Since the task 1 event flag is now set and its priority is higher than task 2, task 1 resumes execution and task 2 is suspended.

## 7.5 V+ program Stacks

When program calls are made, V+ uses an internal storage area called a stack to save information required by the executing program. The following information is saved.

- The name and step number of the calling program.
- Data necessary to access subroutine arguments.
- The values of any automatic variables specified in the called program.

The V+ system allows you to explicitly allocate storage to the stack for each program task. The amount of stack space can be adjusted for a particular application to optimize the use of system memory. Stacks can be made arbitrarily large, limited only by the amount of memory available on the system.

### V+ Program Stack Requirements

When a V+ program is executed in a given Task, each program stack is allocated 32 kilobytes of memory. This value can be adjusted once the desired stack requirements are determined by using the STACK monitor command keyword. This is typically accomplished in a start-up Monitor Command Program.

### Determining Stack Requirements

One method of determining the stack requirements of a program task is simply to execute its program. If the program runs out of stack space, it stops with the error message \*Not enough program stack space\*. After increasing the stack size, issue the RETRY keyword to continue program execution to avoid restarting the program from the beginning.

**Additional Information:** The STATUS keyword will return the amount of stack space a failed Task requested.

Alternatively, you can set a large stack size before running a program. After the program has been run and all the execution paths have been followed, use the STATUS monitor command keyword to examine the stack statistics for the program Task. The stack MAX value shows how much stack space the program Task needs in order to execute. The stack size can then be adjusted to accommodate the Task stack space requirements (it is recommended to add 10 to 20% extra as a safety factor).

If it is impossible to invoke all the possible execution paths, the theoretical stack limits can be calculated from the figures provided in the following table. You can calculate the worst-case

stack size by summing the overhead for all the program calls that can be active at one time. Divide the total by 1024 to get the size in kilobytes. Use this number in the STACK monitor command statement to set the size.

Table 7-2. Stack Space Requirements

Item	Stack Space Requirements (Bytes)
The subroutine call.	20
Each subroutine argument (plus one of the following items below).	32
Each real subroutine argument or automatic variable.	4
Each double-precision real subroutine argument or automatic variable.	8
Each transformation subroutine argument or automatic variable.	48
Each precision-point subroutine argument or automatic variable.	Requires four bytes for each joint of the robot. For multiple robot systems, use the robot with the most joints.
Each belt variable argument or automatic variable.	84
Each string variable argument or automatic variable.	132

**NOTE:** If any subroutine argument or automatic variable is an array, the size shown must be multiplied by the size of the array (array indexes start at 0).

If a subroutine argument is always called by reference, the value can be omitted for that argument.

## 7.6 Exchanging Information Between V+ programs

There are three methods of exchanging information between V+ programs.

1. Global variables
2. Soft-signals
3. Program argument list

### Exchanging Information Using Global Variables

When using global variables to exchange information between V+ programs, simply use the same variable names in the different programs.

Unless used carefully, this method can make program execution unpredictable and difficult to debug. It also makes it difficult to write generalized subroutines because the variable names in the main program and subroutine must always be the same.

### Exchanging Information Using Soft-signals

Soft-signals are digital software switches whose state can be read and set by all tasks and programs. Refer to Soft Signals on page 27 for more information.

## Exchanging Information Using the Program Argument List

Exchanging information through the program argument provides better control over changes made to variables. It also eliminates the requirement that the variable names in the calling program be the same as the names in the subroutine.

The following sections describe exchanging data through the program argument list.

### Argument Passing

There are two important considerations when passing an argument list from a calling program to a subroutine.

1. Ensure the calling program passes arguments in the way the subroutine expects to receive them (mapping).
2. Determine how you want the subroutine to be able to alter the variables (passing by value or reference).

### Mapping the Argument List

An argument list is a list of variables or values separated by commas. The argument list passed to a calling program must match the subroutine's argument list in number of arguments and data type of each argument or an error will occur.

**Additional Information:** If the calling program omits an argument, either by leaving a blank in the argument list (e.g., `arg_1, , arg_3`) or by omitting arguments at the end of a list (e.g., `arg_1, arg_2`), the argument are passed as undefined. The subroutine receiving the argument list can test for this value using the `DEFINED` function keyword and take appropriate action.

The variable names do not have to match. When a calling program passes an argument list to a subroutine, the subroutine does not evaluate the variable names in the list. It only evaluates the position of the arguments in the list. The argument list in the `CALL` statement is mapped to the argument list of the subroutine. It is this mapping feature that allows you to write generalized subroutines that can be called by any number of different programs, regardless of the actual values or variable names the calling program uses.

The following diagram shows the mapping of an argument list in a `CALL` statement to the argument list in a subroutine. The arrows indicate that each item in the list must match in position and data type, but not necessarily in name. The `CALL` statement argument list can include values and expressions as well as variable names.

When the main program reaches a `CALL` statement as shown below, the subroutine "a\_routine" is called and the argument list is passed as indicated.

Statement in the calling program:

```
CALL a_routine(loc_var_a, real_var_a, 43.654, $string_var_a)
```

Called program line 1:

```
.PROGRAM a_routine(any_loc, any_real_x, any_real_y, $any_string)
```

Statement in the calling program:

```
CALL a_routine(loc_var_a:NULL, (real_var_a), real_var_b, $string_var_a+"")
```

Called program line 1:

```
.PROGRAM a_routine(any_loc, any_real_x, any_real_y, $any_string)
```

**Additional Information:** Refer to the CALL keyword in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information on passing arrays.

### Argument Passing by Value or Reference

Variables can be changed by a subroutine and the changed value can be passed back to the calling program. Argument passing by value occurs when a copy of the value is passed to the called subroutine and the change to the corresponding value in the subroutine is not passed back to the calling routine. This can be forced with the following methods described below. These methods may be useful if a subroutine is being called multiple times but the subroutine's modifier is only to be passed to the main routine during one instance.

If a calling program passes a variable to a subroutine but the subroutine cannot pass the variable back in an altered form, the variable is considered to be passed by value. Variables that require modification by a subroutine should be passed by reference.

To force the value of a real variable "real\_1" to not be passed back to the calling routine "program", use the statement below that encloses the variable in parentheses.

```
CALL program((real_1))
```

To force the value of a string variable "string\_1" to not be passed back to the calling routine "program", use the statement below that adds an empty string to the variable.

```
CALL program($string_1+"")
```

To force the value of a transformation variable "location\_1" to not be passed back to the calling routine "program", use the statement below that compounds the transformation by null.

```
CALL program(location_1:NULL)
```

If a calling program passes a variable to a subroutine and the subroutine can change the variable and pass the changed variable back to the calling program, the variable is considered passed by reference.

Changes made by the subroutine are reflected in the state of the variables in the calling program. Any argument that is to be changed by a subroutine and passed back to the calling routine must be specified as a variable (not an expression or value).

When calling a subroutine, it is acceptable to pass some arguments by value and others by reference. When a variable is passed by value, a copy of the variable, rather than the actual

variable, is passed to the subroutine. The subroutine can make changes to the variable, but the changes are not returned to the calling program (the variable in the calling program has the same value it had when the subroutine was called).

The following diagram shows how to pass the different types of variables by value. Real numbers and integers are surrounded by parentheses, :NULL is appended to location variables, and "+" is appended to string variables.

In the following figure, "real\_var\_b" is still being passed by reference and any changes made in the subroutine will be reflected in the calling program. The subroutine cannot change any of the other variables. It can make changes only to the copies of those variables that have been passed to it. It is considered poor programming practice for a subroutine to change any arguments except those that are being passed back to the calling routine. If an input argument must be changed, it is recommended you make an automatic copy of the argument and manipulate that copy.

Statement in the calling program:

```
CALL a_routing(loc_var_a, real_var_a, 43.654, $string_var_a)
```

Called program line 1:

```
.PROGRAM a_routine(any_loc, any_real_x, any_real_y, $any_string)
```

Statement in the calling program:

```
CALL a_routing(loc_var_a:NULL, (real_var_a), real_var_b, $string_var_a+"")
```

Called program line 1:

```
.PROGRAM a_routine(any_loc, any_real_x, any_real_y, $any_string)
```

Values as well as variables can be passed by a CALL statement. The following example is an acceptable call to "a\_routine".

```
CALL a_routine(loc_1, 17.5, 121, "some string")
```

## 7.7 Reentrant and Recursive Programs

The following sections describe reentrant and recursive programs.

### Reentrant Programs

The V+ system allows the same program to be executed concurrently by multiple program Tasks. The program can be reentered while it is already executing. This allows different Tasks that are running concurrently to use the same general-purpose subroutine.

To make a program reentrant, you must observe the following guidelines when writing the program.

- Global variables can be read but must not be modified.
- Local variables should not be used.
- Only automatic variables and subroutine arguments can be modified.

In special situations, local variables can be used and global variables can be modified, but then the program must explicitly provide program logic to interlock access to these variables. The TAS real-valued function may be useful in these situations.

## Recursive Programs

Recursive programs are subroutines that call themselves, either directly or indirectly. A direct call occurs when a program calls itself.

Indirect calls are more common and occur when program A calls program B, which eventually leads to another call to program A before program B returns. For example, an output routine may detect an error and call an error-handling routine, which in turn calls the original output routine to report the error. If recursive subroutine calls are used, the program must observe the same guidelines as for reentrant programs. Refer to Reentrant Programs on page 83 for more information.

**IMPORTANT:** Ensure that the recursive calls do not continue indefinitely. Otherwise, the program Task will run out of stack space.

## 7.8 Asynchronous Processing

A significant feature of V+ system is the ability to respond to an event such as an external signal or error condition that occurs independently of the program's control structure. This type of program execution is considered to be asynchronous since its execution is not synchronized with the normal program flow. V+ will react to an event by invoking a specified program as if a CALL statement had been executed, but only if event handling is properly enabled.

Asynchronous processing is enabled by the REACT, REACTE, and REACTI keywords. Each program Task can use these keywords to prepare for independent processing of events. In addition, the WINDOW keyword enables asynchronous processing of belt window violations when the robot is tracking a conveyor belt.

Sometimes a reaction must be delayed until a critical program section has completed. Since some events are more important than others, a program should be able to prioritize events. V+ allows the relative importance of a reaction to be specified by a program priority value from 1 to 127. The higher the program priority setting, the more important the reaction.

A reaction subroutine is called only if the main program priority is less than that of the reaction program priority. If the main program priority is greater than or equal to the reaction program priority, execution of the reaction subroutine is deferred until the main program priority drops. Since the main program (for example, the robot control program) normally runs at program priority zero and the minimum reaction program priority is one, any reaction can normally interrupt the main program.

The main program priority can be raised or lowered with the LOCK program command keyword and its current value can be determined with the PRIORITY real-valued function. When the main program priority is raised to a certain value, all reactions of equal or lower priority are locked out.

When a reaction subroutine is called, the main program priority is automatically set to the reaction program priority, thus preventing any reactions of equal or lower program priority from interrupting it. When a RETURN statement is executed in the reaction program, the main program priority is automatically reset to the level it had before the reaction subroutine was called.

**Additional Information:** Refer to the LOCK, PRIORITY, REACT, and REACTI in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about reactions and program priority.

## Error Trapping

Normally, when an error occurs during execution of a program, the program is terminated and an error message is displayed on the Monitor Window. However, if the REACTE keyword has been used to enable an error-trapping program, the V+ system invokes that program as a subroutine instead of terminating the program that encountered the error. Each program Task can have its own error trap enabled.

Before invoking the error-trapping subroutine, V+ locks out all other reactions by raising the main program priority to 254.

**Additional Information:** Refer to the description of the REACTE keyword in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information on error trapping.



# Chapter 8: T20 Pendant Programming

This section provides an overview of optional customization functions for the T20 pendant. Refer to the *T20 Pendant User's Manual (Cat. No. I601)* for information on installing and operating the pendant.

## 8.1 Attaching and Detaching the T20 Pendant

Before a V+ program can communicate with the pendant, the pendant must first be attached using the ATTACH keyword. The pendant should be detached when the associated V+ programs are finished.

