FANUC PMC-MODEL SD7

PROGRAMMING MANUAL

B-63823EN/02

DEFINITION OF WARNING, CAUTION, AND NOTE

This manual includes safety precautions for protecting the user and preventing damage to the machine. Precautions are classified into Warning and Caution according to their bearing on safety. Also, supplementary information is described as a Note. Read the Warning, Caution, and Note thoroughly before attempting to use the machine.

> ⚠ **WARNING**
> Applied when there is a danger of the user being injured or when there is a damage of both the user being injured and the equipment being damaged if the approved procedure is not observed.

> ⚠ **CAUTION**
> Applied when there is a danger of the equipment being damaged, if the approved procedure is not observed.

> **NOTE**
> The Note is used to indicate supplementary information other than Warning and Caution.

* Read this manual carefully, and store it in a safe place.

PREFACE

This manual covers the specifications and the instructions and operations used for programming with the following devices.

Product name	Applicable CNC
FANUC PMC-MODEL SD7 (PMC-SD7)	FANUC Series 16*i*-MODEL B (Series 16*i*-B) FANUC Series 160*i*-MODEL B (Series 160*i*-B) FANUC Series 18*i*-MODEL B (Series 18*i*-B) FANUC Series 180*i*-MODEL B (Series 180*i*-B) FANUC Series 21*i*-MODEL B (Series 21*i*-B) FANUC Series 210*i*-MODEL B (Series 210*i*-B)

The FANUC PMC (Programmable Machine Controller) is a sequence control unit that is built into the FANUC CNC. It is used to execute IEC61131-3-compatible sequence programs.

IEC61131, as established by the International Electrotechnical Commission (IEC), is a set of rules consisting of the following five parts:

Part 1: General information
Part 2: Equipment requirements and tests
Part 3: Programming languages
Part 4: User guidelines
Part 5: Communications
Part 6: Reserved
Part 7: Fuzzy-control programming
Part 8: Guidelines for the application and implementation of programming languages for programmable controllers

IEC61131-3 relates to Part 3 of these rules. It is intended to specify a world-wide standard programming language and related techniques for industrial sequence control units.

Related Manuals

The following manuals provide an explanation of the use of the above hardware and software, as well as the connection interface. Refer to the corresponding manual as necessary.

Manual Name	Specification Number
FANUC Series 16*i*/18*i*/21*i*/160*i*/180*i*/210*i*-MODEL B Connection Manual (Function)	B-63523EN-1

This product can be programmed using the following software. This software provides with a lot of help documents which contain the information of PMC-SD7. Please also refer to these help documents.

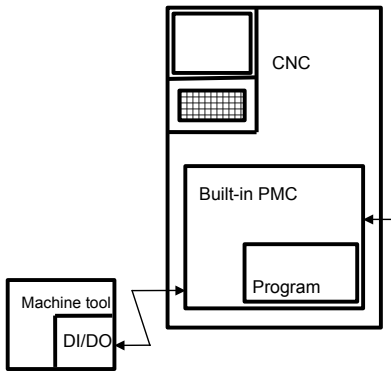
Software
FANUC LADDER-IIIC

TABLE OF CONTENTS

1 OVERVIEW

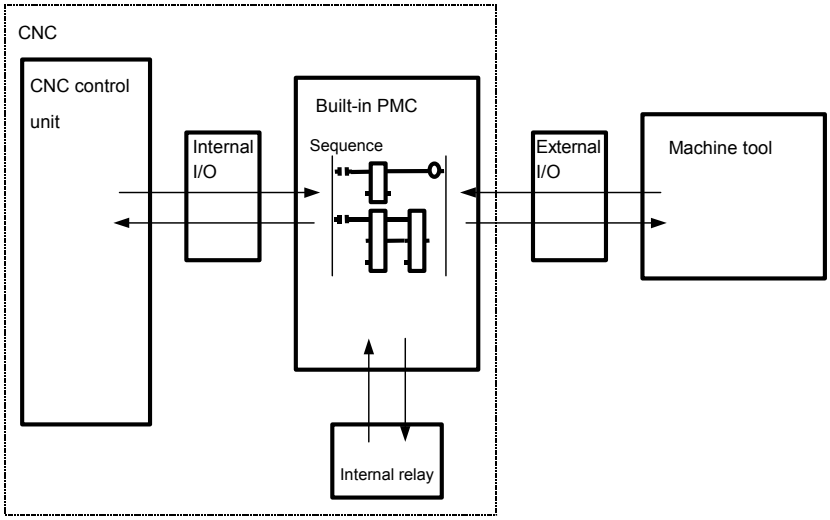
1.1 SYSTEM CONFIGURATION

The PMC-SD7 is used to create and execute sequence programs conforming to the IEC61131-3 standard. The system requires the following hardware and software (development software, etc.).

<table>
<tr>
<td> 
</td>
<td></td>
<td>Hardware</td>
</tr>
<tr>
<td>• Sequence program start/stop
• Communication service (up- and down-loading, online monitor)
• Reference table display/setting
• Writing of sequence program into flash memory</td>
<td>• Sequence program editing, compiling, down-/up-loading, start/stop
• Online monitor
• Reference table display/setting
• Writing of sequence program into flash memory
• Printing</td>
<td>Function</td>
</tr>
<tr>
<td>FANUC PMC control software</td>
<td>FANUC LADDER-IIIC.</td>
<td>Software</td>
</tr>
</table>

1.2 PMC BASIC CONFIGURATION

The basic configuration of the PMC is as shown below.



There are two types of PMC:

- Those using programming that conforms with IEC61131-3, as described in this manual
- Those using existing FANUC proprietary instructions and related items

For example, in the case of the Series 16i-B, the former is represented by the PMC-SD7, while the latter is represented by the PMC-SB7

For these two types of PMC, the mapping and functions of the interface signals between the CNC or machine tool and the PMC are identical, and correspond on a one-to-one basis. Upon reference from a PMC sequence program, only the address or instruction declaration changes.

Therefore, upon reference from a CNC, open CNC PC function, or external PMC application, the PMC signal address and so on are converted to those corresponding to the corresponding PMC type.

Example: Emergency stop signal (*ESP)

PMC-SD7		PMC-SB7
%I00069	⟷	X008.4
%QG00069	⟷	G008.4

1.3 CONTENTS OF THIS MANUAL

1. Overview
2. Specifications
   explains the PMC specifications, program structuring, data type, and references.
3. I/O Link
4. The LD Instruction Group
   explains the LD programming instructions that can be used with the PMC.
5. The IL Instruction Group
   explains the IL programming instructions that can be used with the PMC.
6. PMC Operation
   explains the PMC operations that can be performed using the CNC display.
7. WINDOW Functions (Appendix)
8. Differences between PMC-SB7 (Appendix)
9. Alarm message list (Appendix)
10. CHINESE character code, HIRAGANA code, and special code list (Appendix)

2 SPECIFICATIONS

2.1 SPECIFICATIONS OF PMC

The following table lists the specifications of the PMC-SD7.

Type of PMC	PMC-SD7
	Series 16*i*/160*i*/18*i*/180*i*/21*i*/210*i*-B
Programming language	LD (Ladder Diagram) IL(Instruction List)
Number of ladder levels	2
Ladder execution cycle	8 ms
Instruction processing time	0.033 us/step (relay instruction)
Program size	256 KB 512 KB 768 KB
I/O points • I/O Link (Master) Input Output	 Max. 2,048(Note) Max. 2,048(Note)
Programmer	FANUC LADDER-IIIC

NOTE
Maximum of basic input/output points are 1024/1024 points. I/O Link expansion option extends the maximum to 2048/2048 points.

Programming Language

LD and IL are supported as the programming language.

About LD Logic

Ladder Diagram (LD) logic is one of five programming languages specified by the IEC 61131-3 standard. This graphical language is likely the most popular control language in use today. LD logic is represented as a sequence of rungs with a series of instructions placed on each rung. Generally, each rung is executed from left to right as power flows through the rung and subsequent rungs are generally executed from top to bottom. A basic LD instruction set, including arithmetic, logical and program control operations, is supported by PMC-SD7.

About Instruction List Logic

Instruction List (IL) logic is one of five programming languages specified by the IEC 61131-3 standard. This text language is accumulator-based and much like the assembly languages used for programming microprocessors. In its simplest form an IL program proceeds like the following:

1) Load an accumulator with a data value.
2) Perform an operation on the accumulator, saving the result in the accumulator.
3) Store the accumulator's value to memory (a variable).
4) Do another accumulator load or operation, etc.

(Accumulators)
IL logic is accumulator-based, much like the assembler languages used for programming microprocessors. Most IL instructions operate on the content of an accumulator and then store the result of the operation back to the accumulator. Most functions use an accumulator to store the result of an operation. The content of both accumulators is retained until modified by an instruction or function.

Number of Ladder Levels

A sequence program for PMC consists of two parts: 1st level sequence and 2nd level sequence.

Priority of execution

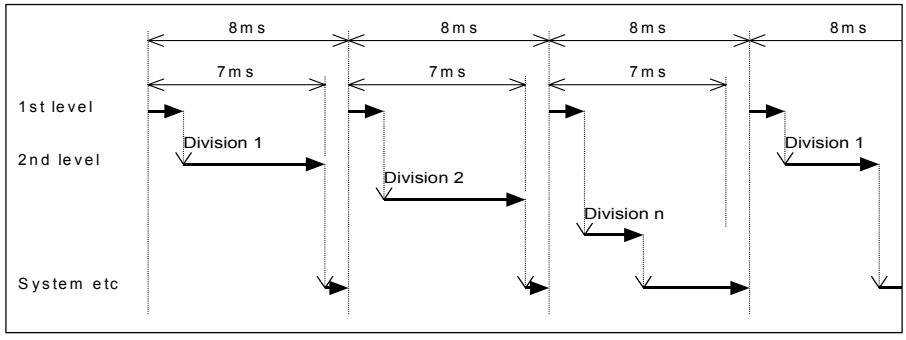


Fig. 2.1 About priority of execution.

The 1st level sequence part operates every 8 ms (high-speed sequential operation).
If the 1st level sequence part is long, the total operating time, including the 2nd level sequence part, is extended. Therefore the 1st level sequence part must be programmed to be processed in as short time as possible.
The 2nd level sequence part operates every 8×n ms. Here n is a dividing number for the 2nd level sequence part.
The 2nd level sequence part is divided automatically when the sequence program is executed. The time for one cycle of the sequence program is displayed on the TITLE screen.

(1) Division of the 2nd level sequence part
The 2nd level sequence part must be divided in order to execute the 1st level sequence part. For example a sequence program is executed in the following sequence when the dividing number is n. (See Fig. 2.1)
After the last division of 2nd level sequence part (division n) is executed, the sequence program is executed again from the beginning. Thus, when the dividing number is n, the cycle of execution is 8msec×n.
The 1st level sequence operates every 8msec, and the 2nd level sequence every 8×n msec. If the steps of the 1st level sequence is increased, the steps of the 2nd level sequence operating within 8msec becomes less, thereby increasing the dividing number and making the processing time longer. Therefore, it is desirable to program so as to reduce the 1st level sequence to a minimum.
In the PMC-SD7, 7msec of 8msec is assigned to execution of the 1st and 2nd level sequences.

(2) 1st level sequence part
Only short-width pulse signals are processed. These signals include emergency stop, over-travel of each axis, reference point return deceleration, external deceleration, skip, measuring position arrival and feed hold signals.

Instruction Processing Time

Relay instructions are processed at high speed, such that it requires 0.033 us to execute one step.

Program Size

A FANUC PMC program is contained in the flash memory of the CNC.
The size of the flash memory varies depending on which option is installed
(ex., 256KB, 512KB, 768KB).

I/O Points

I/O units are connected via the FANUC I/O Link. The maximum number of I/O points is as follows.
Series 16*i*/160*i*/18*i*/180*i*/21*i*/210*i*-B
I/O Link master function: Input 1,024/ Output 1,024
Series 16*i*/160*i*/18*i*/180*i*/21*i*/210*i*-B (With I/O Link expansion option)
I/O Link master function: Input 2,048/ Output 2,048

Programmer

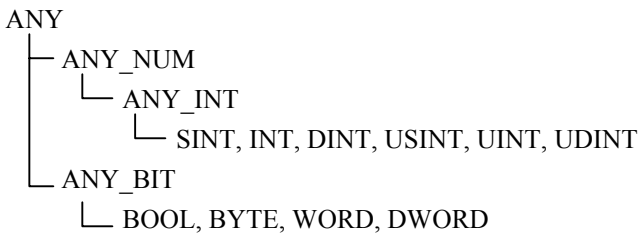
A sequence program is created and edited using a dedicated programmer (FANUC LADDER-IIIC).
A created program is transferred to the PMC and executed.
The programmer supports online operations.

2.2 DATA TYPES

This section explains the data types which can be used with PMC programs.
The PMC can use the basic IEC61131-3 data types shown below.

Keyword	Data type	Bit count	Range
BOOL	Boolean	1	The value obtained with this data type is either 0 (false) or 1 (true).
SINT	8-bit integer	8	Range of values obtained with this data type: $-2^{\text{bit count }-1}$ to $2^{\text{bit count }-1}-1$
INT	16-bit integer	16	
DINT	32-bit integer	32	
USINT	Unsigned 8-bit integer	8	Range of values obtained with this data type: 0 to $2^{\text{bit count}}-1$
UINT	Unsigned 16-bit integer	16	
UDINT	Unsigned 32-bit integer	32	
BYTE	Byte	8	--
WORD	Word	16	
DWORD	Double-word	32	

This manual uses the IEC61131-3 general data type that uses prefix ANY as shown below.



In a location indicated by the general data type, basic data of that data type can be specified.

2.3 REFERENCES

2.3.1 Reference Address

PMC-SD7

The PMC-SD7 address comprises a "%" letter, one or more alphabets, and digits as defined in IEC61131.

%I 1024

%I : The type of the signal ("%" + one or more alphabets)
1024: Bit or word number

Bit address and word address are distinguished by the first alphabet letter related with memory type. %R represents word address and others like %I, %Q, %M and %S represents bit address. Bit address at the beginning of each byte/word/dword can be used in functional instructions operating on byte/word/dword data.

PMC-SB7

The PMC-SB7 address comprises the address number and the bit number in the format as shown below.

X 127. 7

X 127 : Address number (digits after alphabet)
7 : Bit number (0 to 7)

An alphabet must be specified at the beginning of the address number to indicate the type of the signal. When specifying the address in the byte unit in the functional instruction, specify X127. In this case, "." and the bit number are not necessary.

2.3.2 User References

This section explains the references which can be used in a PMC program. The following lists the valid range of PMC addresses.

Name	Range	Size	Attribute	Remarks		Corresponding PMC-SB7 references
%I	1- 1024	1-bit (BOOL type)	Read-only	Input from I/O Link 1st channel		X0.0 - X127.7
	10001 - 11024			Input from I/O Link 2nd channel	*1	X200.0 - X327.7
	80001 - 81024			Input from built-in I/O	*2	X1000.0 - X1127.7
%IF	1- 6144	1-bit (BOOL type)	Read-only	Input from CNC	*3	F0.0 - F767.7
	10001 - 16144				*4	F1000.0 - F1767.7
	20001 - 26144				*4	F2000.0 - F2767.7
	30001 - 36144				*5	F3000.0 - F3767.7
%Q	1- 1024	1-bit (BOOL type)		Output to I/O Link 1st channel		Y0.0 - Y127.7
	10001 - 11024			Output to I/O Link 2nd channel	*1	Y200.0 - Y327.7
	80001 - 81024			Output to built-in I/O	*2	Y1000.0 - Y1127.7
%QG	1- 6144	1-bit (BOOL type)		Output to CNC	*3	G0.0 - G767.7
	10001 - 16144				*4	G1000.0 - G1767.7
	20001 - 26144				*4	G2000.0 - G2767.7
	30001 - 36144				*5	G3000.0 - G3767.7
%R	1- 5000	16-bit (WORD type)	Retentive	Data Register		D0.0 - D9999.7
%M	1- 64000	1-bit (BOOL type)		Internal relay		R0.0 - R7999.7
%ME	1- 64000	1-bit (BOOL type)			*7	E0.0 - E7999.7
%MA	1- 2000	1-bit (BOOL type)		Message display request		A0.0 - A249.7
%MK	1- 800	1-bit (BOOL type)	Retentive	Internal relay		K0.0 - K99.7
%S	1- 3872	1-bit (BOOL type)	Read-only	System area	*6	R9016.0 - R9499.7
%ST	1- 64	1-bit (BOOL type)	Read-only		*6	R9008.0 - R9015.7
%SA	1- 2000	1-bit (BOOL type)	Read-only		*8	A9000.0 - A9249.7
%SK	1- 160	1-bit (BOOL type)	Retentive		*9	K900.0 - K919.7

NOTE

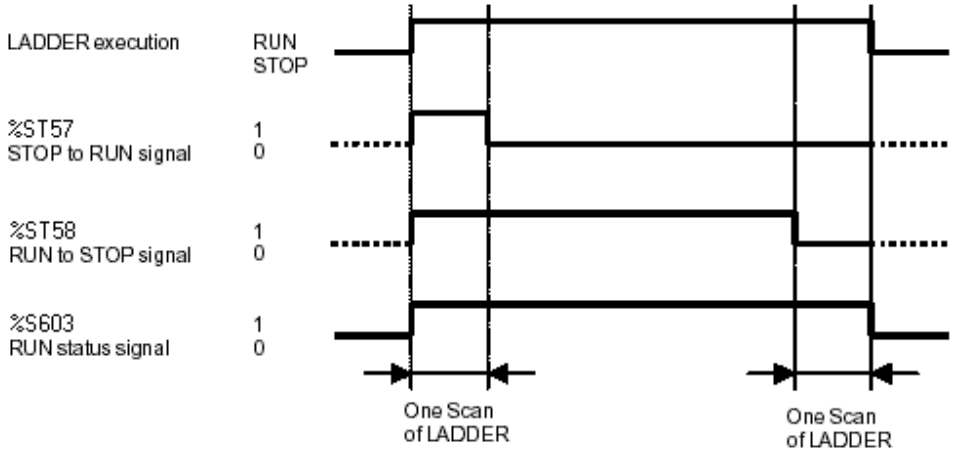
1 These references are used for 2nd channel of I/O Link. I/O link expansion option is necessary.
2 This area is reserved for PMC. I/O can not be assigned in it.
Don't use it in sequence program.
3 This area contains PMC reserve. Actual available references depend on the configuration of CNC system.
4 This area is used for multi path system. It contains PMC reserve. Actual available references depend on the configuration of CNC system.
5 This area is for PMC reserve. Don't use it in sequence program.
6 This area is used for PMC system software as special relay. Please use these according to the explanation of each address.
7 This area can be used equally as internal relay (%M).
These relays (%ME) are volatile type. However, these can be input/output from/to memory card, etc, as PMC parameter.
8 These references are message display state signals that have one-to-one correspondence to message display request signal (%MA). It is impossible to write into these references.
9 This area is used for PMC control software. Please use these signals according to each explanation.

2.3.3 System References

The following explains the system references used by the PMC control software.

Address	Name	Explanation
%S601	#ALW_OFF	Always OFF signal.
%S602	#ALW_ON	Always ON signal.
%S606	#T_200MS	200 ms cycle signal. (104 ms ON, 96 ms OFF)
%S607	#T_SEC	1 sec cycle signal. (504 ms ON, 496 ms OFF)
%ST57	#STP_RUN	STOP to RUN transition signal. Turned ON when the current scan is the first scan.
%ST58	#RUN_STP	RUN to STOP transition signal. Reset from 1 (ON) to 0 (OFF) when the current scan is the last scan.
%S603	#RUNNING	RUN status signal. Turned ON when during RUN.
%SK1	# NEN_OUT	Prevents the edit (output) operation of the sequence program.
%SK2	# EN_PRG	Validates the program debugging function.
%SK3	# RUN_NAT	Manually (soft key) executes a sequence program.
%SK5	# EN_RAM	Enables display of data input on the memory content display screen.
%SK8	# NEN_DTC	The PMC parameter data table control screen is not displayed.
%SK15	# EN_IN	Validates the sequence program edit function.
%SK17	# FROM_AT	The ladder program is automatically written to F-ROM after editing.
%SK19	# DSP_RUN	“RUN” or “STOP” soft key is displayed.
%SK23	#HIDEPRM	The PMC parameter screen is not displayed.
%SK24	#PROTPRM	The PMC parameter is not changed by reading of the I/O screen.
%SK49	#EN_OVR	The override mode becomes enabled.
%SK54	#TRC_AT	Automatically starts the trace function (TRACE) when the power is turned on.