**NOTE:** The logical unit number (LUN) for the pendant is 1 in the following examples.

As with all I/O devices, the IOSTAT keyword should be used to check for errors after each I/O operation.

When the pendant has been attached by a V+ program, the user can interact with the pendant without selecting manual mode.

The following example will attach the pendant with the ATTACH keyword.

```
t20_lun = 1
ATTACH (t20_lun)
```

The following example will detach the pendant with the DETACH keyword.

```
DETACH (t20_lun)
```

## 8.2 Writing to the T20 Pendant Display

The pendant allows users to display a title and a message body and to modify the labels for function keys F1 through F4. Any field may be an empty string (""). The message body can process HTML-tagged code.

Use the following examples to understand how to write and clear messages from the pendant display.

The following example will display "Operator Control" as the title, "Select Option from buttons below" as the message, and create labels for the F1 and F2 buttons as shown in the figure below using the PDNT.WRITE keyword.

```
$p.title = "Operator Control"
$p.msg[0] = "Select Option from buttons below"
$p.f[1] = "Apps"
$p.f[2] = "Status"
$p.f[3] = ""
$p.f[4] = ""
PDNT.WRITE $p.title, $p.msg[], $p.f[1], $p.f[2], $p.f[3], $p.f[4]
```



Figure 8-1. Writing to the Pendant with PDNT.WRITE

The following example will display "Waiting on Parts" and "Please add parts to Feeder" as shown in the figure below using the PDNT.NOTIFY keyword.

```
PDNT.NOTIFY "Waiting on Parts", "Please add parts to Feeder"
```

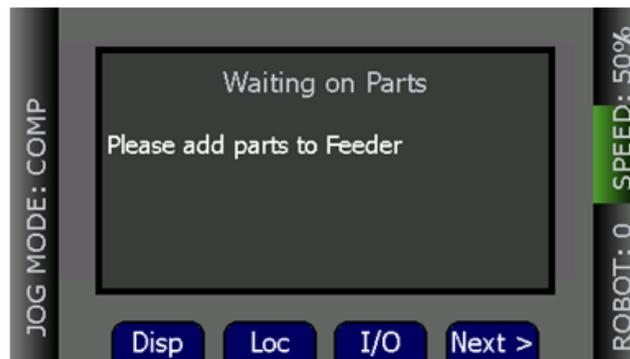


Figure 8-2. Writing to the Pendant with PDNT.NOTIFY

**Additional Information:** The PDNT.CLEAR keyword clears the pendant display and returns to the Home 1 screen.

## 8.3 Detecting User Input

Input from the pendant can be received from a single key press from any of the keys that can be detected. Single-key presses can be monitored in three different modes as detailed in this section.

1. Keyboard mode: the keys can be monitored like keys on a normal keyboard.
2. Toggle mode: the keys can be monitored in toggle mode (ON or OFF). The state of the key is changed each time the key is pressed.
3. Level Mode: the keys can be monitored in level mode. The state of the key is considered ON only when the key is held down.

### Detecting Pendant Key Presses

Individual pendant key presses are detected with the PENDANT program command keyword. This keyword returns the number of the first acceptable key press.

There are three different key press modes for the pendant. The KEYMODE keyword sets which mode is to be used. The key press modes available are described below.

1. Keyboard Mode
2. Toggle Mode
3. Level Mode

The following figure provides a reference for the numbers of the keys on the T20 pendant.



Figure 8-3. T20 Pendant Key Map

**NOTE:** On the T20 pendant, all green keys and Select Robot will still have their normal functionality when in a Custom Window (PDNT.WRITE). All other keys will have their primary functionality disabled. These keys are intended to be read by the V+ program through the PENDANT program command keyword.

## 8.4 Other Pendant Programming Functions

Other pendant functions listed below are possible with the use of the PENDANT program command. Refer to the *V+ Keyword Reference Manual (Cat. No. I672)* for more information.

- Monitor key states
- Monitor the state of the pendant
- Set keyboard modes
- Monitoring the speed signal setting
- Monitor the currently displayed screen number
- Monitor the version number of the pendant



## A.1 Warning, Information, and Error Messages

This section describes system errors, warnings, and information messages that can occur during system operation.

**Additional Information:** Refer to Error Trapping on page 85 and the REACTE keyword in the *V+ Keyword Reference Manual (Cat. No. I672)* for more information about message management.

### System Warning Messages

Use the following information to reference system warning messages.

Table 9-1. V+ System Warning Messages

Message Number	Error in Code	Error ID	MessageDescription	Description	Corrective Action
1	IE_WTMV	1010	Waiting for motion to complete	Contact your local OMRON representative.	
2	IE_WRNPOW	202	Power is now ON		
3	IE_WRNPOF	203	Power is now OFF		
4	IE_WRNCLS	101	Network connection closed	A client connection has closed on the given logical unit.	This is an informational message only.
5	IE_WRNOPN	100	Network connection opened	A new client connection has been established on the given logical unit.	This is an informational message only.
6	IE_WRNHTR	58	Press and hold pendant enable switch	You are attempting to enable High Power while in manual mode. This operation requires that the MCP enable switch be tested. High Power cannot be	Push, then release, then hold the enable switch within a limited time. Alternately, you can change from manual mode to automatic mode.

Message Number	Error in Code	Error ID	MessageDescription	Description	Corrective Action
				enabled unless you toggle the switch to test it.	
7	IE_WRVFP	57	Press High Power button to enable power	The High Power ON/OFF button on the Front Panel must be pressed to complete the process of enabling High Power.	When the High Power ON/OFF button on the Front Panel blinks, press the button to complete the two-step process of enabling High Power. You must press the button within the time period configured.
8	IE_WNOHID	52	*Warning* Protected and read-only programs are not stored	A STORE command has been executed while protected and/or read-only programs are loaded in the V+ system memory. The protected and read-only programs are not stored in the new disk file.	Use the FCOPY command if you want to move read-only programs from one disk to another. Protected programs cannot be moved from one disk to another.
9	IE_WNOCAL	51	*Warning* Not calibrated	The robot servo system and joint position sensors are not calibrated. Any location variables that are defined may not represent the locations desired.	Enter a CALIBRATE command.
10	IE_WRNDRY	50	Executing in DRY.RUN mode	The DRY.RUN switch is	Abort execution of the

Message Number	Error in Code	Error ID	MessageDescription	Description	Corrective Action
				enabled and program execution has been requested.	program and disable the DRY.RUN switch.

## System Information Messages

Use the following information to reference system information messages.

Table 9-2. V+ System Information Messages

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
11	IE_SSDONE	13	Flag single-step done	Contact your local OMRON representative.	
12	IE_BPTMES	17	Breakpoint hit	A breakpoint was encountered before the indicated step. Any output associated with the breakpoint is displayed after the message.	Enter a PROCEED, RETRY, SSTEP, or XSTEP monitor command to resume program execution. Otherwise, enter any other monitor command.
13	IE_PAUSEM	9	PAUSE command hit	A PAUSE command has been executed, and the current program has suspended execution.	Any monitor command can be entered. To continue execution of the program, type proceed followed by the task number if it is not 0.
14	IE_FHALTM	8	HALT command hit	A HALT command has been executed. The execution of the current program has terminated.	Any monitor command can be entered, but PROCEED cannot be used to resume program execution.
15	IE_DOHALT	6	DO done	Contact your local OMRON representative.	
16	IE_DOSTAR	5	Starting a DO command		
17	IE_UHALT	4	Program task stopped		
18	IE_FINI	3	Program completed	The program has been executed the number of times	Any monitor command can be

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				specified in the last EXECUTE command.	entered. The PROCEED operation cannot be used to resume program execution.
19	IE_USTART	2	Starting program execution	Contact your local OMRON representative.	
20	IE_SUCCESS	1	Success		
21	IE_WARN	0	Warning		

## System Error Messages

Use the following information to reference system error messages.

Table 9-3. V+ System Error Messages

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
22	IE_NOTYET	-1	Not implemented	This keyword is not implemented in this version of V+.	Check the version of V+ and verify that the keyword is supported.
23	IE_OBSOL	-2	Obsolete Instruction	This keyword is no longer supported.	Do not use the keyword.
1	IE_NOEXIST	-3	Instruction does not exist		
24	IE_ILLCMD	-300	Illegal monitor command	The name of the monitor command just attempted was not recognized by the system, possibly because it was mistyped or because it was a Program command and not a Monitor command.	Verify the spelling of the Monitor command name and enter it again. Use the DO command to invoke a Program command from the terminal.
25	IE_NOPROG	-301	No program specified	No program was specified for an EXECUTE command or DEBUG	Type the line again, providing a program name.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				command, and no previous program is available as a default.	
26	IE_NODO	-302	DO not primed	A DO program command was attempted without specifying a program command to be executed and no previous DO had been entered.	Provide the desired instruction with the DO program command.
27	IE_NOTAUTO	-303	Controller not in automatic mode	An attempt has been made to initiate program execution or PRIME a program from the monitor window when the controller is not in automatic mode.	Select automatic mode by moving the switch on the front panel to the automatic position, or by activating the proper switch on a custom control panel. Retry the previous command.
28	IE_BADFRE	-305	Storage area format error	During execution of a FREE command, V+ has detected that programs or data in RAM may have been corrupted. This may have been caused by a momentary hardware failure or a software error.	Attempt to save as much as possible onto the disk. Then enter a ZERO command or power OFF the controller and restart the system.
29	IE_PRGNEX	-307	Program not executable	Because of program errors detected during loading or upon exiting from the editor, this program cannot be executed.	Edit the program to remove any errors.
30	IE_PRGILK	-308	Program interlocked	An attempt has	Abort the program or exit the editor as

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				been made to access a program that is already in use by some V+ process. For example, you have attempted to delete or edit a program that is being executed, or execute a program that is being edited.	appropriate and retry the operation.
31	IE_PRGDEF	-309	Program already exists	An attempt has been made to LOAD a program that already exists, or to COPY or RENAME a program to a name that is already in use.	Delete the conflicting program or use a different name.
32	IE_HIDERR	-310	Can't access protected or read-only program	An attempt has been made to edit a protected or read-only program.	
33	IE_PRGACT	-311	Invalid when program task active	An attempt has been made to begin execution of a robot or PC program task when that task is already active.	Abort the currently executing task, or execute the program as a different task.
34	IE_CANTPR	-312	Can't start while program running	An attempt has been made to start execution of a program from the pendant while a program is already executing as task #0.	Stop the program currently executing and then retry the operation.
35	IE_CNTPRO	-313	Can't go on, use EXECUTE or PRIME	An attempt has been made to	Use the EXECUTE or PRIME monitor command, or the pendant

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				continue the execution of a program that has completed or stopped because of a HALT program command. Normally, an error results when a PROCEED, RETRY, or XSTEP monitor command is entered (or the pendant RUN/HOLD button is pressed) after a program has completed all its cycles.	PRIME function, to restart the program from the desired instruction.
36	IE_NOTENB	-314	Switch can't be enabled	An ENABLE command for a certain switch has been rejected because of some error condition. For example, ENABLE POWER fails if the system is in FATAL ERROR state.	Review the description for the switch you are trying to enable, correct the error condition, and try again.
37	IE_BADCFG	-315	Invalid software configuration	<p>During initial startup, V+ has detected that the system software is not configured properly for the options or hardware present.</p> <p>An attempt to control a robot that is not synchronized has been made.</p>	<p>Power OFF the controller and try starting it again. Make sure that the boot disk you are using is valid for your controller.</p> <p>Make sure the network connection to the robot is correct.</p> <p>Change the program so it does not attempt to control a robot that is not synchronized.</p> <p>If the problem persists, contact your local OMRON rep-</p>

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
					representative.
38	IE_BADCHK	-316	Software checksum error	During processing of a FREE command, the V+ system has detected a checksum error in the system memory. This indicates a problem with the system software or hardware. Note that a checksum error is introduced if any patches are made to the system software after the system is loaded from disk and started up.	The following codes are appended to the message indicating where the error occurred: Os (operating system) eV+ eV+ interpreter or trajectory generator Vi (vision software) Sv (servo software) Report the error and information about any possible contributing circumstances to your local OMRON representative. You can continue to use the system, but you should keep in mind the possibility of a problem with the hardware.
39	IE_PRNACT	-318	Program task not active	An attempt was made to abort a task that was not active.	None required if the correct task number was specified. Otherwise, use the STATUS command to determine which task number should have been used.
40	IE_PRNUSE	-319	Program task not in use	A program task cannot be accessed because it has never been used.	Such program tasks do not use any system memory and do not appear in the STATUS display.
41	IE_NODPRG	-350	Can't delete .PROGRAM statement	An attempt has been made to delete the .PROGRAM statement while editing a program.	To change the .PROGRAM statement, replace it with another .PROGRAM statement. To delete lines at the beginning of the program, move down to line 2 before issuing delete commands.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
42	IE_PRG1ST	-351	First statement must be .PROGRAM	An attempt was made to insert or deposit a program statement above the .PROGRAM statement, which must be the first statement in the program.	Move the cursor to below the .PROGRAM line of the program before attempting to insert or deposit statements.
43	IE_READER	-352	Invalid in read-only mode	A V+ editor function was attempted that cannot be performed while accessing a program in read-only mode.	Change to editing the program in read-write mode, or use a different editor command.
44	IE_ONLYCU	-353	No other program referenced	A command was issued that attempted to reference a previously edited program, but no other program has been edited during the current editing session.	Use the New or GoTo function-key command to change to a new program.
45	IE_TOLONG	-354	Line too long	An operation was attempted that would have resulted in accessing a program step that contains too many characters. A single program step can contain at most about 150 characters.	Enter the program step as two or more separate steps.
46	IE_NOEXIT	-355	Can't exit while lines attached	You attempted to terminate execution of the editor while lines were present in the attach buffer.	You can use Shift+Copy to deposit the contents of the attach buffer into the current program. You can also use Esc+K to delete lines from the

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				The attach buffer must be empty before the editor can be exited.	attach buffer (99 Esc+K deletes up to 99 lines from the buffer).
47	IE_NOFIND	-356	Not found	The search operation was unable to locate the specified string.	Enter a new search string, or consider this an informational message and continue with other operations.
48	IE_DMMES	-357	Recursive macros illegal	An attempt was made to execute a macro recursively. That is, the macro contained a command character sequence that (directly or indirectly) restarted execution of the macro.	Change the macro definitions as necessary to make sure neither macro invokes itself. You can have the U macro invoke the Y macro, or vice versa, but not both.
49	IE_SCANCL	-358	Cancelled	An editor, debugger, or pendant operation has been terminated due to operator intervention.  This is usually an informative message to acknowledge the cancellation of the operation.	
50	IE_DEBUGE	-359	Illegal in debug monitor mode	An operation was attempted that is not accepted in debug monitor mode.	Use a different command, change to debug editor mode, or exit from the program debugger.
51	IE_NOTDBG	-360	Must be in debug mode	An editor function was attempted that is accepted only when the program debugger is active.	Use a different editor command or activate the program debugger with the SEE editor DEBUG extended command or the DEBUG monitor command.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
52	IE_CNTSWC	-361	Can't change modes while task running	A command was issued to change from debug monitor mode to debug editor mode while the program task being debugged was executing. You can change to debug editor mode only when the associated task is stopped.	Stop execution of the program task being debugged, or continue without using debug editor mode.
53	IE_NWPRGI	-362	Can't execute from SEE program command	An attempt has been made to use a SEE editor command that cannot be used after the editor has been initiated with the SEE program command.	Enter another command or exit the editor and reenter from the V+ monitor.
54	IE_UNKECM	-363	Unknown editor command	An unknown keystroke or extended command was issued while using the SEE program editor.	Enter another command.
55	IE_NOCCOM	-364	Can't create program in read-only mode	An attempt has been made to initiate editing of a program in read-only access mode, but the program does not exist.	If the program name was entered incorrectly, enter the command again with the correct name. Do not select read-only access (with /R) when creating a new program.
56	IE_WRIOER	-365	Illegal in read-write mode	An editor function was attempted that cannot be performed while accessing a program in read-write mode.	Change to editing the program in read-only mode, or use a different editor command.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
57	IE_PONSTK	-366	Invalid when program on stack	An attempt has been made to edit a .PROGRAM or AUTO statement while the program appears on some task execution stack. While a task is on a stack, its subroutine arguments and automatic variable values are kept on the stack. Changes to these statements modify the stack, which is not allowed.	Remove the program from the stack by allowing the task to run until the desired program executes a RETURN command or issue a KILL monitor command to clear the stack.
58	IE_BPTNAL	-380	Breakpoint not allowed here	An attempt has been made to set a breakpoint before the first executable statement of a program.	
59	IE_ABORT	-400	Abort	The last command requested, or the program that was executing has been aborted.	
60	IE_UNDVAR	-401	Undefined value	A variable has been referenced that has not been assigned a value.  Using the SEE editor, an attempt has been made to use a macro, return to a memorized cursor position, or perform a repeat string search or	Assign the variable a value or correct its name.  Define the macro, record a cursor position, or enter the desired search/replacement string (s).

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				change without first performing the appropriate initialization sequence.	
61	IE_BADNUM	-402	Illegal value	A numeric or expression value that is not in the allowed range was specified within a keyword.	Edit the program to use an allowed value.
62	IE_BADASN	-403	Illegal assignment	The assignment operation just attempted was invalid, possibly because it attempted to assign a value to a variable name that is a reserved word or a function.	Reenter the line using a different variable name if necessary.
63	IE_BADARR	-404	Illegal array index	<p>One of the following attempts was made:</p> <p>Used a negative value as an array index.</p> <p>Used a value greater than 32767 as an array index.</p> <p>Specified a simple variable where an array variable is required.</p> <p>Omitted an array index in a situation where it is required (for example, a 1-dimension array is specified when a 2- or 3-dimension array is required).</p>	Correct the line.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				<p>Specified an explicit index in an argument for an V+ operation that requires a null array.</p> <p>Specified an index to the right of a blank index for a multiple-dimension array.</p>	
64	IE_BADSIG	-405	Illegal digital signal	<p>A number or bit field specifies a digital signal that is not in one of the allowed ranges or that is not installed.</p> <p>Attempting to set software signal 2032 (brake solenoid) will also give this error.</p>	Correct the signal number and check your digital I/O configuration.
65	IE_UNDNAM	-406	Undefined program or variable name	The program or variable, referenced in a command or program step, does not exist possibly because the name was mistyped.	If the correct name was entered, create the program or variable using one of the V+ editors or the appropriate monitor commands, or by loading from a disk file.
66	IE_ILLARG	-407	Invalid argument	An argument for a keyword is not in the accepted range.	Verify the range for the arguments used.
67	IE_SUBAMM	-408	Program argument mismatch	The arguments in a CALL, CALLS, or EXECUTE command do not match the arguments in the program being referenced because they	Modify the CALL, CALLS, or EXECUTE command, or the .PROGRAM statement of the referenced program, so that the argument types match. If arguments are omitted in the CALL, CALLS, or EXECUTE

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				are of different types.	command make sure the appropriate commas are included to position the arguments that are present.
68	IE_ARITHO	-409	Arithmetic overflow	The result of a calculation was outside the allowable range for real variables or V+ has encountered a number that is outside the allowed range for integers while converting a real-valued number to a decimal, hexadecimal, octal integer, or logical value. Logical values use 32-bit integers, but most program commands that require integer arguments allow only 16-bit integers. Real variables can have only magnitudes in the range from 5.4E-20 to 9.2E+18.	Modify the program as required.
69	IE_NEGSQR	-410	Negative square root	An attempt has been made to calculate the square root of a negative number.	Correct the program as required.
70	IE_NOFRES	-411	Not enough storage area	There is no more space in RAM for programs or variables.	Delete unused programs and variables. If the memory is fragmented because of numerous deletions, it

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
					can be consolidated by issuing the commands STORE save_all, ZERO, and LOAD save_all. This writes the memory contents to the disk and reads them back into memory. This procedure does not retain any variables that are not referenced by any program in memory, nor does it retain the values of variables that are defined to be AUTO or LOCAL.
71	IE_MISLBL	-412	*Branch to undefined label* Step XXX	A program command references a program label that is not defined in the program. Either the label is missing or was mistyped when defined or in the reference.	Check the label definition and reference.
72	IE_SUBERR	-413	Not enough program stack space	An attempt was made to call a subroutine, process a reaction subroutine, or allocate automatic variables when the stack for the program task was too full.	Reorganize the program logic to eliminate one or more nested subroutine calls or reactions. Eliminate some of the automatic variables that are allocated by the programs.  Use the STACK monitor command to increase the size of the stack for the program task. The program may be restarted with the RETRY command.
73	IE_MCERR	-414	Can't mix MC & program instructions	A program command has been encountered during processing of a command pro-	Edit the command program to use the DO program command, or remove the MC instruction from the normal program.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				gram, or an MC instruction has been encountered in a normal program.	
74	IE_STGOVF	-416	String variable overflow	An attempt has been made to create a string value that is greater than the maximum string length of 128 characters.	Edit the program to generate strings of the proper length.
75	IE_STRSHT	-417	String too short	A program command expected a string argument with a certain minimum length and received one that was too short.	Review the syntax for the program command and edit the program to pass a string of the correct length.
76	IE_NXMERR	-418	Illegal memory reference	An operation has attempted to reference an invalid memory address that is either out of the allowed range, or that is not in use for any input/output module.	Correct the address or install the missing module.
77	IE_CMDACT	-419	Illegal when command program active	A command program is active and an attempt has been made to execute a keyword that interferes with operation of the command program. For example, processing a ZERO keyword causes the command program to be deleted from the system	Edit the command program and delete the command causing the error.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				memory.	
78	IE_UNDVCX	-420	Undefined value in this context	An automatic variable or subroutine argument value appears in a monitor command, but the specified program is not on the execution stack for the specified program task. Automatic variables and subroutine arguments have values only when the program that defines them is on a stack.	Change the monitor command to not reference the variables. Make sure that the program is on the expected execution stack. You can place a PAUSE command or breakpoint in the program to stop it while it is on the execution stack.
79	IE_PRGNOT	-421	Program not on top of stack	A DO context specification has referenced an automatic variable or a subroutine argument in a program that is not on the top of the stack for the specified task.	Reenter the DO command and specify the correct program context or eliminate references to automatic variables and subroutine arguments. Use the STATUS command to determine which program is on the top of the stack.
80	IE_FALENB	-422	Function already enabled	Certain functions or operations must not be enabled when they are already enabled or active. ALTER mode is an example of such a function.	Avoid reenabling the function or operation.
81	IE_ILLOPR	-423	Illegal operation	A program command has attempted to perform an operation that is not possible.	Check the command executing when the error occurred. Make sure all conditions necessary for its successful completion are met.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
82	IE_NOCALP	-425	Calibration program not loaded	A program required for calibration has not been loaded from disk. This error occurs if a robot-specific calibration file cannot be found on the disk (i.e., in the same location as the file for the main calibration program), or if a required calibration program is not present in memory when it is expected. The latter situation can occur if the CALIBRATE program command or monitor command is executed with an input mode that does not cause the calibration programs to be loaded from disk, and the programs are not already present in memory.	Reissue the CALIBRATE program command or monitor command with the proper mode. The default mode of zero causes CALIBRATE to automatically load the required programs from disk, perform the calibration, and then delete the programs. Alternatively, issue a CALIBRATE program command or monitor command with mode = 1 (which causes the calibration programs to be loaded into memory), and then reissue the CALIBRATE program command or monitor command that was originally attempted.
83	IE_NOCALF	-426	Can't find calibration program file	While processing a CALIBRATE program command or monitor command, the V+ system cannot find the calibration utility program file CAL_UTIL.V2.	Restore the missing file from the V+ distribution disk to the current default directory, or to the directory \CALIB\ on the local "C" or "D" drive.
2	IE_SYSABORT	-429	Aborted by system	An execution of	Make sure that the controller has started