RUN to STOP Transition Signal, STOP to RUN Transition Signal and RUN Status Signal



(i) "STOP to RUN transition signal" (%ST57)
When a STOP to RUN event is detected on system software, this signal will be turned on during the 1st scan of LADDER program after LADDER started. This signal has individual status corresponding the scan of each LADDER execution level. This signal is completely turned on during whole of the 1st scan in any execution level of LADDER program.

- When does a STOP to RUN event happen?
- LADDER starting at every power on cycle
- Pressing a "RUN" soft-key on a PMC screen
- "RUN" commanded by FANUC LADDER-IIIC

Referring this signal in a LADDER program, you can recognize and handle a "STOP to RUN" transition event. You can program a pre-processing for LADDER execution.

NOTE
This signal is available only in LADDER program. Don't refer this signal on other systems or programs, such a network board, C executor program, FOCAS1/Ethernet or FOCAS1/HSSB library etc. because this signal has individual status in each LADDER execution.

(ii) "RUN to STOP transition signal" (%ST58)
When a RUN to STOP event is detected on system software, this signal will be turned off during the last scan of LADDER program before LADDER stopped. This signal has individual status corresponding the scan of each LADDER execution level. This signal is completely turned off during the last scan before any execution level of LADDER program stops.

- When does a RUN to STOP event happen?
- Pressing a "STOP" soft-key on a PMC screen
- "STOP" commanded by FANUC LADDER-IIIC
- Storing a LADDER program to PMC on PMC "I/O" screen
- Storing a LADDER program to PMC using FANUC LADDER-IIIC

Referring this signal in a LADDER program, you can recognize and handle a "RUN to STOP" transition event. You can program a post-processing for LADDER execution (i.e. pre-processing for stopping of LADDER execution). For example, set or reset any appropriate signals into certain condition for the safety.

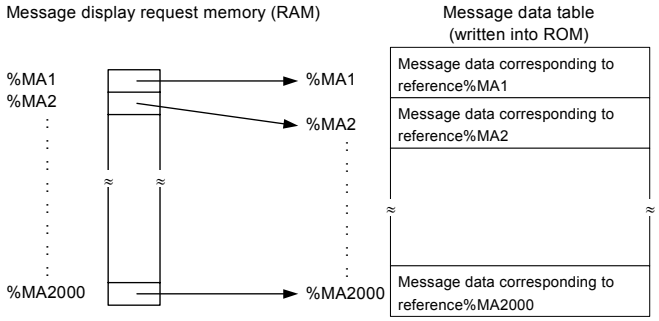
NOTE
1 This signal is available only in LADDER program. Don't refer this signal on other systems or programs, such a network board, C executor program, FOCAS1/Ethernet or FOCAS1/HSSB library etc. because this signal has individual status in each LADDER execution level.
2 You can not handle this event using this signal at a power off sequence and system alarm of CNC in which the execution of LADDER and I/O scanning are completely shut down.

(iii) "RUN status signal" (%S603)
Referring this signal on other systems or programs, such a network board, C executor program, FOCAS1/Ethernet or FOCAS1/HSSB library etc. you can know the status of LADDER program execution.

2.3.4 Message Display Reference

Function

PMC can display messages on the CNC screen using the CNC external operator and alarm message functions. An external alarm message can also be specified with its message number to place the CNC in the alarm state.
To use this function, create message data using a programmer (FANUC LADDER-IIIC) and store it in the message data table.
When a message display request memory bit (reference%MAxx) is turned on using the sequence program, the corresponding message in the message data table is displayed.



Message display request memory and message data table

Message Display Request Memory

Message display request memory consists of 2000 bits corresponding to reference%MA1 to reference%MA2000. Each bit corresponds to one message data item. The entire memory indicates up to 2000 message display requests. To display a message on the CNC screen, set the corresponding display request memory bit to "1". To erase the message from the screen, set the corresponding display request memory bit to "0".

Message Data Table

This table is used to store message data corresponding to message display request memory bits. It is written into program memory together with the sequence program.

Number of Characters Consisting of a Message

Messages are displayed on the NC alarm and operator message screens. The number of characters that can be displayed on the screen is 32 (32 bytes) for an alarm message or 255 (255 bytes) for an operator message.
An alphanumeric character requires 1 byte. For other special characters including half-size kana characters and kanji characters (full-size characters), note the following: a half-size kana character requires 2 bytes, a kanji character requires 4 bytes, and character "@" used to code a special character requires 1 byte.

Message Number

Always specify a message number for each message data item.
The relationships between message numbers and their display are shown below:

Path number	Message numbers	Destination CNC screen	Display
1	1000 to 1999	Alarm screen (on path 1)	Alarm messages - Path 1 is placed in the alarm state.
	2000 to 2099	Operator message screen	Operator messages
	2100 to 2999		Operator messages (with no message numbers) - Only message data is displayed, and no message number is displayed.
2	1000 to 1999	Alarm screen (on path 2)	Alarm message - Path 2 is placed in the alarm state.
3	1000 to 1999	Alarm screen (on path 3)	Alarm message - Path 3 is placed in the alarm state.

NOTE
If a message number other than the above is specified, no message is displayed.

Path Number

Always specify a path number for each message data item.
Each message is displayed on the screen for the path indicated by the number.

NOTE
If a nonexistent path number such as 0 is specified, no message is displayed.

Setting Message Data

Input message data characters using a programmer. Kana and kanji characters can be input as numeric data with a special symbol "@". See the later section "Inputting kana and kanji characters" for details.

Number of Displayed Messages

Up to four alarm messages or one operator message can be displayed at a time.

NOTE
The number of messages that can be displayed at a time may differ depending on the NC model or setting.

Message Display Order

When multiple message display request memory bits are turned on, which message is displayed first is unpredictable.
When the number of message display request memory bits that are turned on exceeds the maximum number of messages that can be displayed at a time, the excess message display requests are not executed, but held as long as they are on. When the displayed messages are cleared, the message display requests being held are recognized, then the corresponding messages are displayed. At this time, which message display requests are recognized is unpredictable.

Using the Message Display Function Together with the External Data Input Function

Use the PMC_EXIN functional instruction to perform external tool compensation, external workpiece number search, and other operations using the message display function, together with the external data input function which uses the same interface.

Numeric Data Display

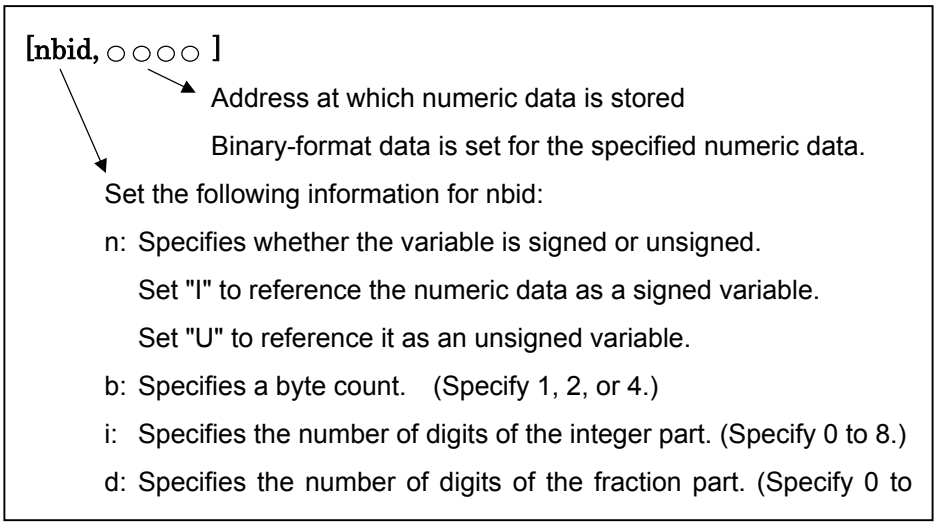
Once programmed, message data is treated as fixed data that cannot be modified. For numeric values to be made variable during execution, define a memory address in message data and set numeric data in the memory using the sequence program to display any numeric data. Using this function, data such as a tool number that frequently changes during automatic operation can be displayed. See the later section "Setting numeric data display" for details.

Option

To use the message display function, the CNC external data input or message display option is required.

Setting Numeric Data Display

To display numeric data, define, in message data, the number of digits of the numeric value and the address in memory containing the numeric data. The data for defining numeric data must be enclosed in brackets ([]) to distinguish it from other message data. Therefore, brackets ([]) cannot be used as message display symbols because they are used for numeric data. Set data in brackets as follows:



An address for PMC-SD7 (such as %R or %I) can be used for the address at which numeric data is stored.

Example: Displaying UINT-type numeric data with a 5-digit integer part and 3-digit fraction part from %R1 using a PMC-SD7 address

[U253,R1]

NOTE
1. The total number of digits of the integer and fraction parts must not exceed 8.
2. For erroneous numeric data such as data consisting of more than eight digits, a blank is displayed.
3. The numeric data of address of the BYTE boundary such as %I1, %I9 or %I17 is displayed. When the address is not the BYTE boundary such as %I2, %I3 or %I4, the value that rounded in the BYTE boundary such as "%I1" is displayed.
4. The % of address (such as %R) within the message data is unnecessary.

Inputting Kana and Kanji Characters

When the personal computer environment in which the programmer runs supports Japanese-language input, kana and kanji characters can directly be input. When the environment does not support Japanese-language input, input kana and kanji characters as follows:

Half-size Kana Characters

1 Data format
Input the numeric codes corresponding to desired kana characters between @ and @.

2 Input method
Input kana characters using their numeric codes by referencing the character code table given later. One kana character requires 2 bytes. Alphanumeric characters can also be input using their numeric codes.

3 Example
Inputting ATC ? チョウサ OK:

```
ATC @20 3F C1 AE B3 BB @OK
      |  |  |  |  |  |
   Blank  ?  チ  ョ  ウ  サ
```

Kanji Characters (Full-size Characters)

1 Data format
Input the numeric codes corresponding to desired kanji characters between @02 and 01@.

2 Input method
Input kanji characters using their numeric codes by referencing the kanji, hiragana, and special character code table. One kanji character requires 4 bytes.

3 Example
Inputting ATC ? 調査 OK:

```
ATC @20 3F @@02 4434 3A3A 01 @OK
      |  |       |    |
   Blank  ?      調    査
```

NOTE

1 To define character @, input @40...@.
```
          |
          @
```
2 To define a line feed of a screen display message, input @0A@ at the end of data.

3 For numeric code input, code @ requires 1 byte. When a character, for example, a space is input using its numeric code (20), 1 byte for "2" and 1 byte for "0", a total of 2 bytes, are required.

4 Control codes are assigned for characters as follows:
02: 2-byte code (kanji and hiragana characters)
01: 1-byte code (alphanumeric and half-size kana characters)
Do not specify 02 or 01 between @02 and @01, as shown below. If so, the data may not be displayed normally.
@02...02...01@, @02...01...01@

5 No variable display specification is allowed between @ and @ or between @02 and 01@.

Character code table

	2	3	4	5	A	B	C	D
0	Space	0	2	P	～	ー (*3)	タ	ミ
1	!	1	A	Q	。	ア	チ	ム
2	“	2	B	R	「	イ	ツ	メ
3	#	3	C	S	」	ウ	テ	モ
4	$	4	D	T	、	エ	ト	ヤ
5	%	5	E	U	・	オ	ナ	ユ
6	&	6	F	V	ヲ	カ	ニ	ヨ
7	‘	7	G	W	ァ	キ	ヌ	ラ
8	(	8	H	X	ィ	ク	ネ	リ
9	)	9	I	Y	ゥ	ケ	ノ	ル
A	*	:	J	Z	ェ	コ	ハ	レ
B	+	;	K	[	ォ	サ	ヒ	ロ
C	,	<	L	¥	ャ	シ	フ	ワ
D	- (*1)	=	M	]	ュ	ス	ヘ	ン
E	.	>	N	^	ョ	セ	ホ	゛
F	/	?	O	_ (*2)	ッ	ソ	マ	゜

NOTE
(*1) Minus
(*2) Under bar
(*3) Long bar

About 4 digits of code for available kanji characters, please refer to the Appendix D.

3 I/O LINK

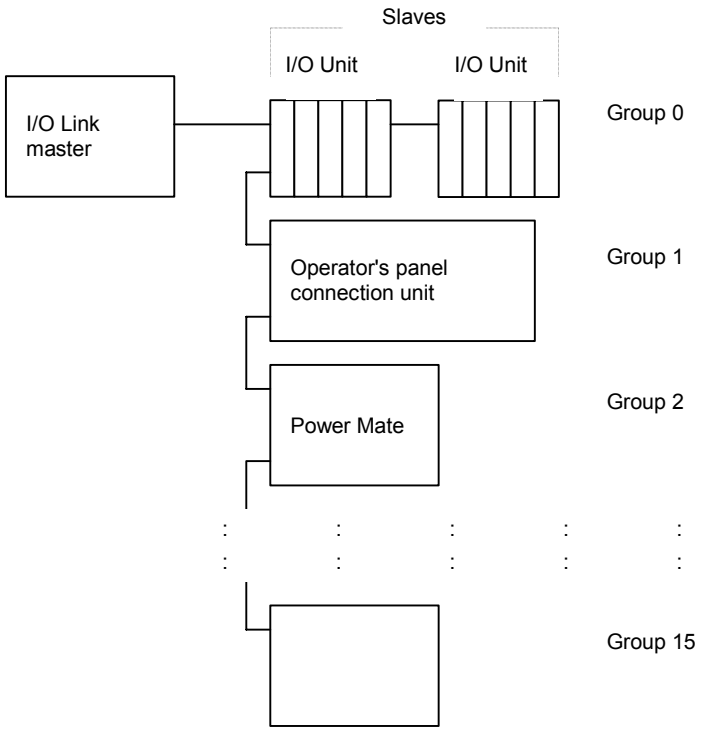
3.1 WHAT IS THE I/O LINK?

The FANUC I/O Link is a serial interface which passes input/output signals between the PMC and each I/O device at a high speed.
For each channel, up to 1024 DI points and up to 1024 DO points can be connected and controlled from the PMC.

NOTE
1 To use 2nd channel of the I/O Link, the I/O Link point expansion option is required.
2 The transfer cycle of signals from I/O devices is 2 ms.

3.1.1 Configuration of an I/O Link

The following figure shows a basic configuration of the I/O Link.



(1) The I/O Link consists of one master and multiple slaves.
Master: CNC (such as Series 16*i*-B)
Slaves: I/O Unit-MODEL A, Power Mate, operator's panel connection unit, and other devices

(2) Up to 16 groups of slaves can be connected to one I/O Link.
Group numbers 0 to 15 are sequentially assigned. Number 0 is assigned to the group nearest to the master.
The number of connected slaves in a group differs depending on the types of slaves.

(3) Any slave can be connected in any group. One group must consist of slaves of the same type, however.

NOTE
1 Turn the power to the slaves and master on simultaneously or turn the power to the slaves on before turning the power to the master.
2 When turning the power to the master off, also turn the power to all slaves off. Turn the power to all slaves on again before turning the power to the master on or turn the power to all slaves and the master on simultaneously. Turn the power to the master on after turning the power to all slaves on or turn the power to the master and all slaves on simultaneously.
3 For the maximum number of slaves per group that can be connected, refer to the hardware connection manual for each I/O device used as a slave.

3.1.2 Numbers of Input Points and of Output Points of the I/O Link

The I/O Link has up to 1024 input points and up to 1024 output points for each channel when viewed from the master. These I/O points can be assigned to each slave to periodically pass I/O data between the master and each slave.
Each slave occupies the predetermined number of I/O points.
The total number of I/O points occupied by all slaves connected to one channel is up to 1024 points (128 bytes) for each of input and output.
The number of I/O points occupied by one group is up to 256 points (32 bytes) for each of input and output.

NOTE
The number of occupied I/O points may differ from the actual number of I/O points. For example, if the number of input points is smaller than or equal to that of output points for a group, the number of input points is assumed equal to that of output points. For this reason, when the number of input points for the actually connected hardware components is 128 and that of output points is 256, the number of occupied input points is assumed to be 256 because there is the following relationship between the numbers of input points and of output points:

128 (number of input points) ≤ 256 (number of output points)

For more specific rules, see Section 3.2.

3.2 ASSIGNMENT METHOD

To use an I/O device as a slave, assign connection information to %I addresses (input) and %Q addresses (output) of the PMC. The machine tool builder should determine addresses to be used for input/output of each I/O device in a sequence program. Connection information can be assigned to these determined addresses using the FANUC LADDER-IIIC. The information is written in the flash ROM together with the sequence program. For this reason, the set information is not changed unless the sequence program is changed.
Information to be set to addresses includes the connection location and module name of each I/O device.
The connection location of an I/O device is represented by its group, base, and slot numbers.

Setting the connection location

I/O devices can roughly be divided into the following three types according to the method for specifying the connection location.

(1) Type of I/O device whose connection location is specified with its group, base, and slot numbers
I/O Unit-MODEL A is of this type. Specify the connection location with its group, base, and slot numbers.
The range of valid settings of each item is as follows:
Group = 0 to 15
Base = 0 and 1
Slot = 1 to 10 (number of a slot on a I/O Unit-MODEL A base board)

(2) Type of I/O device whose connection location is specified with its group and unit numbers
I/O Unit-MODEL B and handy machine operator's panels are of this type.
The range of valid settings of each item is as follows:
Group = 0 to 15
Unit = 0 to 30 (NOTE)

NOTE
For detailed information on settings, see Subsections 3.2.2 and 3.2.6.

(3) Type of I/O device whose connection location is specified with its group number
Machine operator's panel interface unit, I/O Link connection unit, Power Mate, and other devices are of this type. One unit of this type occupies one group.
The range of valid settings of group is as follows:
Group = 0 to 15

Setting the module name

Assign the %I or %Q addresses to the module as input/output of each I/O device. Because %I and %Q address are BIT memory, a starting address for BYTE must be assigned at “modulo 8 plus 1”, in other words, at 1, 9, 17, 25, 33, 41, etc.
For the module name, see Tables 3.2 (a) to (f).
The number of bytes of the address occupied is determined for each module name. The number of occupied I/O points per byte is 8.

For details of the assignment method, see the assignment method for each I/O device described later.