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				a keyword has been made when V+ is in disabled state. The operation has been aborted.	correctly. Re-attempt the execution of the keyword or program.
84	IE_BADLIN	-450	Can't interpret line	V+ cannot interpret the last program command or monitor command entered.	Verify the spelling and usage, and reenter the line. In the case of an error while loading from the disk, edit the affected programs to correct the indicated lines-they have been converted to bad lines.
85	IE_SWPERR	-451	Unexpected text at end of line	The previous command cannot be recognized by V+, possibly because of a mistyped name or because an argument was specified where none is allowed.	Reenter the line and correct the syntax error.
86	IE_ILLINS	-452	Unknown instruction	A command was entered (or read from a disk file) that was not recognized by the system. This error is often caused by mistyping the command name, or trying to use a command improperly.  If the message occurred while loading a file from the disk, the file was probably created offline, or with a different V+ system (dif-	Correct the line or enter it again, making sure the spelling and usage are correct.  In the case of an error while loading from the disk, edit the program to correct the indicated command.  Statements with errors are turned into bad lines beginning with a question mark.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				ferent version or options), and the indicated line is not compatible with the V+ system in use.	
87	IE_AMBIG	-453	Ambiguous name	The abbreviation used for the last command, function, or system-defined name was not long enough to identify the operation intended.	Reenter the last line using a longer abbreviation.
88	IE_NOARGU	-454	Missing argument	A valid argument was not found for one or more of the arguments required for the requested command. The argument was not present at all or an invalid argument was present.	A possible cause is the use of a single equal sign (=) for the equality relational operator (==). Verify the operation syntax and reenter the line.
89	IE_INVNME	-455	Invalid program or variable name	A user-defined name used in a keyword was not recognized byV+.	Verify the name and retype the line.
90	IE_BADNFM	-456	Invalid number format	A syntax error was detected while reading a number. For example, an 8 or 9 digit was encountered while reading an octal number.	Reenter the line with a valid number.
91	IE_RESWRD	-457	Reserved word illegal	An attempt has been made to use an V+ reserved word	Use a different name for the variable. You can append a prefix or suffix to the attempt-

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				for a variable name.	ted name.
92	IE_SYNERR	-458	Illegal expression syntax	While decoding a numeric or logical expression, a compound transformation, or a string expression, V+ has encountered syntax that it does not understand.	Possible errors include unmatched parentheses, missing variables, or missing operators. Retype the line containing the expression, being careful to follow the V+ syntax rules.
93	IE_MISPAR	-459	Missing parenthesis	An attempt was made to evaluate an expression that did not have correctly matching left and right parentheses.	Correct the expression.
94	IE_MISQUO	-460	Missing quote mark	A quoted string has been encountered that has no matching quote mark before the end of the line.	Insert a quote mark at the end of the string. Strings may not cross line boundaries.
95	IE_ILLFMT	-461	Invalid format specifier	An unrecognized output format specifier was encountered in a TYPE or WRITE, or \$ENCODE keyword.	Edit the program to use a valid format specifier.
96	IE_UNKFUN	-462	Unknown function	While accepting a program statement, V+ has encountered a reference to a command that it does not recognize. This can be due to a mistyped name or the leaving	Verify the spelling and syntax and reenter the line.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				out of an operator between a symbol and a left parenthesis.	
97	IE_BADLBL	-463	Invalid statement label	The program statement label was not an integer from 0 to 65535.	Reenter the line with a valid label.
98	IE_DUPLBL	-464	Duplicate statement label	The same program statement label is used more than once in a user program, likely in a compound transformation but not at the left-most position..	Change one of the duplicate labels.
99	IE_BADTYP	-465	Variable type mismatch	One or more of the variables in the line is of a type inconsistent with the other variables or with the type required by the command.	Verify the syntax for the operation and reenter the line, correcting the mismatch. Delete conflicting global variables, if appropriate.  You may be trying to mix location variables with real-valued variables. If this error occurs upon exiting from the editor, the variable type within the program conflicts with the type of a global variable that is already defined.
100	IE_ILLBLV	-466	Illegal use of belt variable	A belt variable has been used in a context where it is not allowed.	Edit the program to use the belt variable correctly.
101	IE_BADPRG	-467	Illegal .PROGRAM statement	An attempt has been made to enter a line other than a .PROGRAM statement as the first line of a	Move below the first line of the program, or reenter the line correctly.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				program or enter a .PROGRAM statement that contains a syntax error.	
102	IE_DUPARG	-468	Duplicate .PROGRAM arguments	At least two of the arguments in a .PROGRAM statement have the same name.	Edit the .PROGRAM line so that all the arguments have unique names.
103	IE_REDECL	-469	Attempt to redefine variable type	If a program is being edited, the line just entered contains a reference to a variable in a manner inconsistent with its use elsewhere in the program. The most likely problem is confusing a location variable with a real variable. If you just exited from the editor, the named variable conflicts with a global variable that already exists.	If the new use of the variable is correct, you must delete all references to the incorrect variable and then re-enter the statement that caused the error. If the new use is incorrect, use a different variable name. If there is a conflict with a global variable, either use a DELETE_ command to delete that variable, or make the conflicting variable AUTO or LOCAL to the current program.
104	IE_RDFCLS	-470	Attempt to redefine variable class	Upon exiting from the editor, the named variable was found in two of the following places: the .PROGRAM argument list, an AUTO statement, a LOCAL statement, or a GLOBAL statement.	Modify the program to include the variable in only one of these places.
105	IE_MISPDC	-471	Misplaced declaration statement	Upon loading a program or exiting from the pro-	Edit the program to make sure that AUTO and LOCAL statements are preceded

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				gram editor, V+ has encountered an AUTO or LOCAL statement that follows an executable program command.	only by blank lines, comments, or other AUTO and LOCAL statements.
106	IE_CSFERR	-472	*Control structure error* Step X	V+ has detected an incomplete or inconsistent control structure at the specified step when exiting the program editor or loading a program.	Edit the program to correct the control structure. Note that the actual error may not be at the indicated step.
107	IE_BADCON	-473	Control structure error	An incomplete control structure has been encountered during program execution.	Edit the program to correct the control structure.
108	IE_BADDIM	-474	Too many array indices	The specification of an array element contains more than three indexes.	Reenter the line with the correct number of indexes.
109	IE_MISBRK	-475	Missing bracket	In the specification of an array element, a left bracket has been found with no matching right bracket. Either too many left brackets are present or a right bracket has been omitted.	Reenter the line with correctly matching left and right brackets.
110	IE_ILLSQL	-476	Invalid qualifier	An invalid qualifier was specified on the last command.	Enter the command with a valid qualifier.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
111	IE_AMBAUT	-477	Ambiguous AUTO invalid	When exiting from the program editor, V+ has encountered an automatic variable with undetermined type. The system cannot determine if the variable is real-valued or a transformation. Automatic variables cannot be ambiguous, since their storage requirements must be known before they are referenced.	Include the REAL or LOC type specification parameter in the AUTO statement that declares the variable, or reference the variable in a program command in a manner that makes its type clear.
112	IE_FSTFPE	-500	File already exists	There is already a disk file with the name supplied to the last storage request.	Reissue the storage request with a different file name, or delete the old file.
113	IE_NOFILE	-501	Nonexistent file	The requested file is not stored on the disk accessed. Either the name was mistyped or the wrong disk was read.	Verify the file name. Use the FDIRECTORY command to display the directory of the disk.
114	IE_BADIOC	-502	Illegal I/O device command	A command to an I/O device was rejected by that device. Certain devices do not accept all commands. This error may also indicate a hardware problem with the device controller.	Correct the I/O command as required to suit the device. If the problem persists, contact your local OMRON representative.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
115	IE_CEFULL	-503	Device full	There is no more space available on a device. If this error is received for a disk file, it indicates that the disk is full. If there are many small files on the device, this error indicates the directory is full. If this error is received for a servo device, it indicates that an attempt has been made to assign too many servo axes to a single CPU.	Delete unneeded disk files, or use another drive or disk. Reconfigure your system so the maximum number of axes per CPU is not exceeded.
116	IE_IEEOF	-504	Unexpected end of file	<p>If a file was being loaded from the disk, the end of the file was encountered unexpectedly.</p> <p>If a program is reading a file, this error code merely indicates that the end of the file has been reached and should not be interpreted as a real error.</p> <p>This message results if a CTRL+Z is pressed in response to a program PROMPT.</p> <p>A break condition was detected on a</p>	<p>Try again to read the file.</p> <p>Close the file and continue program execution.</p> <p>Treat the program as having been aborted early by user request.</p>

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				serial line.	
117	IE_IEINI	-505	Initialization error	An I/O device reported an error condition during its initialization. Initialization is performed during power-up, after a reset, and may also be performed after certain non-recoverable I/O errors occur.	Be sure that the hardware for the I/O device is properly installed. Repeat the failed I/O operation. If the problem persists, contact your local OMRON representative.
118	IE_IEFAO	-506	File already opened	A disk file is already open on a logical unit, and another open request has been attempted.	Modify the program to use a different logical unit number for the file you want to open, or perform an FCLOSE operation on the file currently open on the specified logical unit number before performing the FOPEN operation.
119	IE_DEVNTR	-508	Device not ready	<p>The requested disk device (or remote network task) is not prepared to communicate with the V+ system.</p> <p>A limited-access device like the terminal, the pendant, or a serial line is attached to a different program task.</p> <p>The SIG.INS or SIGNAL keyword was used for a signal that does not exist.</p> <p>An attempt to control a robot that is not con-</p>	<p>Make sure the network connection to the specified robots is present.</p> <p>Change the program so it does not attempt to control a robot that is not connected to the network.</p> <p>If a limited-access device is specified, ABORT and KILL the program task that has it attached, or wait for the program task to release the device. If the intended device is on the network, verify that the proper connections are made and that the remote system is operating correctly. ABORT and KILL the program</p>

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				nected to the network was made.	task that has the device attached, or wait for the task to release the device.
120	IE_IECHK	-510	Data checksum error	An error was detected while transferring information to or from an external device.	Attempt the transfer again. If the problem persists, contact your local OMRON representative.
121	IE_BADFIL	-512	File format error	The requested disk file is not in a format acceptable to V+ because either it was not created by V+ or the file has been corrupted.	Use another diskette or reference another file.
122	IE_IEFNO	-513	File not opened	A program request was made to read or write data from a disk device when no file was open.	Modify the program to open the file before attempting to read or write data.
123	IE_SUBNAME	-514	File or sub-directory name error	The specified file name or sub-directory was not a valid disk file name, the directory path contained invalid syntax, or the directory path was too long.	Retry the operation with a correct file name or subdirectory name. Verify that syntax of the directory path is correct. Verify that any default directory path specified by the DEFAULT command is correct. Verify that the total directory path is not too long when the default is combined with the current file specification.
124	IE_LUNBSY	-515	Already attached to logical unit	A program has executed more than one ATTACH program command for a specific logical unit, without executing a DETACH	Check the program logic and remove redundant ATTACH program commands.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				in between. The program is still attached to the logical unit after this error occurs.	
125	IE_NOTATT	-516	Not attached to logical unit	A program has attempted to perform I/O to a logical unit that it has not attached with an ATTACH command.	Logical unit 4 allows output without being attached, but all other logical units require attachment for both input and output. Edit the program to make sure it attaches a logical unit before attempting to use it to perform I/O.
126	IE_ILLCHAN	-518	Illegal I/O channel number	An internal I/O channel number has been encountered that is invalid. This indicates an V+ internal software problem.	Contact your local OMRON representative.
127	IE_ICE	-519	Driver internal consistency error	An I/O device or servo has responded in an unexpected manner.	Retry the operation that caused the error. If it persists, contact your local OMRON representative.
128	IE_IEDAT	-522	Data error on device	An error was detected while attempting to read information from an external device, possibly because a disk has been damaged or was not formatted properly.	Attempt the read again. Make sure the correct disk is being used, that it is properly installed in the drive, and that it is formatted.
129	IE_IEDAO	-524	Communications overrun	Data has been received on an I/O device faster than V+ is processing it, and some data	Modify the program to service the I/O device more often, add a handshaking protocol, or slow down the transmission rate to

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				has been lost. This happens only on the serial interface line or the network.	V+.
130	IE_ILGRED	-525	Illegal I/O redirection specified	An unacceptable I/O redirection has been specified in a DEFAULT monitor command, a disk I/O monitor command (LOAD or STORE_), or in an ATTACH program command. Either there is a syntax error, or the requested redirection is not allowed for your I/O configuration.	Check the syntax of the offending statement. Make sure that the requested redirection device is allowed on your I/O configuration.
131	IE_NODATA	-526	No data received	An I/O read request without wait has not found any data to return.	Continue polling the I/O device until data is received, or use a read request that waits automatically for data to be received.
132	IE_BADLUN	-527	Illegal user LUN specified	An I/O command has specified a logical unit number (LUN) that is not defined in the V+ system, or cannot be accessed in the manner attempted. Refer to the description of the ATTACH keyword for a list of the valid logical unit numbers and the	Edit the program to use a logical unit number appropriate for the keyword.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				devices to which they apply.	
133	IE_BADRLN	-528	Illegal record length	An FOPEN program command has specified a record length that is not acceptable. For example, the value is negative or too large, or the record length is zero with random-access mode specified.	Edit the program to specify a correct record length or specify sequential-access mode.
134	IE_IOOVER	-529	Output record too long	A TYPE, PROMPT, or WRITE command has attempted to output a line that is too long.	The maximum line length is 512 characters.  Change the program to output less information from each command. You can concatenate the output from separate commands by using /S to suppress the carriage return and line feed normally done at the end of each TYPE output.
135	IE_IVHCFG	-533	Invalid hardware configuration	Contact your local OMRON representative.	
136	IE_IEPRO	-530	Protection error	An I/O operation cannot be performed because it attempted to write to a disk that is write protected or you do not have the proper access status.	Check the diskette to make sure the write-protect tab is in the correct position. Use an FDIRECTORY command to display the disk directory. If the file has protected or read-only protection, you cannot access it in the way attempted.
137	IE_IETMO	-531	Communication time-out	An I/O operation has not completed	Make sure the remote device is communicating. Make sure connections to

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				within the allotted time interval. For data communications, the remote communications device has not properly acknowledged data that was sent.	the remote device are operating properly.
138	IE_IENER	-536	Too many network errors	The number of errors detected by the DDCMP protocol has exceeded the maximum allowed. The local protocol is stopped, and all pending I/O requests are completed with this error.	Use the NET monitor command to determine the type of errors that have occurred. Check for noise on the communication line, errors in the remote DDCMP implementation, or program logic that sends messages faster than they can be processed. Use the appropriate FCMND command to increase the maximum number of errors.
139	IE_IEUNN	-537	Unknown network node	A reference has been made to a network node address that is not known by the local network.	Make sure that the correct node address was specified. make sure that the remote node is active and connected to the network. If explicit routing tables are used, make sure that they specify this node.
140	IE_IEILC	-540	Invalid connection specified	An invalid logical network connection has been specified. For example, a zero connection ID is invalid.	Specify a valid logical connection ID.
141	IE_IEINP	-541	Invalid network protocol	A message has been received and rejected by a remote node because it does	Verify that the network software version on the remote node is compatible with the network software on

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				not follow the expected protocol.	the local node. DISABLE and ENABLE the affected network nodes and retry the operation.  If the problem persists, contact your local OMRON representative.
142	IE_NETINE	-543	Illegal when network enabled	An attempt has been made to perform certain network functions that require that the network is disabled while the network is still enabled.	Disable the network and retry the operation.
143	IE_NOTCFA	-544	Not configured as accessed	An attempt has been made to access a serial line or other I/O device in a manner for which it is not configured.	Verify the proper way to access the serial line for the current configuration. Use the configuration utility program to display the serial line configuration and change it if desired.
144	IE_NOSUBDIR	-545	Nonexistent subdirectory	The subdirectory referenced in a file specification does not exist on the disk that is referenced.	The subdirectory may be part of a default directory path set by the DEFAULT monitor command.  Make sure that the subdirectory name was entered correctly.  Make sure that the correct disk drive was referenced and that the correct diskette is loaded.  Use an FDIRECTORY command to display the directory containing the subdirectory.  Make sure that the default directory path is correct.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
145	IE_SUBINUSE	-547	Subdirectory in use	An attempt has been made to delete a subdirectory that still contains files or that is being referenced by another operation (for example, an FDIRECTORY command).	<p>Make sure that all the files within the subdirectory have been deleted.</p> <p>Make sure that no other program tasks are referencing the subdirectory.</p> <p>Retry the delete operation.</p>
147	IE_TOOMNY	-553	Too many arguments	Too many arguments were specified for the last command or instruction.	Reenter the command or instruction but with the correct number of arguments.
148	IE_IENoor	-559	Out of network resources	<p>Too many ports are simultaneously in use for networking.</p> <p>There are no more buffers available for incoming and outgoing packets.</p> <p>Too many drives are being mounted.</p> <p>Too many calls were made simultaneously from separate tasks to a non-functional server.</p> <p>Too many node names are being defined.</p> <p>An incoming IP packet was fragmented into too many pieces and V+ was unable to reassemble it.</p>	Correct the problem generating the error.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
149	IE_ERNTIM	-562	Network timeout	This error occurs when a network transaction is initiated but no reply is received from the server.	Check network integrity. Make sure the server running. Make sure the correct IP address is being used.
150	IE_IENFTL	-569	File too large	The operation caused a file to grow beyond the server's limit.	Close the file, open a new file, and retry the previous operation.
152	IE_HDWSTP	-600	Stopped due to servoing error	Program execution has stopped because of one or more servo errors.	Correct the source of the reported servo errors, referring to your system hardware manual as required.
153	IE_NOARM	-601	Robot not attached to this program	An attempt has been made to execute a robot-control command in one of the following invalid situations:  The system is not configured to control a robot.  There is no robot connected to the system.  The robot is attached to a different program task.	Make sure the system is booted from the proper system disk, or remove the robot-control command.  Connect the robot or enable the DRY.RUN system switch.  Modify the program logic as required to ensure that only one program task is controlling the robot at any given time.
154	IE_ARMALA	-602	Robot already attached to program	A program has executed more than one ATTACH command for the robot, without executing a DETACH in between. Or an attempt has been made to	Check the program logic-remove redundant ATTACH command or DETACH the current robot before attempting to SELECT another robot.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				SELECT another robot when one is already attached. The robot is still attached even after this error occurs.	
155	IE_NOTCMP	-603	COMP mode disabled	The command attempted requires computer control of the robot, but COMPUTER mode was not selected on the pendant.	Select COMP mode on the pendant or enable DRY.RUN mode from the terminal, then reissue the command.
156	IE_PWISOF	-604	Robot power off	The requested operation cannot be performed because High Power is OFF.	Enable High Power and retry the operation.
157	IE_NOTCAL	-605	Robot not calibrated	An attempt has been made to execute a robot-control program when the robot is not calibrated. No motion is allowed until the robot is calibrated.	If you want to use the robot, issue a CALIBRATE command.  Enable the DRY.RUN switch to allow program execution without using the robot.
158	IE_AIRERR	-607	No air pressure	V+ detected that the air supply to the robot brakes and hand has failed. High Power is turned OFF and cannot be turned ON until the air pressure is restored.	Restore the air pressure, turn high power back on, and resume program execution.  If the problem persists, contact your local OMRON representative.
159	IE_PANBUT	-608	External E-STOP	An Emergency Stop condition on E-STOP channel 1 has been detected by the	If safe to do so, attempt to enable High Power and note any different error messages which

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				CPU. Normally, this message is suppressed and the cause of the E-STOP is reported instead.	occur. Verify that the CIP is connected securely. If this error occurs frequently, contact your local OMRON support.
160	IE_BADJTN	-609	Illegal joint number	A joint number has been specified out of the allowed range.	Correct the joint number.
161	IE_OUTRNG	-610	Location out of range	V+ has encountered a location that is too far away to represent (possibly within an intermediate computation) or that is beyond the reach of the robot.	This probably indicates an error in a location function argument value or in a compound transformation.  Verify that you are using location functions and operations correctly and edit the program as required.
162	IE_CNTJI	-611	Must use straight-line motion	A joint-controlled motion command was attempted while the system was in a mode requiring that only straight-line motions be used. For example, while tracking a conveyor, only straight-line motions can be used.	Change the motion command to one that requests a straight-line motion.
163	IE_CANCFG	-612	Straight-line motion can't alter configuration	A change in configuration was requested during a straight-line motion. This is not allowed.	Delete the configuration change request, or use a joint-interpolated motion instruction.
164	IE_CNTMVE	-613	Illegal motion from here	The motion just attempted cannot be performed from the current robot location.	Perform the appropriate operation sequence before retrying the desired motion.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
165	IE_BLTER	-614	Attempt to modify active belt	A program command has been executed that modifies the belt variable that is currently being tracked by the robot.	Change the program in order not to modify the variable while the robot is tracking it.
166	IE_NOBELT	-615	Belt not enabled	A robot operation that references a moving conveyor belt has been attempted when the conveyor tracking feature is disabled.	Issue an ENABLE BELT command and retry the operation.
167	IE_WINERR	-616	Belt window violation	Either a robot motion has been planned that moves the robot outside of the belt window, or the robot has moved outside of the belt window while tracking the belt.	Modify the program so that the robot does not move outside the belt window. Check the BELT.MODE system parameter and the WINDOW program command for different ways to define the belt window.
168	IE_BLTDED	-617	Belt servo dead	The belt processor isn't responding to commands from V+.	After saving the programs, power down the controller and power it up again. If this error occurs repeatedly, contact your local OMRON representative.
169	IE_LTOCLS	-618	Location too close	An attempt has been made to move the robot to a location that is too close to the robot. This probably indicates an error in the value of a location function argument or an	Verify that you are using location functions and operations correctly and edit the program as required.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				incorrect compound transformation.	
170	IE_BADORI	-619	Invalid orientation	A motion has been requested to a location that is defined by a transformation with its orientation pointed up instead of down.	Correct the definition of the destination transformation. For example, you may need to correct the base transformation in the compound transformation. For SCARA robots, the p component of all destination transformations should be approximately 180 degrees.
171	IE_MANBTT	-620	Speed pot or STEP not pressed	While the controller was in manual control mode, an V+ program tried to initiate a robot motion, but you failed to press the step button and speed bar on the MCP.	When an V+ program is about to initiate robot motions, press the step button and speed bar on the MCP. To continue the motion once it has started, you can release the step button but must continue to press the speed bar. Set the controller to automatic mode.
172	IE_ROBILK	-621	Robot interlocked	An attempt has been made to access a robot or external device that is already being used by a different program task or by the system monitor.  An attempt has been made to calibrate the robot with the controller in manual mode, which is not allowed for safety reasons.	Review the program logic and make sure the robot or device is being controlled by only one program task.  Change the Front Panel key switch to the automatic mode position or set the user manual mode signals appropriately.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
173	IE_NOROBT	-622	No robot connected to system	An attempt has been made to attach a robot on a system that does not have the robot configured.	<p>Some commands implicitly attach the robot.</p> <p>Make sure the system has corresponding robots configured.</p> <p>Make sure the system has been booted from the correct system disk (use the ID command to display the system identification, for example).</p> <p>Change the program so that it does not attempt to attach a robot that is not configured.</p>
174	IE_FPBSOF	-623	Stop-on-force triggered	A force-sensor Guarded Mode trip occurred when the robot was not under program control.	High Power must be re-enabled before robot motion may continue. If the trip was not desired, make sure that Guarded Mode is disabled before the program relinquishes control of the robot to the manual control pendant.
175	IE_FPBPRT	-624	Force protect limit exceeded	At least one force-sensor strain gauge reading has exceeded the preset limit, causing a robot panic stop. This may happen due to high forces experienced during an insertion, a crash, or high acceleration.	If a crash occurred, ensure that the work area is cleared. If the limit was exceeded in normal operation, the limit should be increased or Protect mode should be disabled. Enable high power with the pendant and continue operation.
176	IE_INVSDT	-625	Invalid servo initialization data	During V+ system initialization after booting from	Make sure your system disk has been configured correctly.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				disk, servo initialization data in the wrong format was found.	
177	IE_CTBLAL	-626	Can't ALTER and track belt	Either a belt-relative motion was specified while ALTER mode was enabled, or an attempt was made to enable ALTER mode while the selected robot was tracking a belt. Both operations are prohibited because belt-tracking and ALTER mode cannot be performed at the same time.	Either disable ALTER mode or stop tracking the belt.
178	IE_PONERR	-627	Robot power on	An attempt has been made to perform an action that requires High Power to be OFF.	DISABLE POWER and reexecute the action.
179	IE_MODERR	-628	*Robot module not loaded* X	This error occurs only during startup when a robot module has been configured using the CONFIG_C utility, but the robot module is not present in memory.	Use the CONFIG_C utility to add the robot module to the boot disk before rebooting.
180	IE_SYFERR	-629	SYSFAIL detected by CPU	A board on the VMEbus has encountered a severe error and asserted the SYSFAIL sig-	Check the SYSFAIL LEDs on the front edge of the boards. The board which has failed should light its LED.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				nal which turns off HIGH POWER. The watchdog timers on the CPU boards assert this signal and light the SF LED if severe software errors occur.	Restart the system. Verify proper seating of the system boards and correct device connections to the boards. Test the system with as many boards removed as possible, adding boards back in until the problem board is identified. If the problem persists, contact your local OMRON representative.
181	IE_ESIERR	-630	Backplane E-STOP detected by CPU	The system has detected an error or problem and has asserted the BRKSTOP signal on the VME bus. If that error is seen, it indicates a transient BRAKEESTOP signal.	Correct the problem that is causing the motion system to report the error.
182	IE_CTLHOT	-631	Controller overheating	Contact your local OMRON representative.	
183	IE_ROBPWF	-632	Power failure detected by robot	Indicates that a robot amplifier has detected an under-voltage or over-voltage of its internal DC Bus.	This may occur if the High Power supply is too high or too low. Verify the voltage on the High Power supply. This may occur if a motion is attempted that is too aggressive for the rated payload of the robot. Lower the speed, acceleration or deceleration of the robot. Reduce the payload if possible.
184	IE_PANCMD	-633	PANIC command	You have entered an V+ PANIC monitor command, or a	To continue the current motion, enter the RETRY monitor command. To continue after the current