Table 3.2 (a) Module names (1)

Name	Module name	Occupied address	Description
Input modules for I/O Unit-MODEL A	A03B-0807-J101(C101)	4 bytes for input	Input32 (AID32A1)
	A03B-0807-J102(C102)	4 bytes for input	Input32 (AID32B1)
	A03B-0807-J111(C111)	4 bytes for input	Input32 (AID32H1)
	A03B-0807-J103(C103)	2 bytes for input	Input16 (AID16C)
	A03B-0807-J104(C104)	2 bytes for input	Input16 (AID16D)
	A03B-0807-J113(C113)	2 bytes for input	Input16 (AID16K)
	A03B-0807-J114(C114)	2 bytes for input	Input16 (AID16L)
	A03B-0807-J105(C105)	4 bytes for input	Input32 (AID32E1)
	A03B-0807-J110(C110)	4 bytes for input	Input32 (AID32E2)
	A03B-0807-J106(C106)	4 bytes for input	Input32 (AID32F1)
	A03B-0807-J109(C109)	4 bytes for input	Input32 (AID32F2)
	A03B-0807-J107(C107)	2 bytes for input	Input16 (AIA16G)
	A03B-0807-J051(C051)	8 bytes for input	Input64 (AAD04A)
	A03B-0807-J053(C053)	4 bytes for input	Input32
	A03B-0819-J053(C053)	4 bytes for input	Input32
	A03B-0807-C108(C108)	1 byte for input	Input8
	A03B-0807-C112(C112)	1 byte for input	Input8
	A03B-0819-J101(C101)	4 bytes for input	Input32 (AID32A1)
	A03B-0819-J102(C102)	4 bytes for input	Input32 (AID32B1)
	A03B-0819-J111(C111)	4 bytes for input	Input32 (AID32H1)
	A03B-0819-J103(C103)	2 bytes for input	Input16 (AID16C)
	A03B-0819-J104(C104)	2 bytes for input	Input16 (AID16D)
	A03B-0819-J113(C113)	2 bytes for input	Input16 (AID16K)
	A03B-0819-J114(C114)	2 bytes for input	Input16 (AID16L)
	A03B-0819-J105(C105)	4 bytes for input	Input32 (AID32E1)
	A03B-0819-J110(C110)	4 bytes for input	Input32 (AID32E2)
	A03B-0819-J106(C106)	4 bytes for input	Input32 (AID32F1)
	A03B-0819-J109(C109)	4 bytes for input	Input32 (AID32F2)
	A03B-0819-J107(C107)	2 bytes for input	Input16 (AIA16G)
	A03B-0819-J051(C051)	8 bytes for input	Input64 (AAD04A)
	Generic In (1byte)	1 byte for input	Input8
	Generic In (2byte)	2 bytes for input	Input16
	Generic In (3byte)	3 bytes for input	Input24
	Generic In (4byte)	4 bytes for input	Input32
	Generic In (5byte)	5 bytes for input	Input40
	Generic In (6byte)	6 bytes for input	Input48
	Generic In (7byte)	7 bytes for input	Input56
	Generic In (8byte)	8 bytes for input	Input64

Table 3.2 (b) Module names (2)

Name	Module name	Occupied address	Description
Output modules for I/O Unit-MODEL A	A03B-0807-J162(C162)	4 bytes for output	Output32 (AOD32A1)
	A03B-0807-J151(C151)	1 byte for output	Output8 (AOD08C)
	A03B-0807-J152(C152)	1 byte for output	Output8 (AOD08D)
	A03B-0807-J153(C153)	2 bytes for output	Output16 (AOD16C)
	A03B-0807-J154(C154)	2 bytes for output	Output16 (AOD16D)
	A03B-0807-J155(C155)	4 bytes for output	Output32 (AOD32C1)
	A03B-0807-J172(C172)	4 bytes for output	Output32 (AOD32C2)
	A03B-0807-J156(C156)	4 bytes for output	Output32 (AOD32D1)
	A03B-0807-J167(C167)	4 bytes for output	Output32 (AOD32D2)
	A03B-0807-J157(C157)	1 byte for output	Output8 (AOA05E)
	A03B-0807-J158(C158)	1 byte for output	Output8 (AOA08E)
	A03B-0807-J159(C159)	2 bytes for output	Output16 (AOA12F)
	A03B-0807-J160(C160)	1 byte for output	Output8 (AOR08G)
	A03B-0807-J161(C161)	2 bytes for output	Output16 (AOR16G)
	A03B-0807-J165(C165)	2 bytes for output	Output16 (AOR16H2)
	A03B-0807-J052(C052)	4 bytes for output	Output32 (ADA02A)
	A03B-0807-C164(C164)	1 byte for output	Output8
	A03B-0807-C169(C109)	1 byte for output	Output8
	A03B-0807-C170(C170)	1 byte for output	Output8
	A03B-0807-C171(C171)	2 bytes for output	Output16
	A03B-0807-C166(C166)	1 byte for output	Output8
	A03B-0807-C168(C168)	1 byte for output	Output8
	A03B-0807-J182(C182)	2 bytes for output	Output16
	A03B-0819-J162(C162)	4 bytes for output	Output32 (AOD32A1)
	A03B-0819-J151(C151)	1 byte for output	Output8 (AOD08C)
	A03B-0819-J152(C152)	1 byte for output	Output8 (AOD08D)
	A03B-0819-J153(C153)	2 bytes for output	Output16 (AOD16C)
	A03B-0819-J154(C154)	2 bytes for output	Output16 (AOD16D)
	A03B-0819-J155(C155)	4 bytes for output	Output32 (AOD32C1)
	A03B-0819-J172(C172)	4 bytes for output	Output32 (AOD32C2)
	A03B-0819-J156(C156)	4 bytes for output	Output32 (AOD32D1)
	A03B-0819-J167(C167)	4 bytes for output	Output32 (AOD32D2)
	A03B-0819-J157(C157)	1 byte for output	Output8 (AOA05E)
	A03B-0819-J158(C158)	1 byte for output	Output8 (AOA08E)
	A03B-0819-J159(C159)	2 bytes for output	Output16 (AOA12F)
	A03B-0819-J160(C160)	1 byte for output	Output8 (AOR08G)
	A03B-0819-J161(C161)	2 bytes for output	Output16 (AOR16G)
	A03B-0819-J165(C165)	2 bytes for output	Output16 (AOR16H2)
	A03B-0819-J052(C052)	4 bytes for output	Output32 (ADA02A)
	A03B-0819-J182(C182)	2 bytes for output	Output16
	Generic Out (1byte)	1 byte for output	Output8
	Generic Out (2byte)	2 bytes for output	Output16
	Generic Out (3byte)	3 bytes for output	Output24
	Generic Out (4byte)	4 bytes for output	Output32
	Generic Out (5byte)	5 bytes for output	Output40
	Generic Out (6byte)	6 bytes for output	Output48
	Generic Out (7byte)	7 bytes for output	Output56
	Generic Out (8byte)	8 bytes for output	Output64
Input/output module for I/O Unit-MODEL A	A03B-0807-C200(C200)	3 bytes for input 2 bytes for output	Input24 / Output16 (AIO40A)
	A03B-0807-J183(C183)	1 byte for input 1 byte for output	Input8 / Output8 (AOD08DP)
	A03B-0819-J183(C183)	1 byte for input 1 byte for output	Input8 / Output8 (AOD08DP)

Table 3.2 (c) Module names (3)

Name	Module name	Occupied address	Description
Modules for I/O Unit-MODEL B Handy Machine Operator's Panel	##	4 bytes for input	Input32 (Power on/off)
	#1(In)	1 byte for input	Input8
	#2(In)	2 byte for input	Input16
	#3(In)	3 byte for input	Input24
	#4(In)	4 byte for input	Input32
	#6(In)	6 byte for input	Input48
	#8(In)	8 byte for input	Input64
	#10(In)	10 byte for input	Input80
	#1(Out)	1 byte for output	Out put8
	#2(Out)	2 byte for output	Out put16
	#3(Out)	3 byte for output	Out put24
	#4(Out)	4 byte for output	Out put32
	#6(Out)	6 byte for output	Out put48
	#8(Out)	8 byte for output	Out put64
	#10(Out)	10 byte for output	Out put80
I/O Connection Panel	B	24 bytes for input 16 bytes for output	BASIC Unit only
	BE	48 bytes for input 32 bytes for output	BASIC Unit + expansion unit
	BEM	104 bytes for input 32 bytes for output	BASIC Unit + expansion unit + MPG
	BEE	72 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit
	BEEM	104 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit + MPG
	BEEMM	112 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit + 2 MPG
	BEEE	96 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit
	BEEEM	104 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + MPG
	BEEEMM	112 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + 2 MPG
	BEEEMMM	120 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + 3 MPG
	BA	128 bytes for input 16 bytes for output	BASIC Unit only (Use Alarm DO)
	BEA	128 bytes for input 32 bytes for output	BASIC Unit + expansion unit (Use Alarm DO)
	BEMA	128 bytes for input 32 bytes for output	BASIC Unit + expansion unit + MPG (Use Alarm DO)
	BEEA	128 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit (Use Alarm DO)
	BEEMA	128 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit + MPG (Use Alarm DO)
	BEEMMA	128 bytes for input 48 bytes for output	BASIC Unit + 2 expansion unit + 2 MPG (Use Alarm DO)
	BEEEA	128 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit (Use Alarm DO)
	BEEEMA	128 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + MPG (Use Alarm DO)
	BEEEMMA	128 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + 2 MPG (Use Alarm DO)
	BEEEMMMA	128 bytes for input 64 bytes for output	BASIC Unit + 3 expansion unit + 3 MPG (Use Alarm DO)

Table 3.2 (d) Module names (4)

Name	Module name	Occupied address	Description
I/O Link Connect Unit	A20B-2000-0410 (1)	4 bytes for input 4 bytes for output	Input32 / Output32
	A20B-2000-0410 (2)	8 bytes for input 8 bytes for output	Input64 / Output64
	A20B-2000-0410 (3)	16 bytes for input 16 bytes for output	Input128 / Output128
	A20B-2000-0410 (4)	32 bytes for input 32 bytes for output	Input256 / Output256
	A20B-2000-0410	32 bytes for input 32 bytes for output	Input256 / Output256
	A20B-2000-0411 (1)	4 bytes for input 4 bytes for output	Input32 / Output32
	A20B-2000-0411 (2)	8 bytes for input 8 bytes for output	Input64 / Output64
	A20B-2000-0411 (3)	16 bytes for input 16 bytes for output	Input128 / Output128
	A20B-2000-0411 (4)	32 bytes for input 32 bytes for output	Input256 / Output256
	A20B-2000-0411	32 bytes for input 32 bytes for output	Input256 / Output256
	A20B-2000-0412 (1)	4 bytes for input 4 bytes for output	Input32 / Output32
	A20B-2000-0412 (2)	8 bytes for input 8 bytes for output	Input64 / Output64
	A20B-2000-0412 (3)	16 bytes for input 16 bytes for output	Input128 / Output128
	A20B-2000-0412 (4)	32 bytes for input 32 bytes for output	Input256 / Output256
	A20B-2000-0412	32 bytes for input 32 bytes for output	Input256 / Output256
Connection Unit	A16B-2200-0661	8 bytes for input 4 bytes for output	Input64 / Output32 Operator's panel connection unit sink DO
	A16B-2200-0660	12 bytes for input 8 bytes for output	Input96 / Output64 Operator's panel connection unit sink DO
	A16B-2202-0731	8 bytes for input 4 bytes for output	Input64 / Output32 Operator's panel connection unit source DO
	A16B-2202-0730	12 bytes for input 8 bytes for output	Input96 / Output64 Operator's panel connection unit source DO
	A16B-2201-0110(1)	16 bytes for input 16 bytes for output	Input128 / Output128
	A16B-2201-0110(2)	32 bytes for input 32 bytes for output	Input256 / Output256
	A16B-2201-0071	6 bytes for input 4 bytes for output	Input48 / Output32
	A16B-2201-0070	12 bytes for input 8 bytes for output	Input96 / Output64
	A16B-2600-0150	3 bytes for input 2 bytes for output	Input24 / Output16
	A16B-2202-0733	6 bytes for input 4 bytes for output	Input48 / Output32
	A16B-2202-0732	12 bytes for input 8 bytes for output	Input96 / Output64
	A16B-2600-0170	3 bytes for input 2 bytes for output	Input24 / Output16

Table 3.2 (e) Module names (5)

Name	Module name	Occupied address	Description
Operator's Panel	A20B-2002-0470 (1)	12 bytes for input 8 bytes for output	Input72 / Output56
	A20B-2002-0470 (2)	13 bytes for input 8 bytes for output	Input72 / Output56 1MPG
	A20B-2002-0470 (3)	14 bytes for input 8 bytes for output	Input72 / Output56 2MPG
	A20B-2002-0470 (4)	15 bytes for input 8 bytes for output	Input72 / Output56 3MPG
	A20B-2002-0470 (5)	16 bytes for input 8 bytes for output	Input72 / Output56 (use Alarm DO)
	A20B-2002-0470 (6)	16 bytes for input 8 bytes for output	Input72 / Output56 1MPG (use Alarm DO)
	A20B-2002-0470 (7)	16 bytes for input 8 bytes for output	Input72 / Output56 2MPG (use Alarm DO)
	A20B-2002-0470 (8)	16 bytes for input 8 bytes for output	Input72 / Output56 3MPG (use Alarm DO)
	A20B-2002-0520 (1)	6 bytes for input 4 bytes for output	Input48 / Output32
	A20B-2002-0520 (2)	13 bytes for input 4 bytes for output	Input48 / Output32 1MPG
	A20B-2002-0520 (3)	14 bytes for input 4 bytes for output	Input48 / Output32 2MPG
	A20B-2002-0520 (4)	15 bytes for input 4 bytes for output	Input48 / Output32 3MPG
	A20B-2002-0520 (5)	16 bytes for input 4 bytes for output	Input48 / Output32 (use Alarm DO)
	A20B-2002-0520 (6)	16 bytes for input 4 bytes for output	Input48 / Output32 1MPG (use Alarm DO)
	A20B-2002-0520 (7)	16 bytes for input 4 bytes for output	Input48 / Output32 2MPG (use Alarm DO)
	A20B-2002-0520 (8)	16 bytes for input 4 bytes for output	Input48 / Output32 3MPG (use Alarm DO)
	A20B-2002-0521 (1)	6 bytes for input 4 bytes for output	Input48 / Output32
Beta Servo	A06B-6093-H151	16 bytes for input 16 bytes for output	Input128 / Output128
FANUC CNC SYSTEM FANUC Power Mate	FS04A(In) / (Out)	4 bytes for input 4 bytes for output	Input32 / Output32
	FS08A(In) / (Out)	8 bytes for input 8 bytes for output	Input64 / Output64
	FANUC CNC(OC02I) / (OC02O)	16 bytes for input 16 bytes for output	Input128 / Output128
	FANUC CNC(OC03I) / (OC03O)	32 bytes for input 32 bytes for output	Input256 / Output256

Table 3.2 (f) Module names (6)

Name	Module name	Occupied address	Description
Others	1 byte Unit (In)	1 byte for input	Input8
	2 byte Unit (In)	2 bytes for input	Input16
	3 byte Unit (In)	3 bytes for input	Input24
	4 byte Unit (In)	4 bytes for input	Input32
	5 byte Unit (In)	5 bytes for input	Input40
	6 byte Unit (In)	6 bytes for input	Input48
	7 byte Unit (In)	7 bytes for input	Input56
	8 byte Unit (In)	8 bytes for input	Input64
	16 byte Unit (In)	16 bytes for input	Input128
	32 byte Unit (In)	32 bytes for input	Input256
	1 byte Unit (Out)	1 byte for output	Output8
	2 byte Unit (Out)	2 bytes for output	Output16
	3 byte Unit (Out)	3 bytes for output	Output24
	4 byte Unit (Out)	4 bytes for output	Output32
	5 byte Unit (Out)	5 bytes for output	Output40
	6 byte Unit (Out)	6 bytes for output	Output48
	7 byte Unit (Out)	7 bytes for output	Output56
	8 byte Unit (Out)	8 bytes for output	Output64
	16 byte Unit (Out)	16 bytes for output	Output128
	32 byte Unit (Out)	32 bytes for output	Output256

NOTE
1 For the specifications and connection of each I/O device, refer to the relevant hardware connection manual.
2 For the assignment method for each I/O device, see Subsections 3.2.1 to 3.2.7.
3 As assignment data for a handy machine operator's panel, assign multiple module names successively. For details, see Subsection 3.2.6.

3.2.1 Assignment Method for I/O Unit-MODEL A

Figs. 3.2.1 (a) and 3.2.1 (b) show sample configurations of I/O Unit-MODEL A.

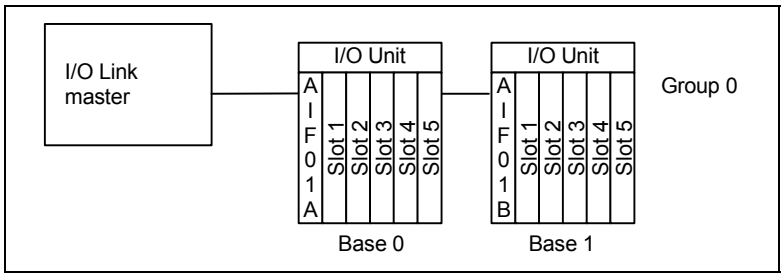


Fig. 3.2.1 (a)

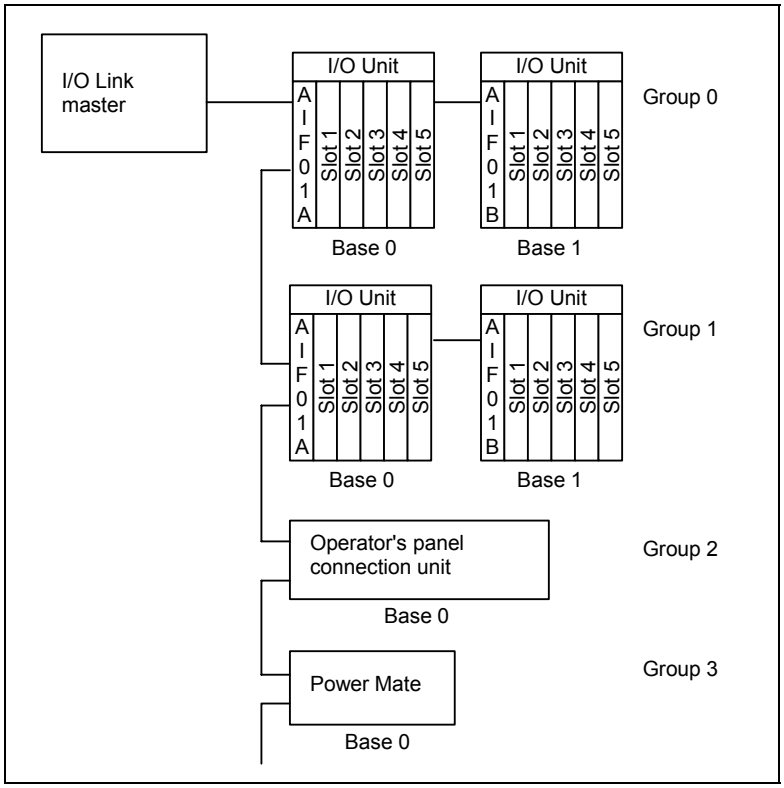


Fig. 3.2.1 (b)

Assignment method

(1) Group number
For I/O Unit-MODEL A, up to two I/O units can be connected when interface module AIF01A is used as the basic unit and expansion interface module AIF01B is also used. This is called the base expansion function. This set of up to two I/O units comprises one group (see Fig. 3.2.1 (a)). When required I/O modules cannot be contained only in one group or when multiple I/O units are to be distributed at distant locations on the machine side, the second AIF01A can be connected to the first AIF01A using a cable to add a group. (See Fig. 3.2.1 (b).)

(2) Base number
One group consists of up to two I/O base units. The base number of the I/O unit on which interface module AIF01A is mounted is 0; the base number of the other I/O unit is 1.
In other words, when the base expansion function is used, the base number of the basic unit is always 0 and that of the expansion unit is always 1. When the base expansion function is not used, the base number is always 0.

(3) Slot number
On one I/O base unit, up to five (ABU05A) or ten (ABU10A) I/O modules can be mounted depending on the type of I/O base unit. The location of each module on the I/O base unit is represented by a slot number. For each base unit, the location of the I/O interface module is 0 and slot numbers 1 to 10 are assigned from left to right. Each module can be mounted into any desired slot. I/O modules may not be mounted closely from left to right. An intermediate slot may not be used.

(4) Module name
For module names, see Tables 3.2 (a) to (b) in Section 3.2 above.

Number of occupied I/O points

Obtain the number of occupied I/O points as follows.

[Number of output points]

Total number of points required for output modules used in one group	Number of occupied I/O points
0 to 32	32
40 to 64	64
72 to 128	128
136 to 256	256

[Number of input points]

Total number of points required for input modules used in one group	Number of occupied I/O points
0 to 32	32
40 to 64	64
72 to 128	128
136 to 256	256

If the obtained total number of input points is smaller than or equal to that of output points in the same group, however, the number of input points is assumed equal to that of output points. For this reason, when the number of input points for the actually connected hardware components is 128 and that of output points is 256, the number of occupied input points is assumed to be 256.