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				program has executed a PANIC program command which has stopped the current robot motion. High Power is still enabled.	motion, enter the PROCEED monitor command.
185	IE_FRCCER	-634	Force sensor communication error	Contact your local OMRON representative.	
186	IE_NOCTCL	-635	Cartesian control of robot not possible	A program has attempted to perform a straight-line motion with a robot that does not support such motions.	Change the program to use joint-interpolated motion.
187	IE_HSTOVRE	-636	overrun detected by host system	<p>One of these three conditions has occurred:</p> <p>The time for a new trajectory point has arrived, but the internal trajectory task has not finished computing the previous point.</p> <p>The servos did not receive trajectory data at the expected time because the trajectory task took too long to compute and write out the data</p> <p>The trajectory interval is equal to or less than the servo interval.</p>	<p>Perform one or more of the following:</p> <p>If the trajectory cycle time is less than 16 msec, change it to the next longer time</p> <p>Move servo tasks off CPU #1 to allow more time for trajectory generation.</p> <p>Reduce the number of robots or axes that you are operating.</p> <p>If the trajectory cycle time is set to 2 ms, make sure the servo interval is 1 ms.</p>
188	IE_SPNERR	-637	Illegal while joints SPIN'ing	An attempt has been made to	Stop the SPIN trajectory with a SPIN or

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				execute a regular motion command while a SPIN trajectory is being executed.	BRAKE command before executing a regular motion command.
189	IE_CNTSPN	-638	SPIN motion not permitted	Either a SPIN command has attempted to move a joint that has not been configured with the continuous-rotation capability or the robot is currently tracking a belt or moving under control of an ALTER command.	Configure the joint with continuous-rotation capability, or complete the belt tracking or ALTER command before attempting to execute the SPIN command.
190	IE_MNBRAK	-639	Manual brake release	The robot's manual brake-release button is active. It is not possible to enable power when this button is pressed.	Make sure that the manual brake-release button (usually located on the robot) is not active. If the problem persists even though the button is not pressed, contact your local OMRON representative.
191	IE_ESTROB	-640	E-STOP from robot	The motion interface board has detected an Emergency Stop condition generated by the RSC in the robot. This error is probably due to low air pressure, joint-1 overtravel, or motor overheating. A subsequent error message may provide more information.	Check for a subsequent message. To determine if there was an unreported RSC error, use the LISTR keyword. If no additional information is available, check for low air pressure, joint 1 overtravel, or motor overheating.
192	IE_ESTAMP	-641	E-STOP from amplifier	The system has	Check for a sub-

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				detected an Emergency Stop condition generated by the motor amplifiers. It indicates that the amplifiers have detected some fault condition.	sequent message. To determine if there was an unreported RSC error, use the LISTR monitor command.
193	IE_ESTSYF	-642	SYSFAIL detected by robot	The motion interface board has detected a SYSFAIL signal on the VMEbus and has asserted the backplane E-Stop signal. This error is normally superseded by other errors and not seen.	Check for a subsequent message. To determine if there was an unreported RSC error, type listr error (task,4), where task is the number of the task that received the error.  If no additional information is available, contact your local OMRON representative.
194	IE_ESTBAC	-643	E-STOP detected by robot	The motion interface board has detected an Emergency Stop due to the BRAKE-ESTOP signal being asserted on the VMEbus.	Check for a subsequent message. To determine if there was an unreported RSC error, use the LISTR monitor command. If no additional information is available, contact your local OMRON representative.
195	IE_MOVING	-644	Illegal while robot is moving	The robot and controller do not have the same safety options.	Make sure that the correct robot and controller are being used together. Install (or remove) the appropriate EN954 Safety Category license in the controller.
196	IE_PWDSBL	-645	Power disabled: Manual/Auto changed	V+ disables power when the Controller Interface Panel (CIP) switch moves from manual to auto or vice versa.	Use any valid method to enable High Power.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
197	IE_PWENBL	-646	HIGH POWER button not pressed	You did not press the High Power on/off button before the timeout period expired.	If working from the keyboard, reissue the enable power monitor command and promptly press the High Power on/off button when instructed to do so. If working from the MCP, follow the procedure appropriate for enabling High Power for the safety category of your system. Promptly press the High Power on/off button when instructed to do so. If the timeout period is too short, adjust it. This message also can result from a faulty connection.
198	IE_EC3DED	-649	Timeout: MCP enable switch not toggled	V+ did not enable High Power because you failed to properly toggle the MCP enable switch on the manual control pendant.	Do one or more of the following:  When toggling the MCP enable switch, release it for a minimum of two seconds and a maximum of ten seconds, and then press it back in.  Make sure that you are pressing the MCP enable switch and not the run/hold button by mistake.
199	IE_TPFAIL	-650	Manual control pendant failure	A program has attempted to access the pendant when it is disconnected or has failed.	Make sure the pendant is connected properly.  If the problem persists, contact your local OMRON representative.
200	IE_RSCDWN	-651	RSC communications failure	V+ has lost communications with the robot signature card (RSC). Either a hardware prob-	Check the connections of the robot cables. Turn High Power ON, calibrate the robot, and resume program execution.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				lem has occurred or the robot is being operated in an environment with excessive electrical noise.	If the problem persists, contact your local OMRON representative.
201	IE_RSCRST	-652	RSC reset	V+ has detected that the robot signature card (RSC) has lost power temporarily, but is now functioning.	Turn high power back on and resume program execution. If the problem persists, check the cabling to the robot.  If the problem persists, contact your local OMRON representative.
202	IE_RSCTMO	-653	RSC time-out	V+ has not received a response from the robot signature card (RSC) when expected, during the initial calibration data load. The RSC or its cabling is probably faulty.	Power OFF the controller and check the cables to the robot.  If the problem persists, contact your local OMRON representative.
203	IE_RSCTER	-654	RSC transmission garbled	V+ has received an invalid transmission from the robot signature card (RSC).  Either a hardware problem has occurred or the robot is being operated in an environment with excessive electrical noise.	None unless the calibration load fails or RSC communications fail.  If the problem persists, contact your local OMRON representative.
204	IE_RSCBAD	-655	RSC bad packet format		
205	IE_RSCNCL	-656	RSC calibration load failure	V+ cannot load calibration data from the robot	Power OFF the controller and make sure the robot cables are

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				signature card (RSC).	correctly and securely connected.
206	IE_PENCON	-657	Pendant not connected	The pendant is not connected to the XMCP connector or it is not responding.	Check the connection, cable, and pendant.
207	IE_HDWNPR	-658	Device hardware not present	An attempt has been made to reference a device that is not present in your system.	Verify that the device was correctly specified. Verify that the device hardware is present and is configured properly.
208	IE_IEDERR	-660	Device error	An error was detected for an external device such as one specified in the last SETDEVICE program command. The actual error depends upon the type of device referenced.	Make sure the program command parameters are valid. Refer to the documentation for the device type referenced for information on how to determine what has caused the error.
209	IE_EEPCHK	-661	NVRAM data invalid	The nonvolatile RAM has not been initialized or the data has been corrupted.	Restart the controller. If the problem persists, contact your local OMRON representative.
210	IE_DEVSER	-662	Device sensor error	A hardware error occurred in the sensing system accessed with the last DEVICE program command.	Refer to the documentation for more information.
211	IE_IEDRES	-663	Device reset	The device is busy processing a reset operation. The reset can have been requested with a SETDEVICE program command by	Use software interlocks to prevent a second program task from accessing the device after a reset operation has been requested. Note that the requesting SETDEVICE program