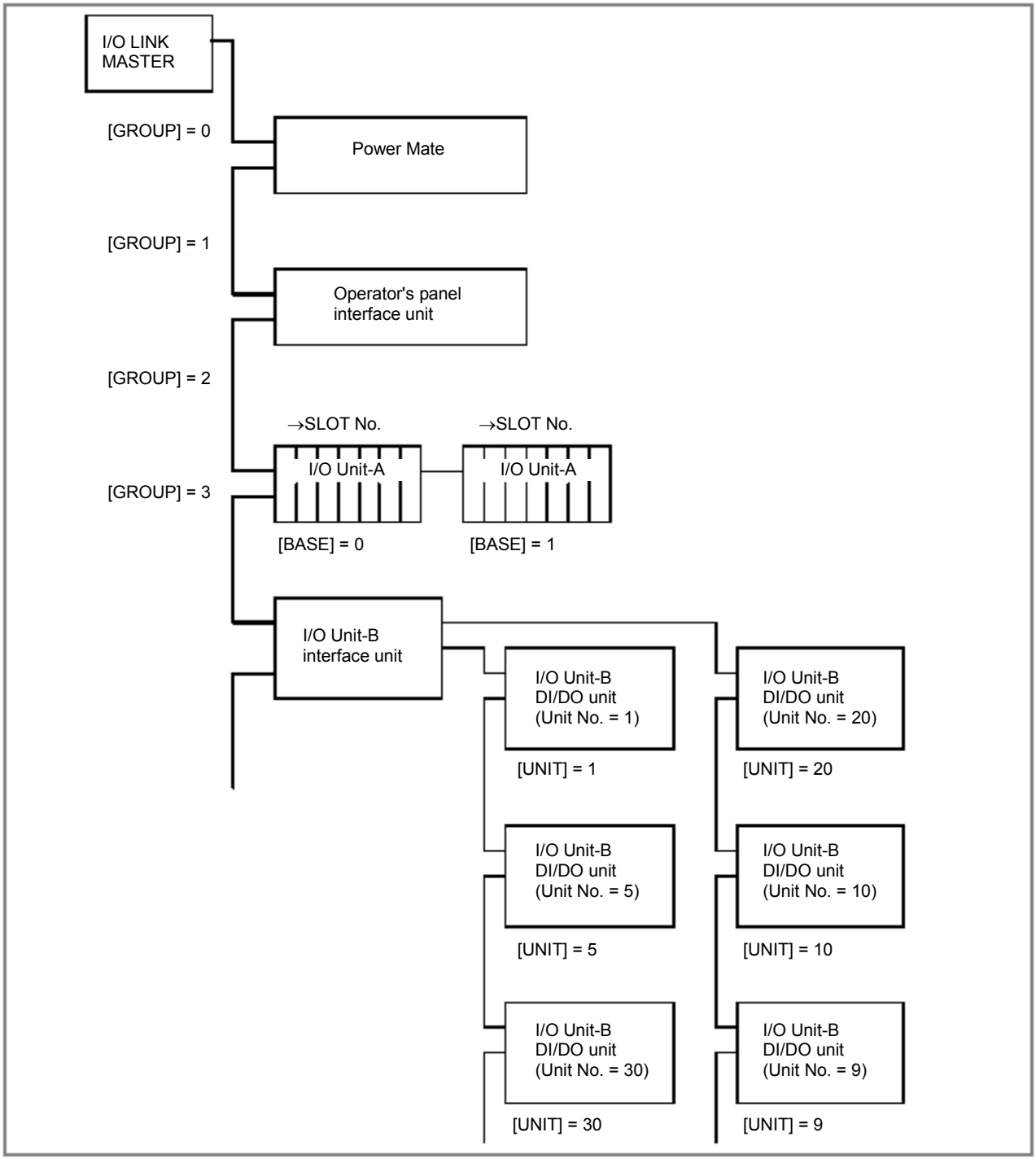
Related hardware manual

"FANUC I/O Unit-MODEL A Connection and Maintenance Manual" (B-61813E)

NOTE
For the specifications and connection of I/O Unit-MODEL A and related I/O modules, refer to the hardware connection manual for each I/O device.

3.2.2 Assignment Method for I/O Unit-MODEL B

I/O Unit-MODEL B can be used together with I/O Link devices such as the Power Mate, operator's panel interface unit, connection unit, and I/O Unit-MODEL A. In this case, I/O Unit-MODEL B comprises one group and other units cannot be contained in the group. An example of connection is shown below.



Assignment method

As the group number, set the group number used in the configuration. As the slot number, set the unit number of a DI/DO unit of I/O Unit-MODEL B. To assign power on-off information, set 0 for the unit number.
Set the following values for the slot number and assignment name:

Unit number: 0: Power on-off information
1 to 30: Unit number

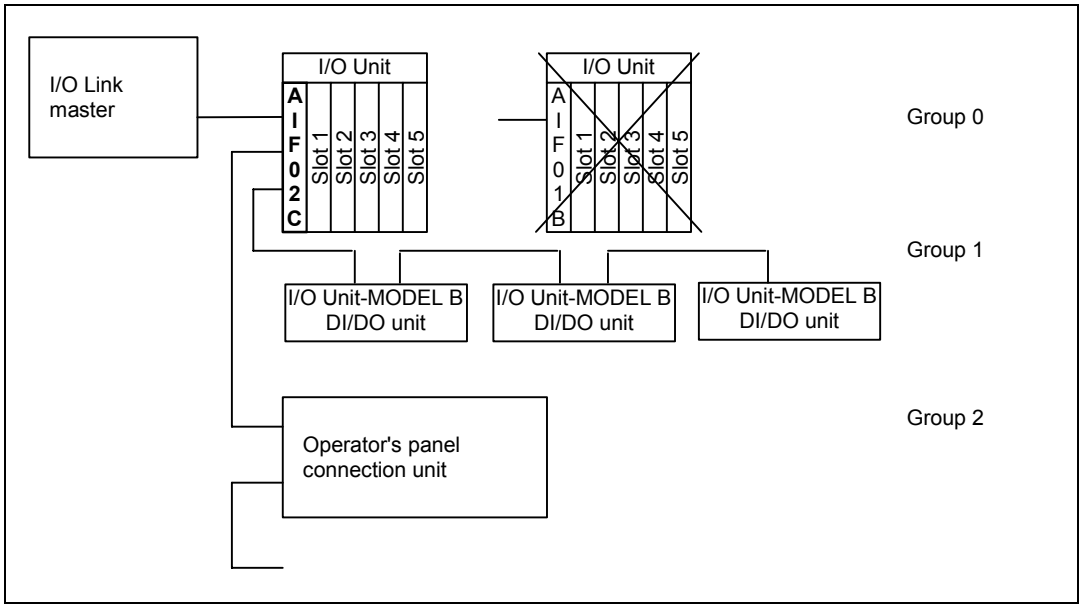
Assignment name: Module name representing the address occupied by the I/O Unit-MODEL B DI/DO unit (see Table 3.2 (c).)

Number of input or output points required for [basic unit] + [expansion unit]	Assignment name	Occupied address
1 byte	#1	1 byte for input/output
2 bytes	#2	2 bytes for input/output
3 bytes	#3	3 bytes for input/output
4 bytes	#4	4 bytes for input/output
6 bytes	#6	6 bytes for input/output
8 bytes	#8	8 bytes for input/output
10 bytes	#10	10 bytes for input/output
Power on-off information	##	4 bytes for input

NOTE
When channels 2 is also used to connect I/O devices, the maximum total number of groups used for connecting I/O Unit-MODEL B with channels 1 to 2 is 8.

Interface module incorporating I/O Unit-MODEL A

Interface module AIF02C can control communication both with I/O Unit-MODEL A and with I/O Unit-MODEL B.



For the AIF02C, the base expansion function of the AIF02A is removed and the functions of the I/O Unit-MODEL B interface unit are added.
When I/O Unit-MODEL A is not used, only I/O Unit-MODEL B cannot be used. The base expansion function cannot also be used.
The AIF02C occupies two groups. Assignment is required for each of I/O Unit-MODEL A and I/O Unit-MODEL B.

NOTE
For details of the AIF02C, refer to "FANUC I/O Unit-MODEL A Connection and Maintenance Manual" (B-61813E).

Related hardware manual

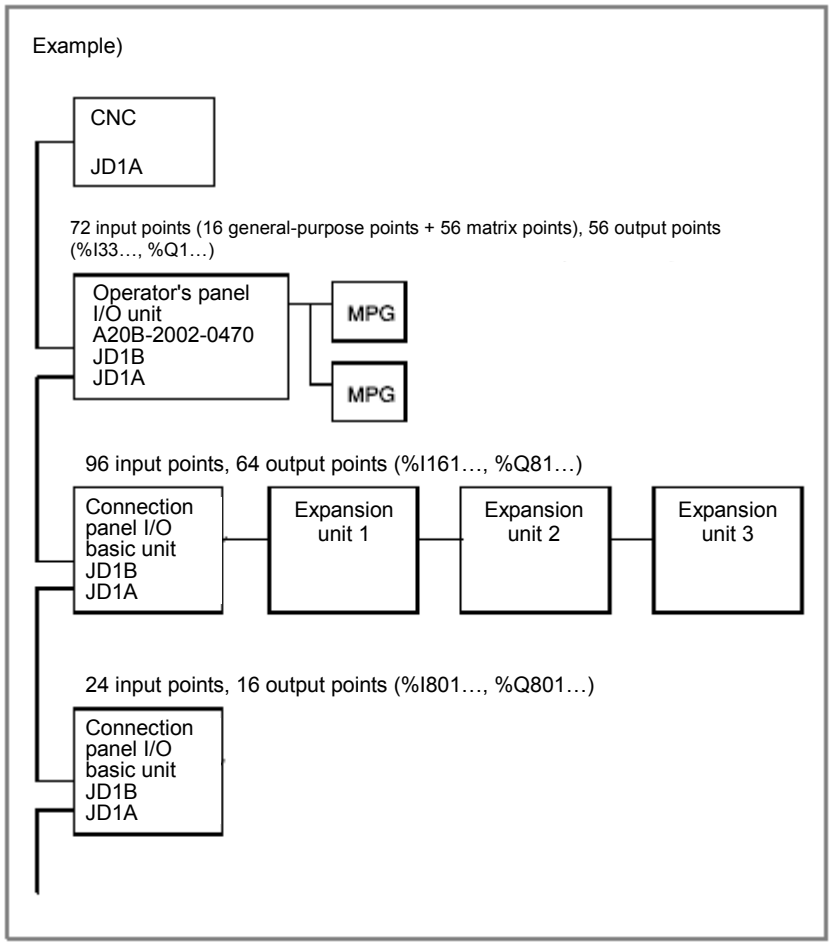
"FANUC I/O Unit-MODEL B Connection Manual" (B-62163E)

NOTE
For the setting of each I/O Unit-MODEL B unit and the specifications and connection of related I/O modules, refer to the hardware connection manual for each I/O device in addition to the above connection manual.

3.2.3 Assignment Method for Distribution I/O Connection Panel I/O Unit and Distribution I/O Operator's Panel I/O Units

For the I/O Link, when assigning connection information of a connection panel or operator's panel I/O module, set an I/O Link serial number (0 for the module nearest to the I/O Link master CNC) for the group number, always set 0 for the base number, and always set 1 for the slot number. When basic and expansion connection panel I/O modules are used, assign one connection information item for all modules in one I/O Link group. For a distribution I/O module unlike I/O Unit-MODEL A, the slot number need not be specified. For the module name used to set assignment data, see "Distribution I/O connection panel I/O modules" in Table 3.2 (c) to (d).
An example of assignment is shown below.

Example of assignment

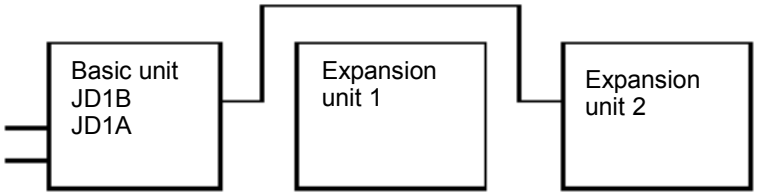


Input Address	Output Address	Group number	Module name
%I33	%Q1	0	A20B-2002-0470(3)
%I161	%Q81	1	BEEE
%I801	%Q801	2	B

Connection panel I/O units

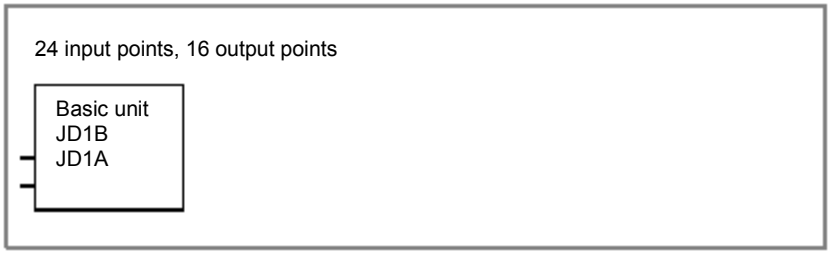
For signal mapping of connection panel I/O modules, refer to the connection manual (hardware) for the CNC used as the I/O Link master.
Assignment data is described below for each configuration of basic and expansion units.

⚠CAUTION
Always connect expansion units 1, 2, and 3 in this order closely when required. Any intermediate expansion unit cannot be skipped.



You may want to make the above setting so that expansion unit 1 is not mounted to connect it later and connection information of only expansion unit 2 is assigned, but the setting is disabled.

(1) Only basic unit

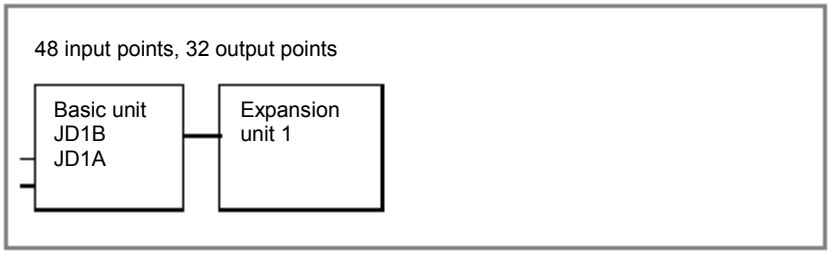


(a) When DO alarm detection is not used
- When no manual pulse generator is used
  Module Name = B

(b) When DO alarm detection is used
- Regardless of the number of manual pulse generators
  Module Name = BA

NOTE
"B" means a BASIC unit.
"A" means that alarm DO is used.

(2) Basic unit + expansion unit 1

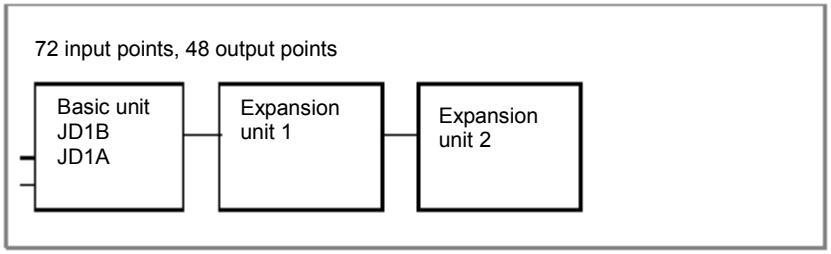


(a) When DO alarm detection is not used
- When no manual pulse generator is used
  Module Name = BE
- When one manual pulse generator is used
  Module Name = BEM

(b) When DO alarm detection is used
- When no manual pulse generator is used
  Module Name = BEA
- When one manual pulse generator is used
  Module Name = BEMA

NOTE
"B" means a BASIC unit.
"E" means an expansion unit.
"M" means that manual pulse generator is used.
"A" means that alarm DO is used.

(3) Basic unit + expansion unit 1 + expansion unit 2

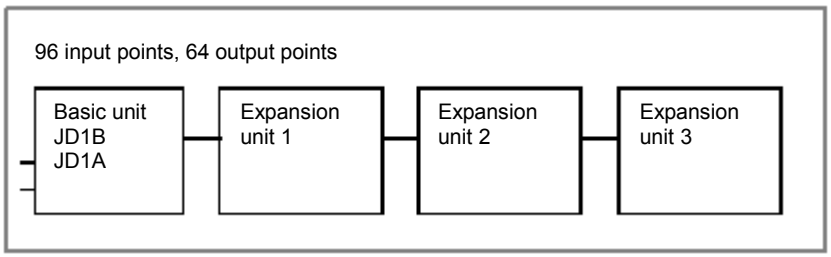


(a) When DO alarm detection is not used
- When no manual pulse generator is used
  Module Name = BEE
- When one manual pulse generator is used
  Module Name = BEEM
- When two manual pulse generators are used
  Module Name = BEEMM

(b) When DO alarm detection is used
- When no manual pulse generator is used
  Module Name = BEEA
- When one manual pulse generator is used
  Module Name = BEEMA
- When two manual pulse generators are used
  Module Name = BEEMMA

NOTE
"B" means a BASIC unit.
"E" means an expansion unit.
"M" means that manual pulse generator is used.
"A" means that alarm DO is used.

(4) Basic unit + expansion unit 1 + expansion unit 2 + expansion unit 3



(a) When DO alarm detection is not used

- When no manual pulse generator is used
  Module Name = BEEE
- When one manual pulse generator is used
  Module Name = BEEEM
- When two manual pulse generators are used
  Module Name = BEEEMM
- When three manual pulse generators are used
  Module Name = BEEEMMM

(b) When DO alarm detection is used

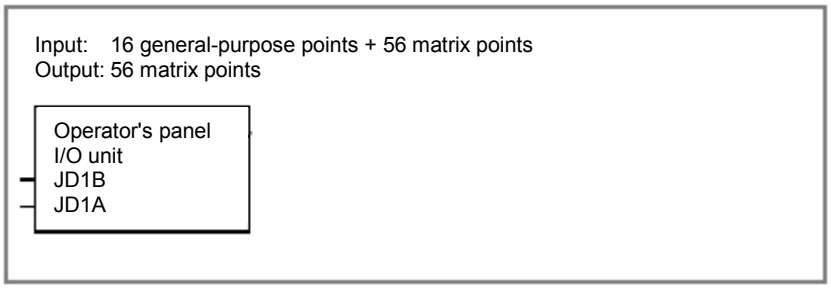
- When no manual pulse generator is used
  Module Name = BEEEA
- When one manual pulse generator is used
  Module Name = BEEEMA
- When two manual pulse generators are used
  Module Name = BEEEMMA
- When three manual pulse generators are used
  Module Name = BEEEMMMA

NOTE
"B" means a BASIC unit.
"E" means an expansion unit.
"M" means that manual pulse generator is used.
"A" means that alarm DO is used.

Operator's panel I/O units

For signal mapping of operator's panel I/O units, refer to the connection manual (hardware) for the CNC used as the I/O Link master.