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				another program task that is accessing the device, or the device can have initiated the reset on its own.	command waits for the reset to complete. Refer to the documentation for the specific device for information on its self-generated resets.
212	IE_INVCAL	-664	Invalid calibration data	Contact your local OMRON representative.	
213	IE_EEPBAT	-665	NVRAM battery failure	The nonvolatile RAM battery backup has failed and the RAM may not hold valid data.	Replace the NVRAM battery.
214	IE_NOTSYS	-666	Must use CPU #1	A command that requires execution on CPU #1 has been attempted on a different CPU.	Reexecute the command or instruction on CPU #1.
215	IE_PWFERR	-667	Power failure detected	Indicates that a controller High Power failure condition has been detected. If battery backup is installed, this error is reported (when power is restored) by any I/O operations that were canceled due to the power failure.	This error code may be trapped by a program using the REACTE command in order to provide some level of automatic power failure response.  You may need to restart or repeat any operations that were interrupted by the controller High Power failure. Some reinitialization of the system may be required. For example, any robot(s) connected to the controller must be recalibrated after a controller power failure.
216	IE_DEVINU	-668	Device in use	An attempt has been made to attach, assign, or configure a	Check the program code to make sure the requested device has not already been attached.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				hardware device (e.g., a VMI axis) that is already being used.	
217	IE_ERSCHW	-669	RSC hardware failure	The RSC has reported an internal failure.	Because RSC failures almost always cause the RSC to stop communicating altogether (rendering it incapable of reporting the failure), this error message may be due to some other cause, such as electrical noise at the RSC or within or around the arm signal cable.  If the problem persists, contact your local OMRON representative.
218	IE_ERSCPW	-670	RSC power failure	The RSC has reported that its power is failing.	If the problem persists, contact your local OMRON representative.
219	IE_ESR12F	-671	Servo board 12V fuse open	Your servo board has a fused 12-volt bus, and the system has detected an open circuit at that location.	Replace the fuse.
220	IE_ESRSLF	-672	Servo board solenoid fuse open	Your servo board has a fused robot solenoid control line, and the system has detected an open circuit at that location.	Replace the fuse.
221	IE_ESRESF	-673	Servo board E-STOP fuse open	Your servo board has a fused E-STOP circuit, and the system has detected an	Contact your local OMRON representative for details about types and locations of fuses, and replace the fuse.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				open circuit at that location.	
222	IE_SEROVR	-674	Servo task overloaded	A servo interrupt task has used up all the execution time. The detection algorithm reports an error when the servo interrupt task completely occupies 10 or more time slices per second of real time. The robot went to a fatal error state when this error occurred, and the servo interrupt task stopped running.	Change one or more of the following: Move servo tasks off CPU #1 to allow more time for trajectory generation. Reduce the number of robots or axes that you are operating.
223	IE_MOTPFW	-675	Timeout enabling power	High Power did not enable within the allowed amount of time, and the servos reported no other error during the timeout period.	Check your installation (cabling, AC power line voltages, circuit breakers, amplifier retaining screws, cables, and contactors). Make sure that the amplifier chassis is properly connected to a power source and is turned ON. If the problem persists, contact your local OMRON representative.
224	IE_MIDWRG	-676	RSC module ID doesn't match robot	The V+ configuration data contains an explicit ID specification for a robot module (for example, 6 for the 550 robot), and the robot RSC does not contain that	Make sure that the correct type of robot is being used. Use the CONFIG_C utility to change the module ID to -1 in the V+ configuration data. Correct the module ID in the RSC.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				ID number.	
225	IE_IESVIN	-677	Servo protocol incompatible	During startup, V+ has detected a remote node on the servo network that is incompatible with the current V+ system because incompatible versions of software are being used. The servo network does not operate.	Contact your local OMRON representative.
226	IE_IEDUPS	-678	Duplicate servo node ID	During startup, or after a servo bus reset, V+ has detected two or more servo nodes with the same serial number. Either an error has occurred with servo node detection, or a servo node is configured incorrectly. Servo network operation is not allowed.	Reinitialize your servo network. Reboot your robot controller. If a new servo node has been added to the network, verify that its serial number is correct. If the problem persists, contact your local OMRON representative.
227	IE_IENFSV	-679	Expected servo node not found	After a servo bus reset, V+ has not detected all the required nodes on the servo network. Either the network has failed because of noise, a servo node has failed, or a servo node has been unplugged.	Verify that all servo network connectors are secure. Reinitialize your servo network. Reboot your robot controller.  If the problem persists, contact your local OMRON representative.
228	IE_IESNDN	-680	Servo node not downloaded	V+ has attempted to use a	Reboot V+ and try again.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				<p>servo node that does not contain properly downloaded software. Servo node software is only loaded at V+ startup time. One of the following has happened:</p> <p>You are using a servo node that is not supported by the V+ system software.</p> <p>You have connected a new servo node after V+ was started.</p>	Make sure your software is compatible with your 1394 nodes. Update the 1394 firmware.
229	IE_VINOOB	-704	No objects seen	Contact your local OMRON representative.	
230	IE_VINRUN	-705	Camera not running	Contact your local OMRON representative.	
231	IE_VIRUN	-706	Invalid request while camera running	Contact your local OMRON representative.	
232	IE_VIPROT	-707	Unknown prototype	Contact your local OMRON representative.	
233	IE_VINCAL	-713	Vision not calibrated	Contact your local OMRON representative.	
234	IE_VIONAL	-714	Camera already running	Contact your local OMRON representative.	
235	IE_VIOFAL	-719	Camera already off	Contact your local OMRON representative.	
236	IE_VININS	-720	Vision option not installed	Contact your local OMRON representative.	
237	IE_VINOCI	-722	Camera interface board absent	Contact your local OMRON representative.	
238	IE_VINPPD	-723	No picture data available	Contact your local OMRON representative.	
239	IE_VIILDM	-724	Illegal display mode	Contact your local OMRON representative.	

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
240	IE_VIILCL	-726	Bad camera calibration	Contact your local OMRON representative.	
241	IE_IEVTRN	-729	Invalid request while vision training	Contact your local OMRON representative.	
242	IE_IENINF	-730	Information not available	Contact your local OMRON representative.	
243	IE_IESUBP	-731	Unknown sub-prototype	Contact your local OMRON representative.	
244	IE_VIIVNM	-732	Invalid model name	Contact your local OMRON representative.	
245	IE_VINOMM	-733	Vision system out of memory	Contact your local OMRON representative.	
246	IE_ILVARG	-735	Invalid vision argument	Contact your local OMRON representative.	
247	IE_VABORT	-749	Vision aborted	Contact your local OMRON representative.	
248	IE_NVSSSEL	-751	No vision system selected	Contact your local OMRON representative.	
249	IE_VINTER	-767	Vision internal error	Contact your local OMRON representative.	
250	IE_NOVISOPT	-770	The vision option is not licensed	The Robot Vision Manager license is not enabled on the dongle.	Install the Robot Vision Manager license.
251	IE_ILLASS	-771	Invalid Robot Vision Manager sequence	The vision sequence number in the Robot Vision Manager does not exist.	Correct the Robot Vision Manager sequence in the workspace.
252	IE_ILLAST	-772	Invalid Robot Vision Manager tool index	The index of the vision tool in the V+ vision command call does not exist. The Robot Vision Manager vision tool index is not valid.	Check the sequence and the V+ code.
253	IE_ILLASP	-773	Invalid Robot Vision Manager parameter ID	The Robot Vision Manager parameter that V+ is trying to set or get does	Correct the V+ code.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				not exist on ACE. The parameter code specified is not valid for the tool being referenced.	
254	IE_ILLASP2	-774	Invalid Robot Vision Manager parameter index	The Robot Vision Manager parameter that V+ is trying to set is in a parameter array. The array pointed to is valid, but the index is not.	Correct the V+ code.
255	IE_NOASSEQ	-775	Robot Vision Manager instance not found	The index of the specified instance does not exist.	Correct the V+ code.
256	IE_NOCAMIM	-776	Unable to acquire an image from the camera	Robot Vision Manager is unable to acquire an image.	Verify that camera is connected correctly. Make sure the drivers are working properly. Change the camera if necessary.
257	IE_NOROBLAT	-777	Unable to read the robot position latch	The robot position has not been latched.	Verify the latch signal cabling. Verify the latch signal configuration in V+.
258	IE_NOBELLAT	-778	No belt latch was detected	The belt has not been latched.	Verify the latch signal cabling. Verify the latch signal configuration in V+.
259	IE_UNKNER	-800	Unknown error code	An error code that does not correspond to a known error message was received by V+ from an external device.	If an external computer is communicating with V+ when the error occurs, verify that it is sending proper error codes. Otherwise, a software error is indicated.  Contact your local OMRON support with details leading up to this error.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
260	IE_BDVFEA	-801	Invalid VFEATURE access	Contact your local OMRON representative.	
261	IE_NCMARM	-802	Invalid camera calibration		
262	IE_ILLCAM	-803	Illegal camera number		
263	IE_NOOPT	-804	Option not installed	An attempt has been made to use a feature of a V+ system option that is not present in this robot system.	Restart the controller.  If the problem persists, contact your local OMRON representative.
264	IE_NOHDWR	-805	Hardware not in system	A command has attempted to access optional hardware (such as a FORCE board) that is not installed in the system.	Install the needed hardware or remove the command that addresses the hardware.
265	IE_DBERR	-859	Database manager internal error	This error indicates that the system has encountered an inconsistency.	Contact your local OMRON support with details leading up to this error.
266	IE_COLDET	-901	Obstacle collision detected	A possible or actual collision has been detected between the robot and a statically defined obstacle.	Obstacles may include fixed objects in the workcell as well as structural elements of the robot, such as its base. This error is similar to *Location out of range* in that it is often detected by the kinematic solution programs as the robot is moving.  Move the robot away from the obstacle and continue the motion or modify the executing application program to avoid the obstacle. For application programs, this

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
					error may indicate that either the planned end point of the motion collides with an object or that a collision has been detected in the middle of a straight-line motion.
267	IE_MSECME	-902	Interrupted multi-segment motion	The only multiple-segment motions are MOVEF and MOVESF. For these motion commands this error is signaled if the DEPART motion has been initiated, but neither the APPROACH nor the final move to the destination is performed. For example, this situation might occur if an envelope error is detected during the DEPART or the APPROACH motion segments.	Correct the problem that caused the robot motion to terminate prematurely and re-execute or skip the multiple-segment motion.
268	IE_ESTUCK	-904	[Fatal] E-STOP signals are stuck off	During system startup, a test is performed to ensure that no E-STOP signals are stuck in the OFF state. This error message is followed by one or more standard E-STOP error messages that indicate which signals are stuck. If this error occurs,	Check the wiring of your Emergency Stop circuits. Verify that the Controller Interface Panel (CIP) is connected properly. If the problem persists, contact your local OMRON representative for assistance.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				robot power cannot be enabled.	
269	IE_NOHPWR	-906	Robot power off requested	High Power has been turned OFF because of a program or user request, such as issuing a DISABLE POWER command.	This is an informational message only.
270	IE_ESTPWR	-907	E-STOP circuit relay failure	An Emergency Stop condition has occurred because the E-STOP circuit on one channel is not in the same state as the other channel.	An Emergency Stop condition has occurred because the E-STOP circuit on one channel is not in the same state as the other channel. This can be the result of fused/failed contacts a device, a short circuit in one of the E-STOP channels, or the result of a pair of contacts connected to the E-STOP channels being in the opposite state (one open and the other closed).
271	IE_ESTFP	-908	E-STOP from front panel button	An Emergency Stop condition has occurred from pressing the E-Stop button on the Front Panel.	Release the Front Panel E-Stop button.
272	IE_ESTMCP	-909	E-STOP from MCP E-STOP button	An Emergency Stop condition has occurred because the E-Stop button on the Pendant has been pressed.	Reset the E-STOP button and then re-enable High Power when ready.
273	IE_CP44P3	-910	E-STOP from user E-STOP button	An Emergency Stop condition has occurred because the user E-STOP button circuit continuity was	Reset the user E-STOP button circuit and then enable High Power when ready.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				lost.	
274	IE_CP44P5	-912	E-STOP from user enable switch	An Emergency Stop condition has occurred because the user enable switch has been released. During Manual mode, releasing this switch performs a controlled power OFF rather than an Emergency Stop.	Hold the user enable switch and re-enable High Power as desired. If the switch was already pressed, check the switch and the associated connectors and wiring.
275	IE_ESTCPU	-919	E-STOP asserted by CPU	An Emergency Stop condition has been generated by V+ in response to an internal error condition.	If safe to do so, attempt to enable High Power and note any different error messages that occur. If this error persists, contact your local OMRON representative.
276	IE_AMSTUK	-920	[Fatal] Auto mode switch stuck on	During system initialization, a hardware test of the Manual mode circuit has found that the key switch is stuck in Automatic mode. This error indicates a safety hazard and prevents High Power from being enabled.	Check any Manual mode switch that may be in use. Verify that the CIP is connected securely. Restart your V+ system to clear the error and repeat the test. If the problem persists, contact your local OMRON representative.
277	IE_CP44P7	-921	E-STOP from user muted safety gate	An Emergency Stop condition has occurred because the user muted safety gate has been opened during automatic mode. During Manual mode, this error	Close the muted safety gate and re-enable High Power when ready. If the gate was already closed, check the switch, the associated connectors, and wiring.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				should not occur.	
278	IE_ESTINC	-922	E-STOP channels 1 and 2 do not match	<p>An Emergency Stop condition has occurred because the two redundant E-STOP signal channels do not report the same E-STOP state.</p> <p>This error can be caused if redundant E-STOP channels do not match each other within a reasonable time (0.5 seconds).</p>	Verify that both channels are wired and functioning properly.
279	IE_ESTSHR	-923	E-STOP circuit is shorted	An Emergency Stop condition has occurred because a short circuit in the E-STOP wiring has been detected.	Verify that all Emergency Stop channels are wired and functioning properly and that there is no external power source wired to the E-STOP circuit. Check that the E-STOP source jumper on any connected robot is on the EXT position. If the problem persists, contact your local OMRON representative.
280	IE_BADLIT	-924	Front panel HIGH POWER lamp failure	High Power has been disabled because a failure (Open Circuit) in the front panel High Power lamp has been detected. The lamp has probably failed. This condition is considered a safety hazard and an Emergency Stop is	Replace the High Power indicator lamp and enable High Power when ready.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				not signaled. However, High Power cannot be enabled until the lamp is replaced.	
281	IE_CP48P7	-929	E-STOP from Line E-STOP input	An Emergency Stop condition has occurred because it was requested through the Line E-STOP input signal.	Restore the state of the Line E-STOP input signal.
282	IE_PWROVLD	-933	Power output overloaded on Belt/EXPIO	Contact your local OMRON representative.	
283	IE_HPSTUK	-934	[Fatal] HIGH POWER button stuck on		
284	IE_NOSOL1	-935	Orientation out of range	While evaluating the kinematic solution, the deflection of the moving platform of the parallel robot was found to be beyond the allowable extreme.	Press the Brake Release button and manually move the robot joints so that the moving platform is nearer to its square shape.
285	IE_NOSOL2	-936	Kinematic solution not found	While evaluating the kinematic solution, the current joint angles of a parallel robot do not form a consistent set.	Modify the specified destination location (e.g., if a precision point is specified). Press the Brake Release button, and manually move the robot to a different location.  If the problem persists, contact your local OMRON representative.
286	IE_BADKEY	-937	Robot already under manual control	An attempt has been made to perform joint control of a robot for which it is not allowed.	Select a different mode of control. On the pendant you can use WORLD or TOOL mode.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				Joint control of parallel robots are not allowed, for example.	
287	IE_NOJNT	-938	Joint control of robot not possible	An attempt has been made to control the robot from the ACE virtual pendant, or a JOG program command or Monitor command has been issued while the robot is already being controlled from the pendant.	Use COMP mode from the pendant.
288	IE_ESTOPUNSTABLE	-939	E-STOP unstable	The E-STOP source has not been resolved.	Check for electrical disturbances or intermittent connections on the Emergency Stop circuit.
289	IE_ABORTX	-999	Aborted	Contact your local OMRON representative.	
290	IE_UNKSVE	-1001	Invalid servo error	An unrecognized error was reported for the indicated robot motor.	Attempt the operation again. If the problem persists, contact your local OMRON representative.
291	IE_NOSOL	-1002	Position out of range	The requested motion was beyond the software-limited range of motion for the indicated joint.  While enabling High Power, V+ detected that the indicated robot joint was outside the software limit.	Modify the program as required to prevent the invalid motion request because the robot did not actually move out of range. You do not need to move the robot before continuing.  Move the robot back into the working envelope.  Correct whatever caused the robot to get into the restricted area and then enable

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
					High Power.
292	IE_NOTIME	-1003	Time-out nulling errors	The indicated motor took too long to complete the last motion, possibly because the robot is blocked and cannot reach its destination.	Retry the motion after making any necessary program changes. If this error occurs repeatedly, contact your local OMRON representative for assistance.
293	IE_NOREF	-1004	No zero index	The motor controller did not detect a zero-index mark for the indicated joint.	Before you can resume running the program, you must recalibrate the robot.  If the problem persists, contact your local OMRON representative.
294	IE_NOSYNC	-1005	Unexpected zero index	A zero index signal was received from the encoder for this motor at an unexpected time. The encoder may be gaining or losing counts, there may be a hardware problem with the zero index signal, or the Counts per zero index configuration parameter may be set incorrectly.	Turn on High Power, calibrate the robot, and continue to use the system. If this error occurs repeatedly, contact your local OMRON representative.
295	IE_EVLERR	-1006	Soft envelope error	The indicated motor was not tracking the commanded position with sufficient accuracy, indicating a failure in the hardware servo system or something impeding	Try to perform the motion at a slower speed. Make sure nothing is obstructing the robot's motion.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				the path of the robot. Because this was not considered a serious error, a controlled motion stop occurred and high power remains ON.	
296	IE_MSTALL	-1007	Motor stalled	The indicated motor has stalled while the robot was moving.	This is usually caused by the robot encountering an obstruction. Turn High Power ON and restart the program. Remove the obstruction or modify the program to have the robot follow a different path.
297	IE_QADERR	-1008	Encoder quadrature error Mtr X	The position encoder signal from the specified motor is sending information that is not phased correctly. The encoder or its cabling may be defective.	Turn on high power, calibrate the robot, and try to perform the motion at a slower speed. If the error persists, contact your local OMRON representative.
298	IE_AMPERR	-1009	Timeout enabling amplifier	The power amplifier for the indicated motor has signaled a fault condition. A momentary power failure or a hardware error may have occurred.	Turn High Power ON and restart the program. If the error persists, contact your local OMRON representative.
299	IE_MOTHOT	-1016	Motor overheating	The indicated motor is overheating.	Reduce the speed, acceleration, and/or deceleration of the robot motions, or introduce delays in the application cycle to give the motor an opportunity to cool.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
300	IE_AMPER1	-1018	Motor amplifier fault	The power amplifier for the indicated motor has signaled a fault condition on fault line 1.	Turn High Power ON and restart the program.  If the problem persists, contact your local OMRON representative.
301	IE_DUTCYC	-1021	Duty-cycle exceeded Mtr X	The indicated motor has been driven fast for too long a period of time. The servo system has disabled High Power to protect the robot hardware.	Turn ON High Power, reduce the speed and/or acceleration for the motion that was in progress or for motions that preceded that motion and repeat the motion that failed.
302	IE_SKWERR	-1022	Skew envelope error	The two motors associated with a split robot axis were not tracking each other with sufficient accuracy.	Make sure nothing is obstructing the robot motion. Turn High Power ON and try to perform the motion at a slower speed. If necessary, use the SPEC utility to increase the maximum skew error.
303	IE_NOSOLM	-1023	Position out of range, motor	The requested motion was beyond the software-limited range of motion for the indicated motor.  While enabling High Power, V+ detected that the indicated robot motor was outside the software limit.	Modify the program as required to prevent the invalid motion request. Because the robot did not actually move out of range, you do not need to move the robot before continuing.  Move the robot back into the working envelope. Correct whatever caused the robot to get into the restricted area, and then enable High Power.
304	IE_ENCFLT	-1025	Encoder fault Mtr X	The servo board has detected a broken encoder wire on the indicated axis.	Inspect the encoder wiring for intermittent connections or broken wires. Try replacing the encoder cable with

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
					another working cable.
305	IE_SWINC	-1026	Software incompatible	The servo code has detected an incompatibility between the servo code and calibration software.	Make sure that you are using the calibration software (in the \CALIB\ directory) that you received with the V+ system you are using. If you are using the correct software, note the code number, and contact your local OMRON representative.
306	IE_HEVLER	-1027	Hard envelope error	The indicated motor was not tracking the commanded position with sufficient accuracy, indicating a failure in the hardware servo system or something impeding the path of the robot. Because this is considered a serious error, High Power was turned OFF.	Turn on High Power and try to perform the motion at a slower speed. Make sure that nothing is obstructing the robot's motion. If the error recurs, contact your local OMRON representative.
307	IE_SSPDER	-1028	Soft overspeed error	The servos have detected an attempt to move a robot axis at a speed or acceleration that is higher than allowed. The motion is terminated but robot power remains enabled.  When the new motion request implies excessive velocity or acceleration in	Move the robot at a slower speed. If you are near a mechanical singularity, move the robot using joint mode instead of world or tool mode.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				more than one servo / axis, only one error is generated.	
308	IE_HSPDER	-1029	Hard overspeed error	During Manual mode, the safety hardware has detected an attempt to move a robot axis at a speed faster than allowed. The motion is terminated and robot power is disabled. This error should never occur if the servos are properly configured.	Verify that the servos for this motor are properly configured. If the problem persists, contact your local OMRON representative.
309	IE_NGOVER	-1032	Negative overtravel	The negative hardware overtravel switch for the indicated motor has been tripped. Robot power has been disabled.	Manually move the robot back into range and re-enable power. Check the the negative overtravel switch and its wiring. Adjust the robot soft limits so they are inside the hard limits.
310	IE_PSOVER	-1033	Positive overtravel	The positive hardware overtravel switch for the indicated motor has been tripped. Robot power has been disabled.	Manually move the robot back into range and re-enable power. Check the the negative overtravel switch and its wiring. Adjust the robot soft limits so they are inside the hard limits.
311	IE_RBOVER	-1034	Overtravel	The indicated motor has moved beyond the hardware-limited range of motion.	Manually move the robot back into range and re-enable power. Check the overtravel switch and its wiring. Use the CONFIG_C utility to adjust the soft limits for the robot so they are inside hard limits.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
312	IE_SVDEAD	-1104	[Fatal] Servo dead Mtr	The servo process for the indicated motor is not responding to commands from V+. V+ continues to operate, but does not allow High Power to be turned ON.	Power OFF the controller and restart. If the problem persists, contact your local OMRON representative.
313	IE_NOCTRK	-1106	Calibration sensor failure Mtr X	During calibration, the calibration sensor for the indicated motor cannot be read correctly. Either the robot is blocked from moving, or a hardware error has occurred.	Retry the CALIBRATE program command or monitor command after making sure that the robot is not blocked. If the problem persists, contact your local OMRON representative.
314	IE_SAFETYSYS TEMFAULT	-1109	Safety System Fault	Because the status codes are related to hardware, refer to your robot hardware documentation as your primary source of information for correct hardware and safety system setup. You should also re-commission the robot.	If one of these message codes occurs, attempt to enable power again while observing safety precautions. If the same error code occurs again, there may be a fault with the sensor.  If the problem persists, contact your local OMRON representative.
315	IE_EC3EEP	-1111	E-STOP from safety system Code X	An Emergency Stop event has occurred. Use the following information to understand the source.  0: Channel 1 input 1: Channel 2 input	Check safety device status and connections. Refer to the robot user's manual for more information.

Message Number	Error in Code	Error ID	Message	Description	Corrective Action
				2: Line 1 input 3: Line 2 input	
316	IE_ POWERSYS TEMFAILUR- E	-1115	Power System Failure	The servo interface has detected a fault in the robot power system. The code number provides more detailed information about the fault.	Check the servo and amplifier cabling. Power OFF and restart your controller and servo nodes. If the problem persists, contact your local OMRON representative.
317	ENDMARKE- R	-3000	Unknown error code	Contact your local OMRON representative.	

**OMRON Corporation** Industrial Automation Company  
Kyoto, JAPAN

Contact: [www.ia.omron.com](http://www.ia.omron.com)

**Regional Headquarters**

**OMRON EUROPE B.V.**

Wegalaan 67-69, 2132 JD Hoofddorp  
The Netherlands  
Tel: (31)2356-81-300/Fax: (31)2356-81-388

**OMRON ASIA PACIFIC PTE. LTD.**

No. 438A Alexandra Road # 05-05/08 (Lobby 2),  
Alexandra Technopark,  
Singapore 119967  
Tel: (65) 6835-3011/Fax: (65) 6835-2711

**OMRON ELECTRONICS LLC**

2895 Greenspoint Parkway, Suite 200 Hoffman Estates,  
IL 60169 U.S.A.  
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

**OMRON ROBOTICS AND SAFETY TECHNOLOGIES, INC.**

4225 Hacienda Drive, Pleasanton, CA 94588 U.S.A  
Tel: (1) 925-245-3400/Fax: (1) 925-960-0590

**OMRON (CHINA) CO., LTD.**

Room 2211, Bank of China Tower, 200 Yin Cheng Zhong Road,  
PuDong New Area, Shanghai, 200120, China  
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

© OMRON Corporation 2022 All Rights Reserved. In the interest of product improvement, specifications are subject to change without notice.

Cat. No. I671-E-01

0122

28320-000 A