(1) Operator's panel I/O unit (compatible with matrix input, A20B-2002-0470)



(a) When DO alarm detection is not used
- When no manual pulse generator is used
  Module Name = A20B-2002-0470 (1)
- When one manual pulse generator is used
  Module Name = A20B-2002-0470 (2)
- When two manual pulse generators are used
  Module Name = A20B-2002-0470 (3)
- When three manual pulse generators are used
  Module Name = A20B-2002-0470 (4)

(b) When DO alarm detection is used
- When no manual pulse generator is used
  Module Name = A20B-2002-0470 (5)
- When one manual pulse generator is used
  Module Name = A20B-2002-0470 (6)
- When two manual pulse generators are used
  Module Name = A20B-2002-0470 (7)
- When three manual pulse generators are used
  Module Name = A20B-2002-0470 (8)

(2) Operator's panel I/O unit (A20B-2002-0520, A20B-2002-0521)

(a) When DO alarm detection is not used

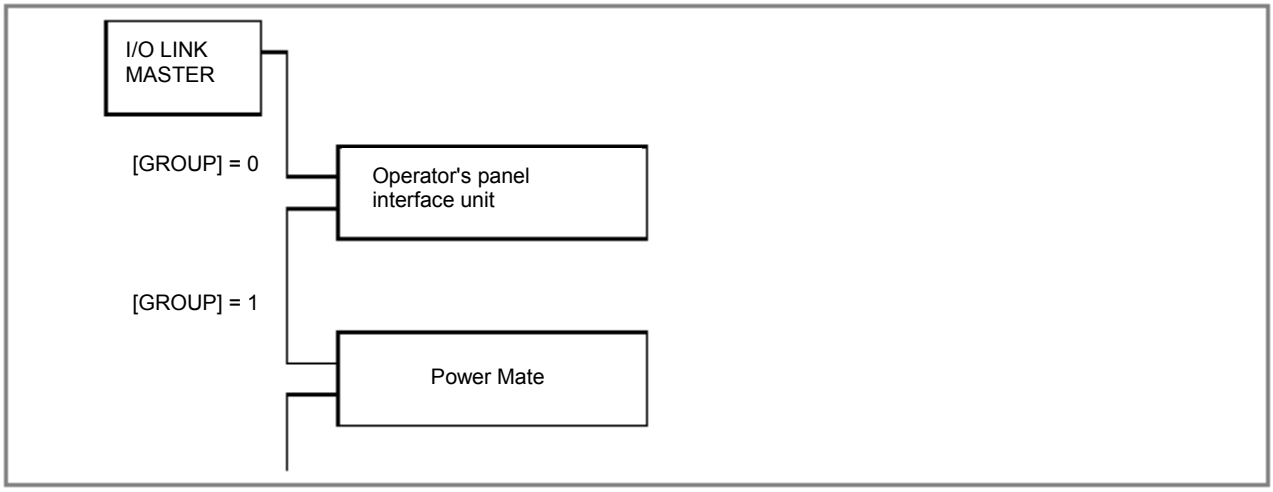
- When no manual pulse generator is used
  A20B-2002-0520 (1)
  A20B-2002-0521 (1)
- When one manual pulse generator is used
  A20B-2002-0520 (2)
- When two manual pulse generators are used
  A20B-2002-0520 (3)
- When three manual pulse generators are used
  A20B-2002-0520 (4)

(b) When DO alarm detection is used

- When no manual pulse generator is used
  A20B-2002-0520 (5)
- When one manual pulse generator is used
  A20B-2002-0520 (6)
- When two manual pulse generators are used
  A20B-2002-0520 (7)
- When three manual pulse generators are used
  A20B-2002-0520 (8)

3.2.4 Assignment Method for the Power Mate

To use Power Mate-MODEL D/H, Power Mate *i* MODEL-D/H, or I/O Link β amplifier as an I/O Link slave, assign its connection information on the I/O Link master.
On the I/O Link slave, assignment is not required because the addresses are fixed.
An example of connection is shown below.



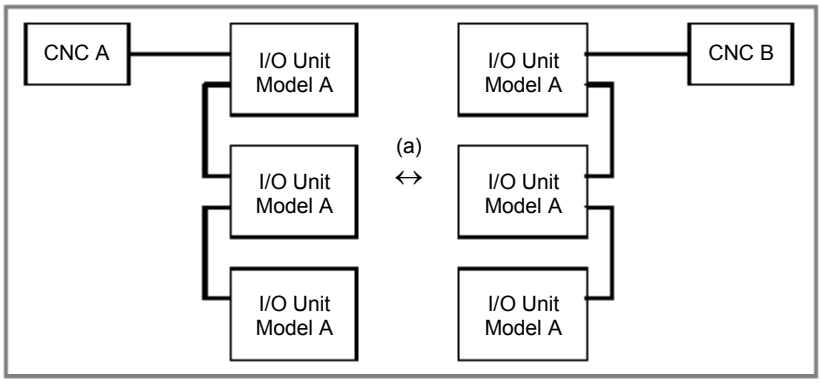
Assignment method

Number of input/output points (input/output)	Input device assignment name (module name)	Output device assignment name (module name)
32/32	FS04A (In)	FS04A (Out)
64/64	FS08A (In)	FS08A (Out)
128/128	FANUC CNC (OC02I)	FANUC CNC (OC02O)
256/256	FANUC CNC (OC03I)	FANUC CNC (OC03O)

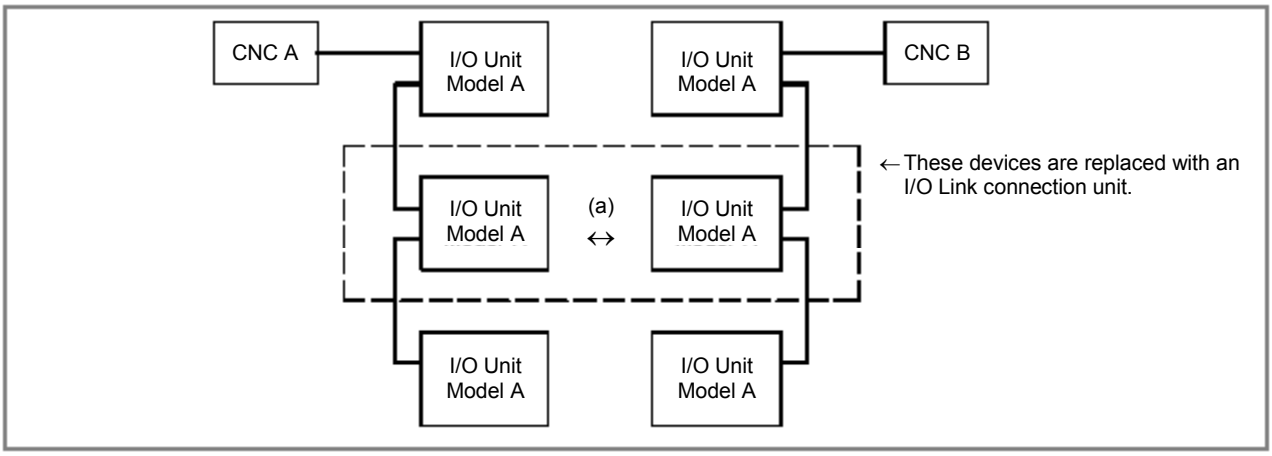
NOTE
1 Assign input and output module names with the same number of points.
2 For the I/O Link β amplifier, assign "A06B-6093-H151".

3.2.5 Assignment Method for I/O Link Connection Units

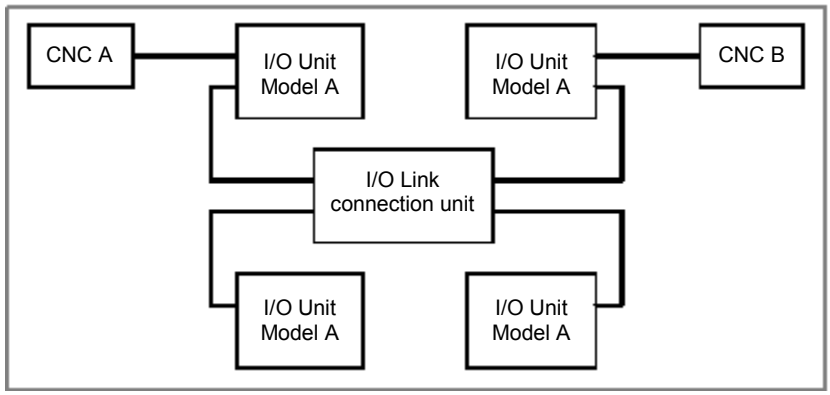
Conventionally, to exchange data between CNCs A and B, the devices indicated by (a) in the figure below must be connected. (Any I/O units can be used to exchange data.)



An I/O Link connection unit replaces the connected devices to eliminate cable connection and enable the power to each master or slave to be turned on and off independently.



Therefore, when an I/O Link connection unit is used, the connection is as follows.



Assignment method

Assignment data is determined according to the types of I/O devices replaced with an I/O Link connection unit.

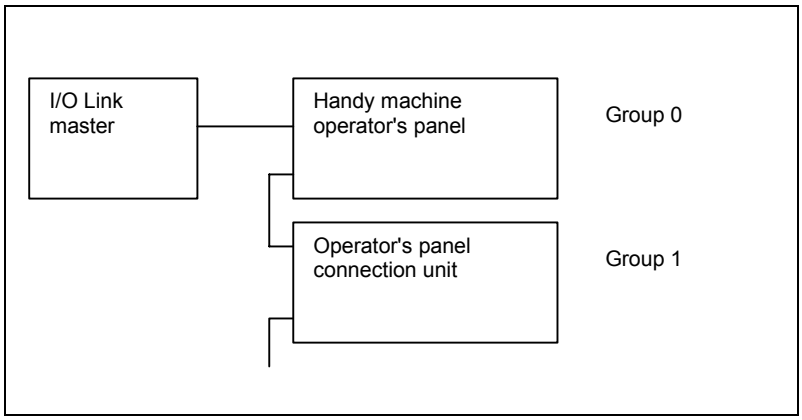
Occupied address (input/output)	Input / Output device assignment name (module name)
4/4	A20B-2000-0410(1)
	A20B-2000-0411(1)
	A20B-2000-0412(1)
8/8	A20B-2000-0410(2)
	A20B-2000-0411(2)
	A20B-2000-0412(2)
16/16	A20B-2000-0410(3)
	A20B-2000-0411(3)
	A20B-2000-0412(3)
32/32	A20B-2000-0410(4)
	A20B-2000-0410
	A20B-2000-0411(4)
	A20B-2000-0411
	A20B-2000-0412(4)
	A20B-2000-0412

Example of setting

NOTE
For details of the hardware connection method, particularly connection of a power supply, refer to the hardware connection manual for each related master/slave device.

3.2.6 Assignment Method for a Handy Machine Operator's Panel

An example of connecting a handy machine operator's panel to the I/O Link is shown below.



Assignment method

Assign 128 bit (16 bytes) to %I addresses and 256 bit (32 bytes) to %Q addresses contiguously starting from any address for each group. Set the group number used in the configuration for the group number and always set 0 for the base number. Set the slot number and assignment name as shown in the table below. The number of occupied input points for each group is 256 bit (32 bytes), which is the same as that of output points, because of limitations of the I/O Link.

[Examples of assigning %I addresses]

%I address	Unit number	Assignment name	Occupied address
%I(8×n+1)	0	##	4 bytes
%I(8×n+33)	1	#2	2 bytes
%I(8×n+49)	2	#2	2 bytes
%I(8×n+65)	3	#2	2 bytes
%I(8×n+81)	4	#2	2 bytes
%I(8×n+97)	5	#2	2 bytes
%I(8×n+113)	6	#2	2 bytes

[Examples of assigning %Q addresses]

%Q address	Unit number	Assignment name	Occupied address
%Q(8×n+1)	7	#2	2 bytes
%Q(8×n+17)	8	#2	2 bytes
%Q(8×n+33)	9	#2	2 bytes
%Q(8×n+49)	10	#2	2 bytes
%Q(8×n+65)	11	#2	2 bytes
%Q(8×n+81)	12	#2	2 bytes
%Q(8×n+97)	13	#2	2 bytes
%Q(8×n+113)	14	#2	2 bytes
%Q(8×n+129)	15	#2	2 bytes
%Q(8×n+145)	16	#2	2 bytes
%Q(8×n+161)	17	#2	2 bytes
%Q(8×n+177)	18	#2	2 bytes
%Q(8×n+193)	19	#2	2 bytes
%Q(8×n+209)	20	#2	2 bytes
%Q(8×n+225)	21	#2	2 bytes
%Q(8×n+241)	22	#2	2 bytes

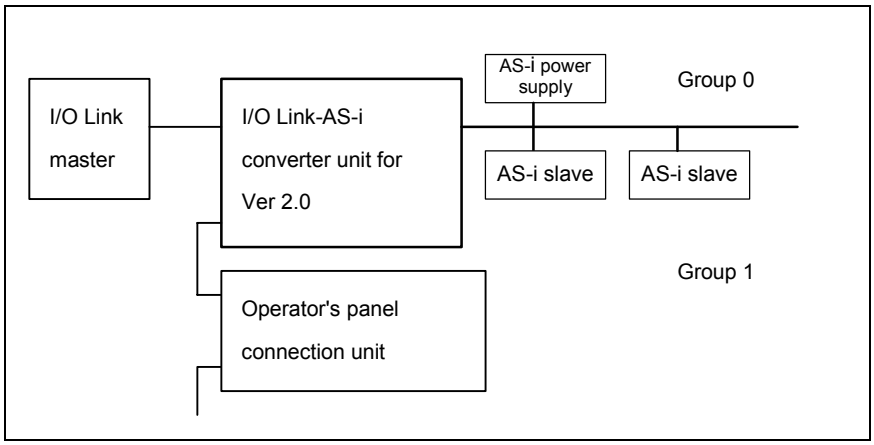
3.2.7 Assignment Method for an AS-i Converter Unit

An I/O Link-AS-i converter unit converts I/O from the I/O Link to the AS-Interface (called AS-i below) to enable the use of AS-i slave module DI/DO signals as a standalone unit.
The AS-i comes in two main versions: Ver 2.0 and Ver 2.1. Two types of I/O Link-AS-i converter units are available for each of these versions.
An I/O Link-AS-i converter unit for Ver 2.0 differs from that for Ver 2.1 in the following points.

	For Ver 2.0	For Ver 2.1
Number of input/output points	256 input points/256 output points	512 input points/512 output points
Occupied groups	1 group	Contiguous 2 groups

For each version, an example of connection is shown and the assignment method is described below.

Example of connection for Ver 2.0

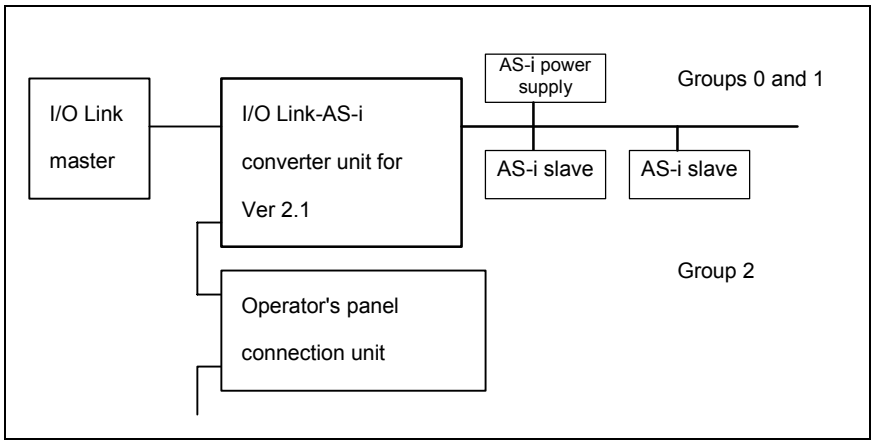


Assignment method for Ver 2.0

An assignment device name is selected from “Others”.
An I/O Link-AS-i converter unit occupies 256 points (32 bytes) for both input and output. Therefore, the assignment names are as follows.

Input device assignment name	Output device assignment name
32 byte Unit (In)	32 byte Unit (Out)

Example of connection for Ver 2.1



Assignment method for Ver 2.1

An assignment device name is selected from “Others”.
An I/O Link-AS-i converter unit occupies 512 points for both input and output, 256 points (32 bytes) per group. Therefore, the assignment names per group are the same as for an I/O Link-AS-i converter unit for Ver 2.0. Set the same assignment names for each occupied group number.

Group number	Input device assignment name	Output device assignment name
n	32 byte Unit (In)	32 byte Unit (Out)
n + 1	32 byte Unit (In)	32 byte Unit (Out)

NOTE
An I/O Link-AS-i converter unit for Ver 2.1 cannot be used as a converter unit for Ver 2.0 with assignment data for Ver 2.0.

3.3 I/O LINK CONNECTION CHECK FUNCTION

The I/O Link connection check function always checks whether the number of I/O Link groups defined in a sequence program is the same as that of actually connected groups. When the selectable I/O Link assignment function is used, the I/O Link connection check function compares the number of selected groups with that of connected groups.
If these numbers of groups do not match, the PMC alarm "ER97 IO LINK FAILURE (CHx yyGROUP)" is issued. For action to be taken, see Section B.1.

NOTE
1 All I/O devices connected to the channel in which this alarm occurs are not linked.
2 The ladder program is executed regardless of whether this alarm occurs.

The execution of this function can be controlled using keep relay %SK51.

%SK51
0: Enables the I/O Link connection check function. (Initial value)
1: Disables the I/O Link connection check function.

⚠CAUTION
If I/O devices are linked in the status in which an I/O device error or I/O device connection error occurs or the setting of an I/O device is changed due to an unintentional operation, the machine may not operate normally. This function can always be operated to detect an I/O device error at power-on. To troubleshoot problems with I/O devices easily, it is recommended that keep relay %SK51 be set to the initial value (0).

4 LD INSTRUCTION GROUP

The following explains the Ladder Diagram(LD) instructions used by a PMC sequence program.

PMC Instruction Set

Number	Class	Instruction
1.	Contact, Coil	--| |--, --|/|--, |+|--- --( ), --(/), --(S), --(R),--(+)
2.	Timer, counter	ONDTR_, TMR_, OFDT_, UPCTR_, DNCTR_
3.	Math operation	ADD_, SUB_, MUL_, DIV_, MOD_, ABS_, SQRT_
4.	Relational	EQ_, NE_, GT_, GE_, LT_, LE_, RANGE_
5.	Bit operation	AND_, OR_, XOR_, NOT_, SHIFTL_, SHIFTR_, ROL_, ROR_, BIT_TEST_, BIT_SET_, BIT_CLR_, BIT_POS_, BIT_SEQ, MASK_COMP_
6.	Data move	MOVE_, SWAP_, BLK_CLR_,SHFR_
7.	Data table function	ARRAY_MOV_, SEARCH_EQ_, SEARCH_NE_, SEARCH_GT_, SEARCH_GE_, SEARCH_LT_, SEARCH_LE_
8.	Conversion	_TO_BCD2, _TO_BCD4, _TO_BCD8, BCD2_TO_, BCD4_TO_, BCD8_TO_, _TO_SINT, _TO_USINT, _TO_INT, _TO_UINT, _TO_DINT, _TO_UDINT
9.	Program flow	CALL, END, MCR, ENDMCR, MCRN, ENDMCRN, JUMPN, LABELN, COMMENT
10.	PMC operation	PMC_ADD_BCD2, PMC_ADD_BCD4,PMC_ADD_BCD8, PMC_SUB_BCD2, PMC_SUB_BCD4, PMC_SUB_BCD8, PMC_MUL_BCD2,PMC_MUL_BCD4,PMC_MUL_BCD8, PMC_DIV_BCD2, PMC_DIV_BCD4, PMC_DIV_BCD8, PMC_MOD_BCD2,PMC_MOD_BCD4,PMC_MOD_BCD8, R_TRIG, F_TRIG, PMC_DECODE_, PMC_EVPAR, PMC_ODPAR, PMC_WINDOW, PMC_EXIN, PMC_AXCTL

4.1 CONTACTS & COILS

Contacts

Instruction	Function	Data type
--| |--	Normally open contact (A contact)	BOOL
--|/|--	Normally close contact (B contact)	BOOL
|+|---	Continuous contact	BOOL

Coils

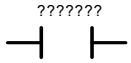
Instruction	Function	Data type
--( )--	Coil	BOOL
--(/)--	Reverse-wound coil	BOOL
--(S)--	Set coil	BOOL
--(R)--	Reset coil	BOOL
---(+)	Continuous coil	BOOL

4.1.1 Normally Open Contacts (A Contacts)

Function

While the corresponding reference is set to ON, a normally open contact allows power to flow.

Format

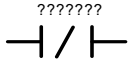


4.1.2 Normally Close Contacts (B Contacts)

Function

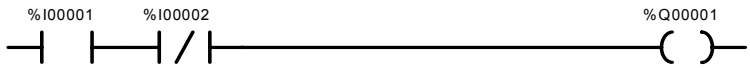
While the corresponding reference is set to OFF, a normally close contact allows power to flow.

Format



Example

While %I00001 is set to ON, but %I00002 is OFF, %Q00001 is set to ON.

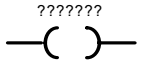


4.1.3 Coils

Function

While the coil is receiving a power flow, it sets the corresponding reference to ON.

Format

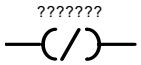


4.1.4 Reverse-wound Coils

Function

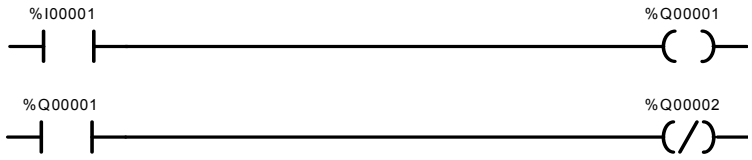
While the coil is not receiving a power flow, it sets the corresponding reference to ON.

Format



Example

While %I00001 is set to OFF (because %Q00001 is OFF), %Q00002 is set to ON.

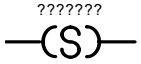


4.1.5 Setting Coil (SET)

Function

While the coil is receiving a power flow, it sets the corresponding reference to ON.
This ON state is maintained until reset by the reset coil (RESET), described below.

Format

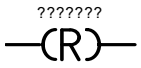


4.1.6 Reset Coil (RESET)

Function

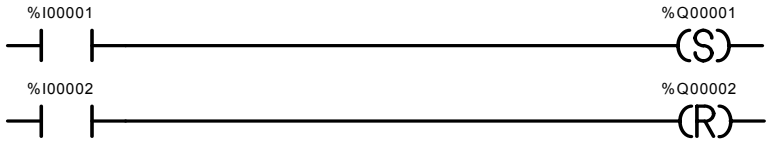
While the coil is receiving a power flow, it sets the corresponding reference to OFF.
This OFF state is maintained until the ON state is set by the set coil (SET), described above.

Format



Example

When %I00001 is ON, %Q00001 is set to ON. (This ON state is maintained until reset by the reset coil (RESET).)
Similarly, when %I00002 is ON, %Q00002 is set to OFF. (This OFF state is maintained until the ON state is set by the set coil (SET).)



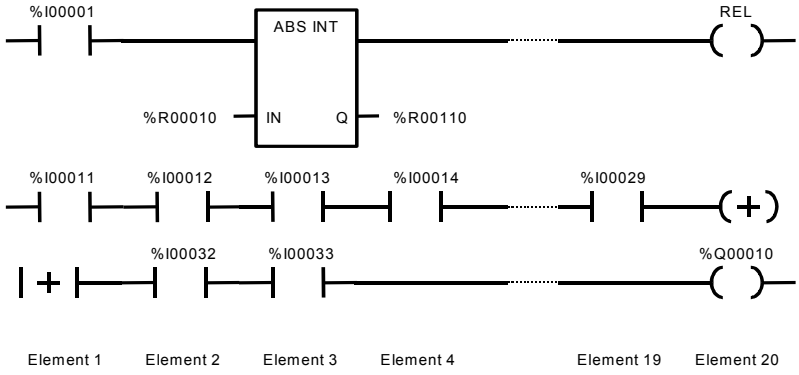
4.1.7 Continuous Contacts & Continuous Coils

Function

Indicates the continuation of a ladder when the ladder for one net can not be described within 20 elements.

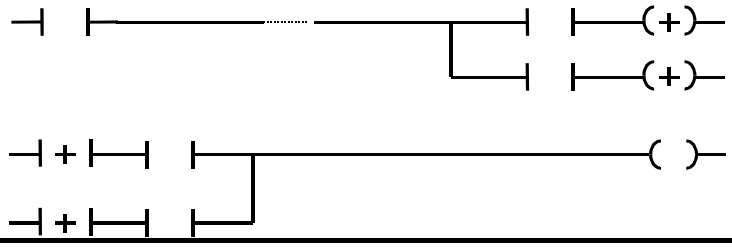
Format

Example



NOTE

Multiple continuous contacts cannot be used in parallel, as shown below.



4.2 TIMERS & COUNTERS

Timers

Instruction	Function	Parameter	Data type	Controlled I/O
ONDTR_(unit)	Integrated on delay timer	ADRS, PV	ADRS = one-dimensional WORD array of 3 words PV = INT	EN, ENO, R
TMR_(unit)	On delay timer	ADRS, PV	ADRS = one-dimensional WORD array of 3 words PV = INT	EN, ENO
OFDT_(unit)	Off delay timer	ADRS, PV	ADRS = one-dimensional WORD array of 3 words PV = INT	EN, ENO

Counters

Instruction	Function	Parameter	Data type	Controlled I/O
UPCTR	Up counter	ADRS, PV	ADRS = one-dimensional WORD array of 3 words PV = INT	EN, ENO, R
DNCTR	Down counter	ADRS, PV	ADRS = one-dimensional WORD array of 3 words PV = INT	EN, ENO, R

4.2.1 ONDTR

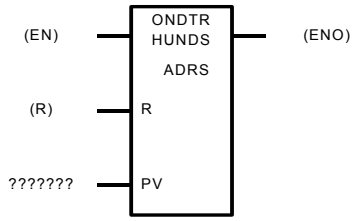
Function

The retentive On-Delay Stopwatch Timer (ONDTR) increments while it receives power flow and holds its value when power flow stops. When this function first receives power flow, it starts accumulating time (Current Value (CV)). When this timer is encountered in the ladder logic, its CV is updated. When the CV equals or exceeds Preset Value (PV), output Q is energized, regardless of the state of the power flow input.
The instruction differs depending on the units in which the timer is set.

Integrated on timer

Name	Setting units
ONDTR_THOUS	0.001 s
ONDTR_HUNDS	0.01 s
ONDTR_TENTHS	0.1 s
ONDTR_SECS	1 s
ONDTR_TENSEC	10 s
ONDTR_MIN	1 min

Format



Parameters

Parameter	Data type	Meaning
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: Current value (CV) Word 2: Preset value (PV) Word 3: Control word
EN	BOOL	Timer start
R	BOOL	Timer reset
PV	INT	Set time (Preset Value) The time required to actually reach time up will be this value to which the set units are applied.
ENO	BOOL	Time up

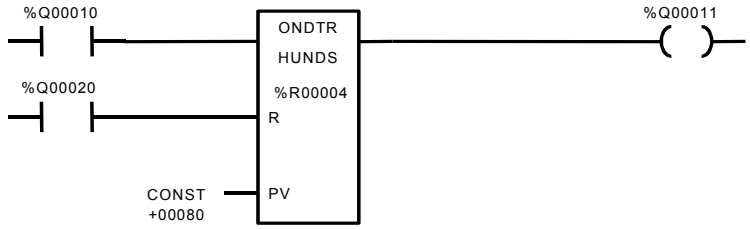
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
ADRS				o		o		
EN	o	o	o	o	o			
R	o	o	o	o	o			
PV		o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

%Q00011 is turned ON 0.8 s after %Q00010 is set to ON. (%Q00011 is turned OFF when %Q00020 is turned ON.)



4.2.2 TMR

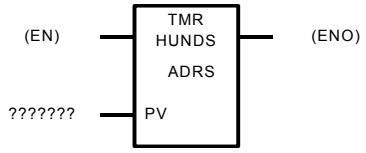
Function

The On-Delay Timer (TMR) increments while it receives power flow and resets to zero when power flow stops. The timer passes power after the specified interval PV (Preset Value) has elapsed, as long as power is received. When the On Delay Timer function receives power flow, the timer starts accumulating time (Current Value (CV)). CV is updated when it is encountered in the logic to reflect the total elapsed time the timer has been enabled since it was last reset. This update occurs as long as the power flow input remains ON. When CV equals or exceeds the Preset Value (PV), the function begins passing power flow to the right. When the power flow input transitions from ON to OFF, the timer stops accumulating time, CV is reset to zero, and Q is turned off.
Output Q is energized when TMR is enabled and PV ≤ CV.

On delay timer

Name	Setting units
TMR_THOUS	0.001 s
TMR_HUNDS	0.01 s
TMR_TENTHS	0.1 s
TMR_SECS	1 s
TMR_TENSEC	10 s
TMR_MIN	1 min

Format



Parameters

Parameter	Data type	Meaning
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: Current value (CV) Word 2: Preset value (PV) Word 3: Control word
EN	BOOL	Timer start
PV	INT	Set time (Preset Value) The time required to actually reach time up will be this value to which the set units are applied.
ENO	BOOL	Time up

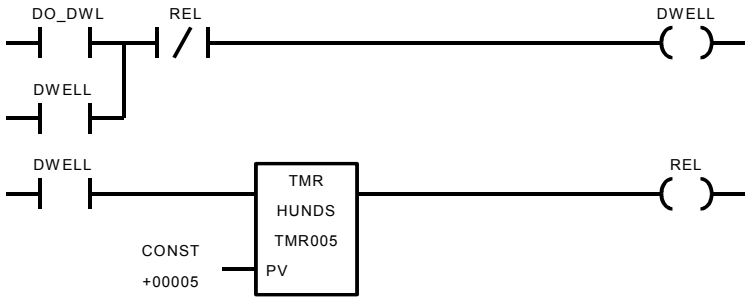
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
ADRS				o		o		
EN	o	o	o	o	o			
PV		o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

While REL is OFF, if DO_DWL is turned ON, DWELL is also turned ON. (The latch circuit causes the ON state to be maintained even if DO_DWL is subsequently set to OFF.)
Then, 0.05 s later, TMR005 causes REL to turn ON, disconnects the latch circuit, and sets DWELL to OFF. As a result, REL is turned OFF, and the initial status is restored.



4.2.3 OFDT

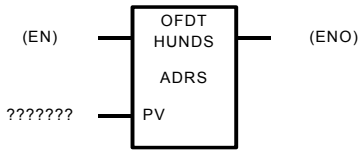
Function

The Off-Delay Timer (OFDT) increments while power flow is off, and the timer's Current Value (CV) resets to zero when power flow is on. OFDT passes power until the specified interval PV (Preset Value) has elapsed.
When OFDT first receives power flow, CV is set to zero and the timer passes power to the right, even if PV=0. The output remains on as long as OFDT receives power flow. If OFDT stops receiving power flow from the left, it continues to pass power to the right and OFDT starts accumulating time in CV. Each time OFDT is invoked with the power flow logic set to OFF, CV is updated to reflect the elapsed time since the timer was turned off. OFDT continues passing power flow to the right until CV=PV. When CV=PV, OFDT stops passing power flow to the right and OFDT stops accumulating time. CV remains equal to PV and never exceeds PV. If PV=0, the timer stops passing power flow to the right as soon as it stops receiving power flow.

Off delay timer

Name	Setting units
OFDT_THOUS	0.001 s
OFDT_HUNDS	0.01 s
OFDT_TENTHS	0.1 s
OFDT_SECS	1 s
OFDT_TENSEC	10 s
OFDT_MIN	1 min

Format



Parameters

Parameter	Data type	Meaning
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: Current value (CV) Word 2: Preset value (PV) Word 3: Control word
EN	BOOL	Timer start
PV	INT	Set time (Preset Value) The time required to actually reach time up will be this value to which the set units are applied.
ENO	BOOL	Time up

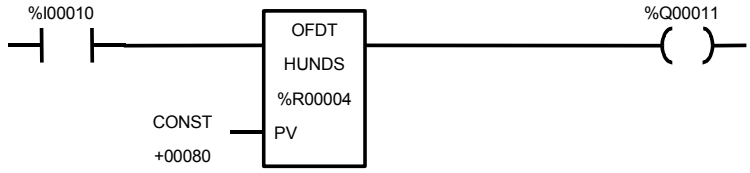
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
ADRS				o		o		
EN	o	o	o	o	o			
PV		o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00010 is turned ON, %Q00011 is turned ON, and that state is maintained.
Then, 0.8 s after %I00010 has been set to OFF, %Q00011 is turned off.

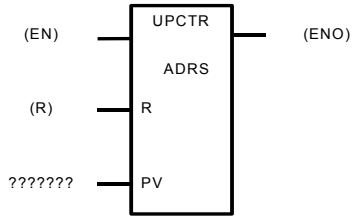


4.2.4 UPCTR

Function

The Up Counter (UPCTR) function counts up to the Preset Value (PV). The range is 0 to +32,767 counts. When the Current Value (CV) of the counter reaches 32,767, it remains there until reset. When the UPCTR reset is ON, CV resets to 0. Each time the power flow input transitions from OFF to ON, CV increments by 1. CV can be incremented past the Preset Value (PV). The output is ON whenever CV ≥ PV. The output stays ON until the R input receives power flow to reset CV to zero.

Format



Parameters

Parameter	Data type	Meaning
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: Current Value (CV) Word 2: Preset Value (PV) Word 3: Control word
EN	BOOL	Counter start
R	BOOL	Reset
PV	INT	Set value (Preset Value)
ENO	BOOL	Reach to the Preset Value

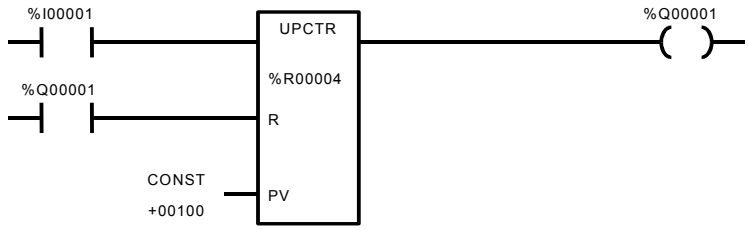
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
ADRS				o		o		
EN	o	o	o	o	o			
R	o	o	o	o	o			
PV		o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

Every time %I00001 goes from OFF to ON, the count is incremented by one.
Once CV exceeds 100, %Q00001 is set to ON. (Also, at this time, CV is reset to 0.)

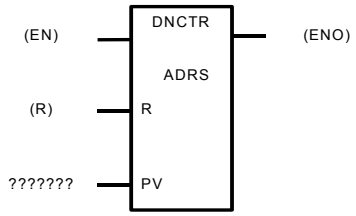


4.2.5 DNCTR

Function

The Down Counter (DNCTR) function counts down from a preset value. The minimum Preset Value (PV) is zero; the maximum present value is +32,767 counts. When the Current Value (CV) reaches the minimum value, -32,768, it stays there until reset. When DNCTR is reset, CV is set to PV. When the power flow input transitions from OFF to ON, CV is decremented by one. The output is ON whenever CV ≤ 0.

Format



Parameters

Parameter	Data type	Meaning
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: Current Value (CV) Word 2: Preset Value (PV) Word 3: Control word
EN	BOOL	Counter start
R	BOOL	Reset
PV	INT	Set value (Preset Value)
ENO	BOOL	Current Value falls below 0

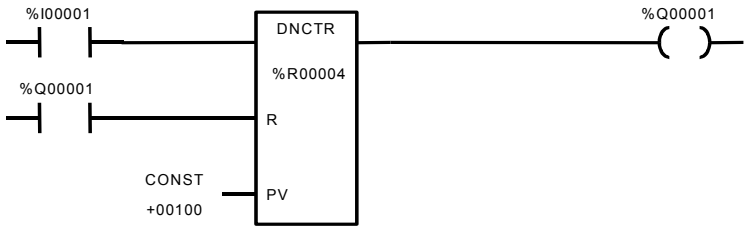
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
ADRS				o		o		
EN	o	o	o	o	o			
R	o	o	o	o	o			
PV	o	o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

Every time %I00001 goes from OFF to ON, the count is decremented by one.
Once the value of CV falls below 0, %Q00001 is set to ON. (Also, at this time, CV is reset to 100.)



4.3 MATH OPERATIONS

Math operation functions

Instruction	Function	Parameters	Data type	Controlled I/O
ADD_(type)	Addition	IN1, IN2, Q	ANY_INT	EN, ENO
SUB_(type)	Subtraction	IN1, IN2, Q	ANY_INT	EN, ENO
MUL_(type)	Multiplication	IN1, IN2, Q	ANY_INT	EN, ENO
DIV_(type)	Division	IN1, IN2, Q	ANY_INT	EN, ENO
MOD_(type)	Mode	IN1, IN2, Q	ANY_INT	EN, ENO
ABS_(type)	Absolute value	IN, Q	SINT, INT, DINT	EN, ENO
SQRT_(type)	Square root	IN, Q	ANY_INT	EN, ENO

4.3.1 ADD_(type)/SUB_(type)/MUL_(type)/DIV_(type)

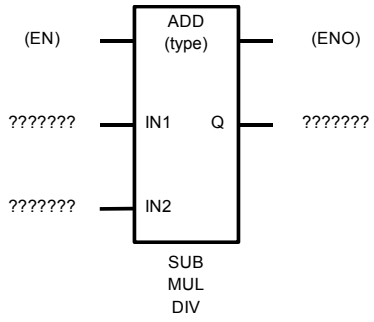
Function

When EN is set to ON and is receiving the power flow, ADD, SUB, MUL, and DIV perform, on input parameters IN1 and IN2, addition (IN1 + IN2), subtraction (IN1 - IN2), multiplication (IN1 X IN2), and division (IN1 ÷ IN2), respectively. The result is output to output parameter Q.
IN1, IN2, and Q must all be of the same data type.
If the arithmetic operation results in an overflow, the maximum value for that data type is set in the output reference. For a signed data type, the direction of the overflow is indicated, based on the sign. When the operation can be completed without any overflow, ENO is set to ON.

Addition/subtraction/multiplication/division

Name	Explanation
ADD_SINT	Addition of SINT data
ADD_USINT	Addition of USINT data
ADD_INT	Addition of INT data
ADD_UINT	Addition of UINT data
ADD_DINT	Addition of DINT data
ADD_UDINT	Addition of UDINT data
SUB_SINT	Subtraction of SINT data
SUB_USINT	Subtraction of USINT data
SUB_INT	Subtraction of INT data
SUB_UINT	Subtraction of UINT data
SUB_DINT	Subtraction of DINT data
SUB_UDINT	Subtraction of UDINT data
MUL_SINT	Multiplication of SINT data
MUL_USINT	Multiplication of USINT data
MUL_INT	Multiplication of INT data
MUL_UINT	Multiplication of UINT data
MUL_DINT	Multiplication of DINT data
MUL_UDINT	Multiplication of UDINT data
DIV_SINT	Division of SINT data
DIV_USINT	Division of USINT data
DIV_INT	Division of INT data
DIV_UINT	Division of UINT data
DIV_DINT	Division of DINT data
DIV_UDINT	Division of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
IN1	ANY_INT	Input data 1
IN2	ANY_INT	Input data 2
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	ANY_INT	Result

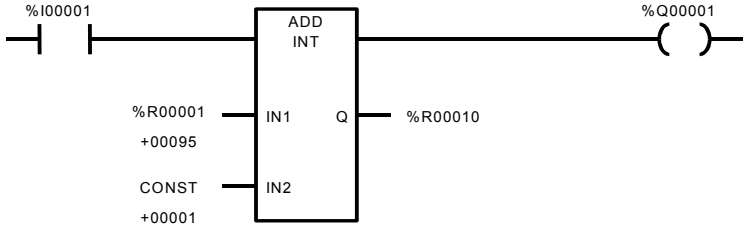
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, 1 is added to the value of %R00001, 95, to give an output of 96 to %R00010. %Q00001 is set to ON, and notification of the operation ending without an overflow is posted.



4.3.2 MOD_(type)

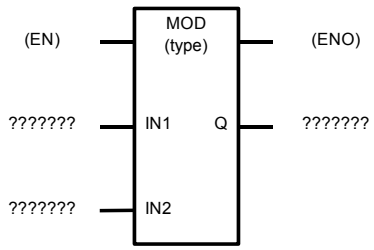
Function

When EN is set to ON and is receiving the power flow, MOD performs, on input parameters IN1 and IN2, division (IN1 ÷ IN2) and outputs the remainder to output parameter Q.
IN1, IN2, and Q must all be of the same data type. Q will have the same sign as IN1.
When the operation terminates without an overflow (division by 0), ENO is set to ON.

The Modulo Division (MOD) function

Name	Explanation
MOD_SINT	MOD of SINT data
MOD_USINT	MOD of USINT data
MOD_INT	MOD of INT data
MOD_UINT	MOD of UINT data
MOD_DINT	MOD of DINT data
MOD_UDINT	MOD of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
IN1	ANY_INT	Input data 1
IN2	ANY_INT	Input data 2
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	ANY_INT	Result

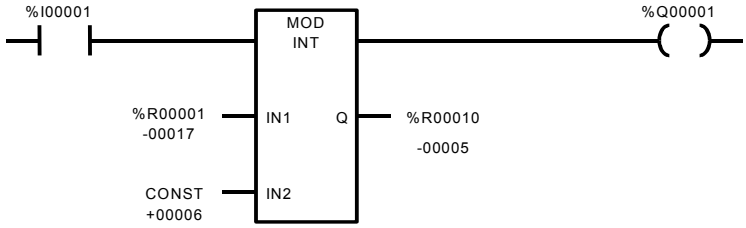
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, division (-17 ÷ 6) is performed and the remainder - 5 is output to %R00010.



4.3.3 ABS_(type)

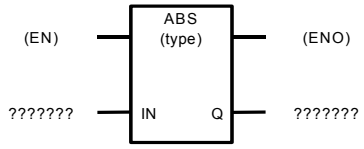
Function

When EN is set to ON and is receiving the power flow, ABS takes the absolute value of input parameter IN and outputs it to output parameter Q.

ABS_(type) instructions

Name	Explanation
ABS_SINT	The absolute value of SINT data
ABS_INT	The absolute value of INT data
ABS_DINT	The absolute value of DINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution
IN	SINT,INT, DINT	Input data
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	SINT,INT, DINT	Absolute value

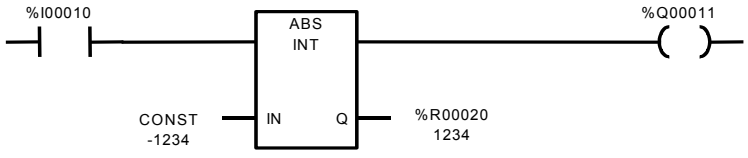
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the absolute value of -1234, namely 1234, is output to %R00020.



4.3.4 SQRT_(type)

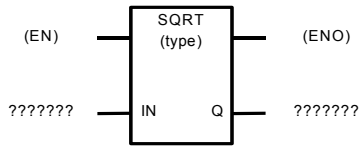
Function

When EN is set to ON and is receiving the power flow, SQROOT takes the square root of input parameter IN and outputs it to output parameter Q.
IN and Q must both be of the same data type.

SQRT_(type) instructions

Name	Explanation
SQRT_SINT	The square root of SINT data
SQRT_USINT	The square root of USINT data
SQRT_INT	The square root of INT data
SQRT_UINT	The square root of UINT data
SQRT_DINT	The square root of DINT data
SQRT_UDINT	The square root of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution
IN	ANY_INT	Input data
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	ANY_INT	Square root

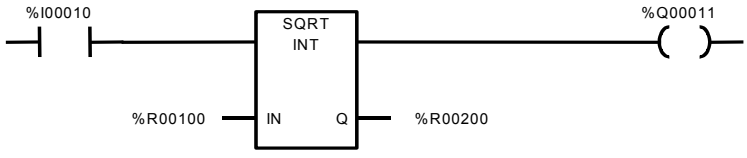
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the square root of the value of %R00100 is output to %R00200.



4.4 RELATIONAL OPERATIONS

Comparison instructions

Instruction	Function	Parameter	Data type	Controlled I/O
EQ_(type)	Equal to	IN1, IN2	BYTE, WORD, DWORD, ANY_INT	EN, Q
NE_(type)	Not equal to	IN1, IN2	BYTE, WORD, DWORD, ANY_INT	EN, Q
GT_(type)	Greater than (IN1 > IN2)	IN1, IN2	ANY_INT	EN, Q
GE_(type)	Greater than or equal to (IN1 ≥ IN2)	IN1, IN2	ANY_INT	EN, Q
LT_(type)	Less than (IN1 < IN2)	IN1, IN2	ANY_INT	EN, Q
LE_(type)	Less than or equal to (IN1 ≤ IN2)	IN1, IN2	ANY_INT	EN, Q
RANGE_(type)	Range compare	L1, L2, IN	ANY_INT	EN, Q

4.4.1 EQ_(type)/NE_(type)

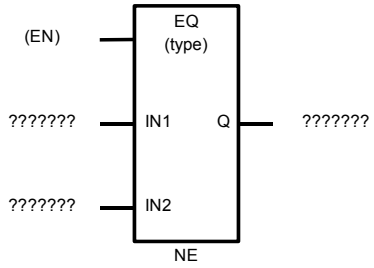
Function

When EN is set to ON and is receiving the power flow, the signs of input parameters IN1 and IN2 are checked for equivalence (IN1 = IN2?) or non-equivalence (IN1≠IN2?). When the result of the comparison is true, power is allowed to flow and Q is set to ON.
IN1 and IN2 must both be of the same data type.

EQ_(type)/NE_(type) instructions

Name	Explanation
EQ_BYTE	Equal to BYTE data
EQ_WORD	Equal to WORD data
EQ_DWORD	Equal to DWORD data
EQ_SINT	Equal to SINT data
EQ_USINT	Equal to USINT data
EQ_INT	Equal to INT data
EQ_UINT	Equal to UINT data
EQ_DINT	Equal to DINT data
EQ_UDINT	Equal to UDINT data
NE_BYTE	Not equal to BYTE data
NE_WORD	Not equal to WORD data
NE_DWORD	Not equal to DWORD data
NE_SINT	Not equal to SINT data
NE_USINT	Not equal to USINT data
NE_INT	Not equal to INT data
NE_UINT	Not equal to UINT data
NE_DINT	Not equal DINT data
NE_UDINT	Not equal UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of comparison
IN1	BYTE, WORD, DWORD, ANY_INT	Input data 1
IN2	BYTE, WORD, DWORD, ANY_INT	Input data 2
Q	BOOL	Result of comparison (set to ON when condition is satisfied)

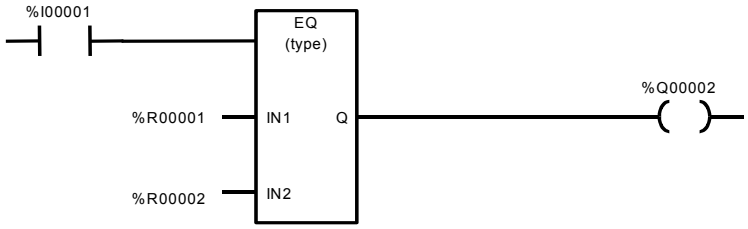
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
Q	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the values of %R00001 and %R00002 are compared and, if they are equivalent, %Q00002 is set to ON.



4.4.2 GT_(type)/GE_(type)/LT_(type)/LE_(type)

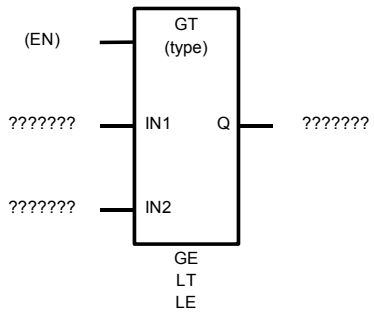
Function

When EN is set to ON and is receiving the power flow, the values of input parameters IN1 and IN2 are compared (IN1 > IN2?/IN1 ≥ IN2?/IN1 < IN2?/IN1 ≤ IN2?). If the result of the comparison is true, power is allowed to flow and Q is set to ON.
IN1 and IN2 must both be of the same data type.

GT_(type)/GE_(type)/LT_(type)/LE_(type) instructions

Name	Explanation
GT_SINT	Greater than (SINT data)
GT_USINT	Greater than (USINT data)
GT_INT	Greater than (INT data)
GT_UINT	Greater than (UINT data)
GT_DINT	Greater than (DINT data)
GT_UDINT	Greater than (UDINT data)
GE_SINT	Greater than or equal(SINT data)
GE_USINT	Greater than or equal(USINT data)
GE_INT	Greater than or equal(INT data)
GE_UINT	Greater than or equal(UINT data)
GE_DINT	Greater than or equal(DINT data)
GE_UDINT	Greater than or equal(UDINT data)
LT_SINT	Less than (SINT data)
LT_USINT	Less than (USINT data)
LT_INT	Less than (INT data)
LT_UINT	Less than (UINT data)
LT_DINT	Less than (DINT data)
LT_UDINT	Less than (UDINT data)
LE_SINT	Less than or equal (SINT data)
LE_USINT	Less than or equal (USINT data)
LE_INT	Less than or equal (INT data)
LE_UINT	Less than or equal (UINT data)
LE_DINT	Less than or equal (DINT data)
LE_UDINT	Less than or equal (UDINT data)

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of comparison
IN1	ANY_INT	Input data 1
IN2	ANY_INT	Input data 2
Q	BOOL	Result of comparison (set to ON when condition is satisfied)

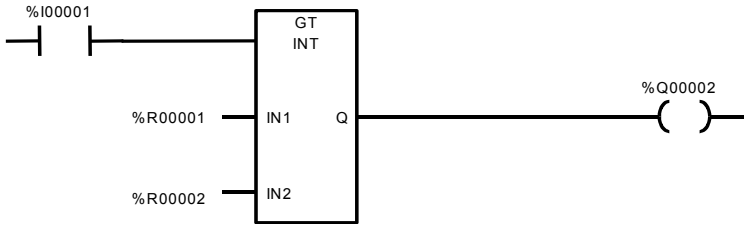
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
Q	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the values of %R00001 and %R00002 are compared. When the former is larger than the latter, %Q00002 is set to ON.



4.4.3 RANGE_(type)

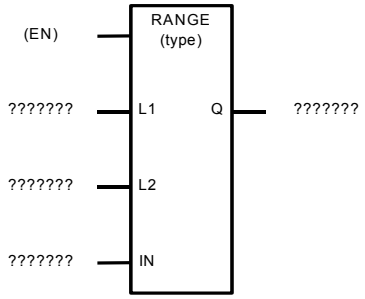
Function

When EN is set to ON and is receiving the power flow, RANGE compares input parameter IN with the range of L1 and L2. When L1 ≤ IN ≤ L2, or L2 ≤ IN ≤ L1, power is allowed to flow and Q is set to ON.
L1, L2 and IN must all be of the same data type.

RANGE_(type) instructions

Name	Explanation
RANGE_SINT	Range compare of SINT data
RANGE_USINT	Range compare of USINT data
RANGE_INT	Range compare of INT data
RANGE_UINT	Range compare of UINT data
RANGE_DINT	Range compare of DINT data
RANGE_UDINT	Range compare of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of comparison
L1	ANY_INT	Threshold 1
L2	ANY_INT	Threshold 2
IN	ANY_INT	Comparison data
Q	BOOL	Set to ON when L1 ≤ IN ≤ L2, or L2 ≤ IN ≤ L1

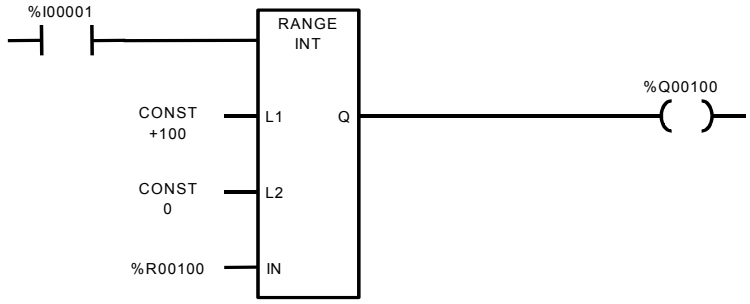
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
L1		o	o	o	o	o	o	
L2		o	o	o	o	o	o	
IN		o	o	o	o	o	o	
Q	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, and the value of %R00100 is between 0 and 100, %Q00100 is set to ON.



4.5 BIT OPERATIONS

Bit operation functions

Instruction	Function	Parameter	Data type	Controlled I/O
AND_(type)	Logical AND	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
OR_(type)	Logical OR	IN1, IN2, Q	BYTE,WORD, DWORD	EN, ENO
XOR_(type)	Exclusive logical OR	IN1, IN2, Q	BYTE,WORD, DWORD	EN, ENO
NOT_(type)	Logical NOT	IN, Q	BYTE,WORD, DWORD	EN, ENO
SHIFTL_(type)	Left shift	LEN, IN, N, Q	LEN = The number of data IN/Q = BYTE, WORD, DWORD N = INT	EN, B1, B2
SHIFTR_(type)	Right shift	LEN, IN, N, Q	LEN = The number of data IN/Q = BYTE, WORD, DWORD N = INT	EN, B1, B2
ROL_(type)	Rotate left	LEN, IN, N, Q	LEN = The number of data IN/Q = BYTE, WORD, DWORD N = INT	EN, ENO
ROR_(type)	Rotate right	LEN, IN, N, Q	LEN = The number of data IN/Q = BYTE, WORD, DWORD N = INT	EN, ENO
BIT_TEST_(type)	Bit test	IN, BIT	IN/Q = BYTE, WORD, DWORD BIT = INT	EN, Q
BIT_SET_(type)	Bit set	LEN, IN, BIT	LEN = The number of data IN/Q = BYTE, WORD, DWORD BIT = INT	EN, ENO
BIT_CLR_(type)	Bit clear	LEN, IN, BIT	LEN = The number of data IN/Q = BYTE, WORD, DWORD BIT = INT	EN, ENO
BIT_POS_(type)	Bit position	LEN, IN, POS	LEN = The number of data IN = BYTE, WORD, DWORD POS = INT	EN, ENO
BIT_SEQ	Sequence a string of bit values.	ADRS, LEN, N, ST	ADRS = one-dimensional WORD array of 3 words LEN = The number of data N = INT ST = BYTE	EN, ENO, R, DIR
MASK_COMP_(type)	Compare the bits in two strings	LEN, IN1, IN2, M, BIT, Q, BN	LEN = The number of data IN1/IN2/M/Q = BYTE, WORD, DWORD BIT/BN = UINT	EN, MC

4.5.1 AND_(type)/OR_(type)

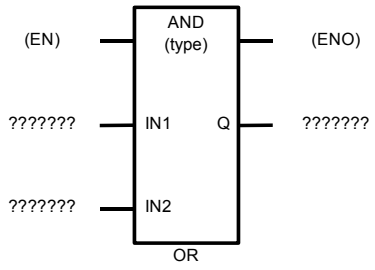
Function

When EN is set to ON and is receiving the power flow, AND/OR performs a logical AND/logical OR operation on each bit of the corresponding bit strings in input parameters IN1 and IN2. The result is output to the corresponding bit of output parameter Q.
IN1, IN2 and Q must all be of the same data type.

AND_(type)/OR_(type) instructions

Name	Explanation
AND_BYTE	Logical AND of BYTE data
AND_WORD	Logical AND of WORD data
AND_DWORD	Logical AND of DWORD data
OR_BYTE	Logical OR of BYTE data
OR_WORD	Logical OR of WORD data
OR_DWORD	Logical OR of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of logical operation
IN1	BYTE, WORD, DWORD	Input data 1
IN2	BYTE, WORD, DWORD	Input data 2
ENO	BOOL	Returns the value of EN
Q	BYTE, WORD, DWORD	Result of operation

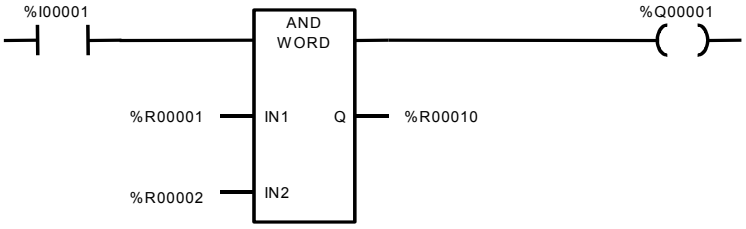
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, a logical operation is performed on each corresponding bit of the bit strings of input parameters %R00001 and %R00002. The result is output to the corresponding bit of the bit string of %R00010.



4.5.2 XOR_(type)

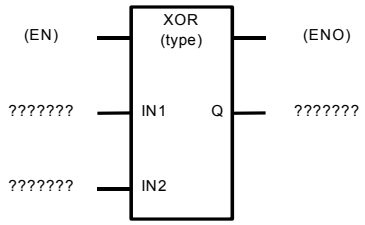
Function

When EN is set to ON and is receiving the power flow, XOR performs an exclusive logical OR operation on each corresponding bit of the bit strings in input parameters IN1 and IN2. The result is output to the corresponding bit of output parameter Q.
IN1, IN2 and Q must all be of the same data type.

XOR_(type) instructions

Name	Explanation
XOR_BYTE	Exclusive logical OR of BYTE data
XOR_WORD	Exclusive logical OR of WORD data
XOR_DWORD	Exclusive logical OR of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of logical operation
IN1	BYTE, WORD, DWORD	Input data 1
IN2	BYTE, WORD, DWORD	Input data 2
ENO	BOOL	Returns the value of EN
Q	BYTE, WORD, DWORD	Result of operation

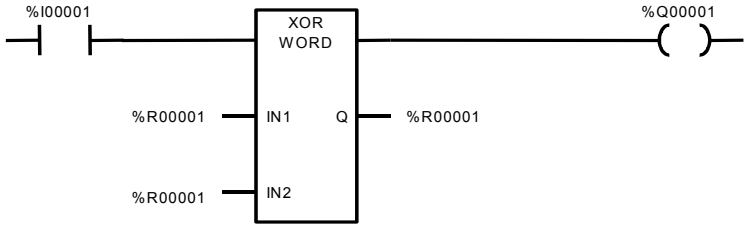
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, a logical operation is performed on each corresponding bit of the bit strings of %R00001 and %R00001. The result is output to the corresponding bit of the bit string of %R00001. (In this case, all the bits of %R00001 are cleared to 0.)



4.5.3 NOT_(type)

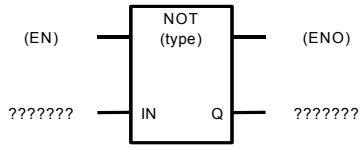
Function

When EN is set to ON and is receiving the power flow, NOT logically inverts each bit of a bit string in input parameter IN. The result is output to the corresponding bit of output parameter Q.
IN and Q must both be of the same data type.

NOT_(type) instructions

Name	Explanation
NOT_BYTE	Logical NOT of BYTE data
NOT_WORD	Logical NOT of WORD data
NOT_DWORD	Logical NOT of DWORD data

Format



Parameter

Parameter	Data type	Meaning
EN	BOOL	Execution of logical operation
IN	BYTE, WORD, DWORD	Input data
ENO	BOOL	Returns the value of EN
Q	BYTE, WORD, DWORD	Result of operation

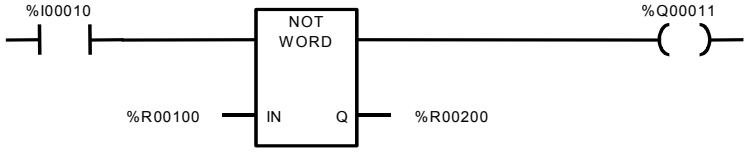
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, each bit of the bit string of %R00100 is logically inverted. The result is output to the corresponding bit of the bit string of %R00200.



4.5.4 SHIFTL_(type)/SHIFTR_(type)

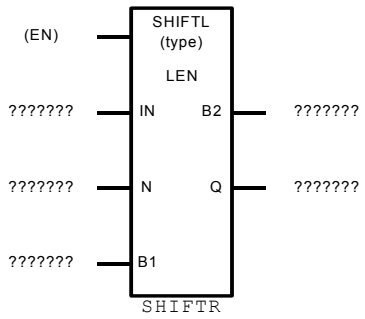
Function

When EN is set to ON and is receiving the power flow, SHIFTL/SHIFTR left-shifts/right-shifts the bit string of input parameter IN by an amount given by input parameter N.
The last bit to overflow is output to output parameter B2, and the value of input parameter B1 (0 or 1) is replaced with an empty bit. The shifted bit string is output to output parameter Q.
IN and Q must both be of the same data type. When the value of N is less than 0, or greater than the number of bits in the bit string of IN, SHIFTL/SHIFTR stops the power flow.

SHIFTL_(type)/SHIFTR_(type) instructions

Name	Explanation
SHIFTL_BYTE	Left shift of BYTE data
SHIFTL_WORD	Left shift of WORD data
SHIFTL_DWORD	Left shift of DWORD data
SHIFTR_BYTE	Right shift of BYTE data
SHIFTR_WORD	Right shift of WORD data
SHIFTR_DWORD	Right shift of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of shift
LEN	Constant	The number of data in the string. 1 ≤ LEN ≤ 256
IN	BYTE, WORD, DWORD	Shifted data
N	INT	Among of shift (bits)
B1	BOOL	Value placed in bit emptied by shift
B2	BOOL	Last bit to overflow
Q	BYTE, WORD, DWORD	Result of operation

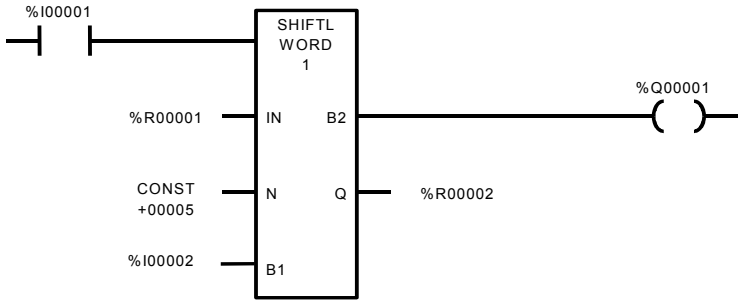
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
N		o	o	o	o	o	o	
B1		o	o	o	o	o		
B2	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the bit string of %R00001 is shifted five bits to the left, and the left-shifted bit string is output to %R00002. The last bit to overflow is output to %Q00001, and the value of input parameter %I00002 is replaced with an empty bit.
When the bit string of %R00001 is 0011100011011011, and %I00002 is set to ON, 0001101101111111 is output to %R00002 and 1 is output to %Q00001.



4.5.5 ROL_(type)/ROR_(type)

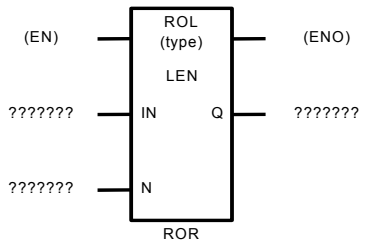
Function

When EN is set to ON and is receiving the power flow, ROL/ROR left rotates/right rotates the bit string of input parameter IN by an amount equal to the value of input parameter N.
The rotated bit string is output to output parameter Q.
IN and Q must be both of the same data type. When the value of N is 0 or less, or larger than the number of bits in the bit string of IN, bit string rotation is not performed, and ROL/ROR is not performed.

ROL_(type)/ROR_(type) instructions

Name	Explanation
ROL_BYTE	Rotate left of BYTE data
ROL_WORD	Rotate left of WORD data
ROL_DWORD	Rotate left of DWORD data
ROR_BYTE	Rotate right of BYTE data
ROR_WORD	Rotate right of WORD data
ROR_DWORD	Rotate right of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
LEN	Constant	The number of data in the string. 1 ≤ LEN ≤ 256
IN	BYTE, WORD, DWORD	Shifted data
N	INT	Amount of shift (bits)
ENO	BOOL	Normal end flag
Q	BYTE, WORD, DWORD	Result of operation

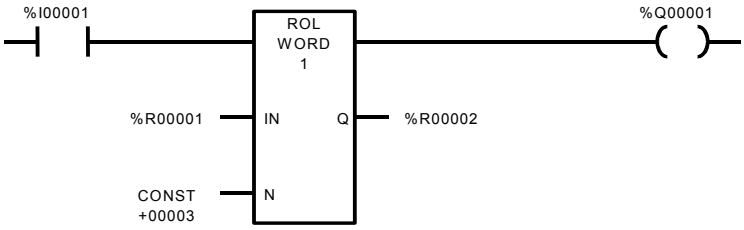
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
N		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the bit string of %R00001 is rotated left by 3 bits, then the rotated bit string is output to %R00002.
When the bit string of %R00001 is 0011100011011011, 1100011011011001 is output to %R00002.



4.5.6 BIT_TEST_(type)

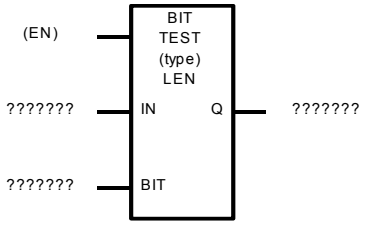
Function

When EN is set to ON and is receiving the power flow and BIT_TEST outputs to output parameter Q, the bit state (0 or 1) specified with input parameter BIT in the bit string of input parameter IN.
When the value of BIT is 1 or more and not less than the number of bits in the bit string of IN, Q is set to OFF.

BIT_TEST_(type) instructions

Name	Explanation
BIT_TEST_BYTE	Bit test of BYTE data
BIT_TEST_WORD	Bit test of WORD data
BIT_TEST_DWORD	Bit test of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
LEN	Constant	The number of data in the string. 1 ≤ LEN ≤ 256
IN	BYTE, WORD, DWORD	Input data
BIT	INT	Test bit
Q	BOOL	Result of operation

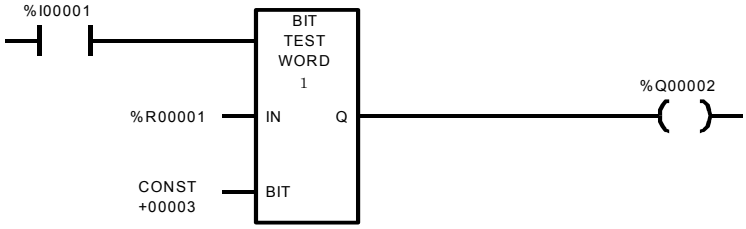
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
BIT		o	o	o	o	o	o	
Q	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the bit state of the 3rd bit of %R00001 is output to %Q00002.
When the bit string of %R00001 is 0011100011011011, 0 is output to %Q00002.



4.5.7 BIT_SET_(type)/BIT_CLR_(type)

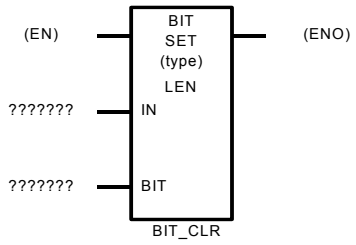
Function

When EN is set to ON and is receiving the power flow, BIT_SET/BIT_CLR sets to 1 or clears to 0 that bit, specified with input parameter BIT, within the bit string of input parameter IN.
When the value of BIT is less than one or more than the number of bits in the IN bit string, ENO is set to OFF.

BIT_SET_(type)/BIT_CLR_(type) instructions

Name	Explanation
BIT_SET_BYTE	Bit set of BYTE data
BIT_SET_WORD	Bit set of WORD data
BIT_SET_DWORD	Bit set of DWORD data
BIT_CLR_BYTE	Bit clear of BYTE data
BIT_CLR_WORD	Bit clear of WORD data
BIT_CLR_DWORD	Bit clear of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
LEN	Constant	The number of data in the string. 1 ≤ LEN ≤ 256
IN	BYTE, WORD, DWORD	Input data
BIT	INT	Set (clear) bit
ENO	BOOL	Normal end flag

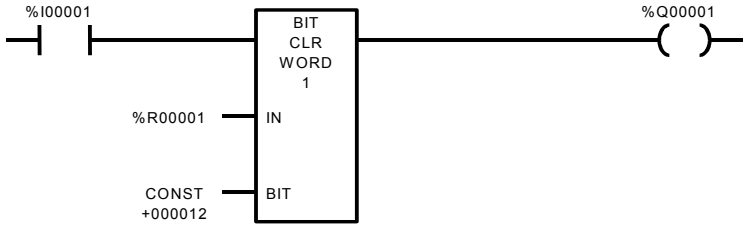
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN			o	o	Note	o		
BIT		o	o	o	o	o	o	
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON the twelfth bit of the bit string of %R00001 is cleared to 0. When the bit string of %R00001 is 0011100011011011, the twelfth bit is cleared to 0 and 0011000011011011 is output.



4.5.8 BIT_POS_(type)

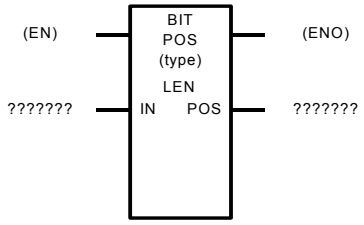
Function

When EN is set to ON and is receiving the power flow, BIT_POS searches the bit string of input parameter IN for bits that are set to 1, then outputs the positions of those bits to output parameter POS. When no bit can be found, 0 is output.
When BIT_POS finds a bit that has been set to 1, it sets Q to ON.

BIT_POS_(type) instructions

Name	Explanation
BIT_POS_BYTE	Bit position of BYTE data
BIT_POS_WORD	Bit position of WORD data
BIT_POS_DWORD	Bit position of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution
LEN	Constant	The number of data in the string. 1≤ LEN ≤256
IN	BYTE, WORD, DWORD	Input data
ENO	BOOL	Returns the value of EN
POS	INT	Position of first non-zero bit to be detected

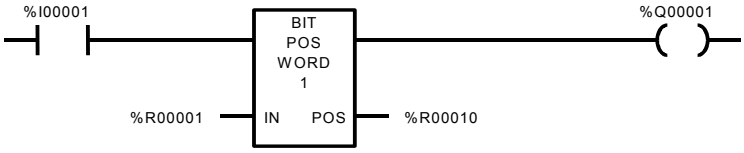
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO			o	o	Note			o
POS	o		o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the bit string of %R00001 is searched for bits that are set to 1.
When the bit string of %R00001 is 0011100011011011, 1 is output to %R00010 and %Q00002 is set to ON.

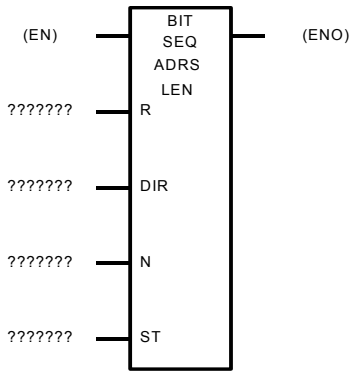


4.5.9 BIT_SEQ

Function

The Bit Sequencer (BITSEQ) function performs a bit sequence shift through a series of contiguous bits. The operation of BITSEQ depends on the value of the reset input (R) and both the current value and previous value of the enabling power flow input (EN):
The reset input (R) overrides the enabling power flow (EN) and always resets the sequencer. When R is active, the current step number is set to the value of the optional N operand. If the user did not specify N, the step number is set to 1. All of the bits in the bit sequencer, ST, are set to 0, except for the bit pointed to by the current step, which is set to 1.
When EN is active and Reset is not active and the previous EN was OFF, the bit pointed to by the current step number is cleared. The current step number is incremented or decremented, based on the direction (DIR) operand. Then the bit pointed to by the new step number is set to 1.
When the step number is being incremented and it goes outside the range of (1 ≤ step number ≤ LEN operand), it is set back to 1.
When the step number is being decremented and it goes outside the range of (1≤ step number ≤ LEN operand), it is set to LEN.

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
ADRS	one-dimensional WORD array of 3 words	ADRS is the beginning address of a three-word WORD array: Word 1: current step number Word 2: length of sequence in bits Word 3: control word, which tracks the status of the last enabling power flow and the status of the power flow to the right.
LEN	Constant	The length. The number of bits in the bit sequencer, ST, that BITSEQ will Step Through. 1 ≤ LEN ≤ 256

Parameter	Data type	Meaning
R	BOOL	Reset
DIR	BOOL	Direction of the bit sequencer
N	INT	Step number
ST	BYTE	The first byte of the bit sequencer
ENO	BOOL	Returns the value of EN

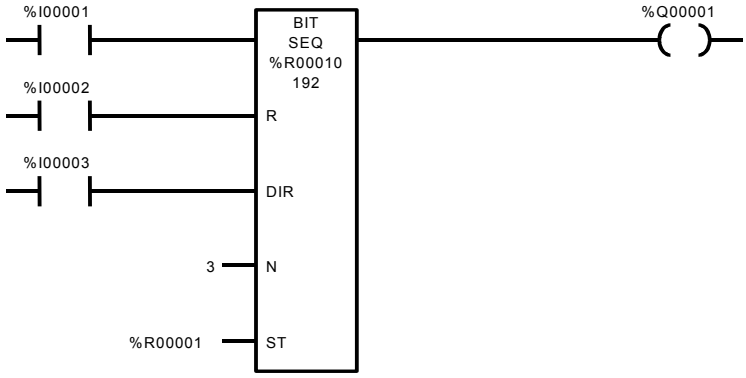
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
R	o	o	o	o	o			
DIR	o	o	o	o	o			
N		o	o	o		o	o	o
ST		o	o	o	o	o		o
ENO	o		o	o	Note	o		o

NOTE
Only %SK can be specified.

Example

The Bit Sequencer operates on register memory %R00001. Its static data is stored in registers %R00010, %R00011, and %R00012. When %I00002 is active, the sequencer is reset and the current step is set to step number 3, as specified in N. The third bit of %R00001 is set to one and the other seven bits are set to zero.
When %I00001 is active and %I00002 is not active, the bit for step number 3 is cleared and the bit for step number 2 or 4 (depending on whether DIR is energized) is set.



4.5.10 MASK_COMP_(type)

Function

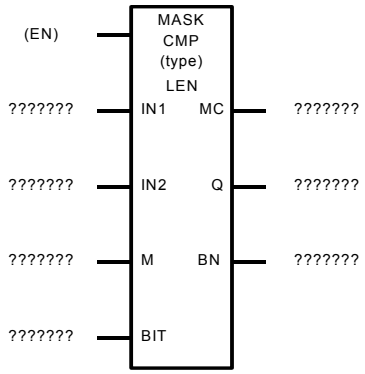
The Masked Compare (function compares the contents of two separate bit strings. It provides the ability to mask selected bits.
Tip: Input string 1 might contain the states of outputs such as solenoids or motor starters. Input string 2 might contain their input state feedback, such as limit switches or contacts.
When the function receives power flow, it begins comparing the bits in the first string with the corresponding bits in the second string. Comparison continues until a miscompare is found or until the end of the string is reached.
The BIT input stores the bit number where the next comparison should start (a 0 indicates the first bit in the string). The BN output stores the bit number where the last comparison occurred (where a 1 indicates the first bit in the string). Using the same reference for BIT and BN causes the compare to start at the next bit position after a miscompare; or, if all bits compared successfully upon the next invocation of the function block, the compare starts at the beginning.
Tip: If you want to start the next comparison at some other location in the string, you can enter different references for BIT and BN. If the value of BIT is a location that is beyond the end of the string, BIT is reset to 0 before starting the next comparison.
The function passes power flow whenever it receives power. The other outputs of the function depend on the state of the corresponding mask bit.
If all corresponding bits in strings IN1 and IN2 match, the function sets the miscompare output MC to 0 and BN to the highest bit number in the input strings. The comparison then stops. On the next invocation of a Masked Compare Word, it is reset to 0.
If a Miscompare is found, i.e., when the two bits currently being compared are not the same, the function checks the correspondingly numbered bit in string M (the mask). If the mask bit is a 1, the comparison continues until it reaches another miscompare or the end of the input strings.
If a miscompare is detected and the corresponding mask bit is a 0, the function does the following:

1) Sets the corresponding mask bit in M to 1.
2) Sets the miscompare (MC) output to 1.
3) Updates the output bit string Q to match the new content of mask string M
4) Sets the bit number output (BN) to the number of the miscompared bit.
5) Stops the comparison.

MASK_COMP_(type) instructions

Name	Explanation
MASK_COMP_BYTE	Compare the bits in two BYTE data
MASK_COMP_WORD	Compare the bits in two WORD data
MASK_COMP_DWORD	Compare the bits in two DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation.
LEN	Constant	The number of data in either compared string. 1 ≤ LEN ≤ 8192(BYTE) 1 ≤ LEN ≤ 4096(WORD) 1 ≤ LEN ≤ 2048(DWORD)
IN1	BYTE, WORD, DWORD	Input data 1.
IN2	BYTE, WORD, DWORD	Input data 2.
M	BYTE, WORD, DWORD	The bit string mask.
BIT	UINT	The bit number where the next comparison should start.
Q	BYTE, WORD, DWORD	The output copy of the compare mask bit string.
BN	UINT	The number of the bit where the latest miscompare occurred.
MC	BOOL	Miscompare output.

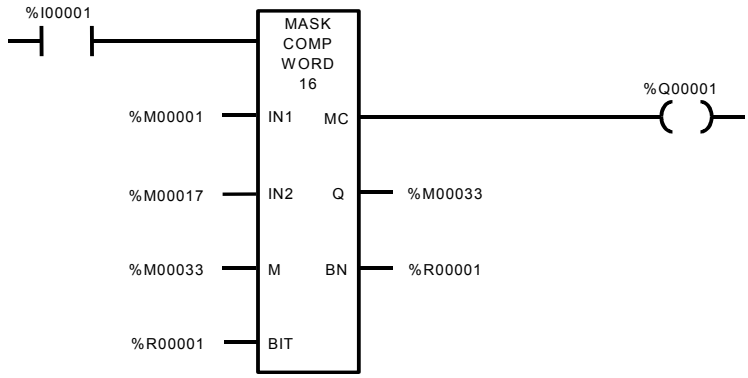
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN1		o	o	o	o	o		
IN2		o	o	o	o	o		
M		o	o	o	o	o		
BIT		o	o	o	o		o	
Q			o	o		o		
BN			o	o		o		
MC	o		o	o	Note	o		

NOTE
Only %SK can be specified.

Example

After first scan, the Masked Compare Word function executes. %M00001 through %M00016 is compared with %M00017 through %M00032. %M00033 through %M00048 contains the mask value. The value in %R00001 determines the bit position in the two input strings where the comparison starts.



Before the function block is executed, the contents of the above references are:

(IN1) - %M00001 =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

(IN2) - %M00017 =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1

(M/Q) - %M00033 =

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(BIT/BN) - %R00001 = 0

(MC) - %Q00001 = OFF

The contents of these references after the function block is executed are as follows:

(IN1) - %M00001 =

0	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0

(IN2) - %M00017 =

0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1

(M/Q) - %M00033 =

0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1

(BIT/BN) - %R00001 = 9

(MC) - %Q00001 = ON

4.6 DATA MOVE

Data transmission functions

Instruction	Function	Parameter	Data type	Controlled I/O
MOVE_(type)	Data movement	IN, Q	ANY	EN, ENO
SWAP_(type)	Swap	IN, Q	WORD, DWORD	EN, ENO
BLK_CLR_(type)	Block Clear	IN	BYTE, WORD, DWORD ANY_INT	EN, ENO
SHFR_(type)	Shift Register	LEN, IN, ST, Q	LEN = The number of data IN/ST/Q = ANY_BIT	EN, ENO, R

4.6.1 MOVE_(type)

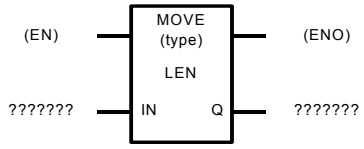
Function

When EN is set to ON and is receiving the power flow, MOVE copies the data in input parameter IN to output parameter Q.

MOVE_(type) instructions

Name	Explanation
MOVE_BOOL	Data movement of BOOL data.
MOVE_BYTE	Data movement of BYTE data
MOVE_WORD	Data movement of WORD data
MOVE_DWORD	Data movement of DWORD data
MOVE_SINT	Data movement of SINT data
MOVE_USINT	Data movement of USINT data
MOVE_INT	Data movement of INT data
MOVE_UINT	Data movement of UINT data
MOVE_DINT	Data movement of DINT data
MOVE_UDINT	Data movement of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution
LEN	Constant	The length of IN; the number of data to move. If IN is a constant and Q is BOOL, then 1 ≤ LEN ≤ 16; otherwise, 1 ≤ LEN ≤ 256
IN	ANY	Input data
ENO	BOOL	Returns the value of EN
Q	ANY	Destination address

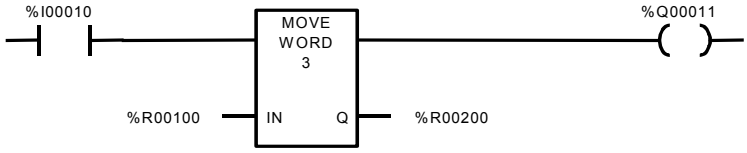
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the value of %R00100 is copied to %R00200.



4.6.2 SWAP_(type)

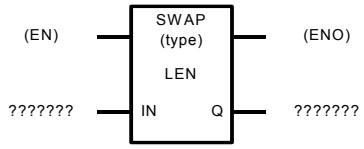
Function

When EN is set to ON and is receiving the power flow, SWAP swaps either WORD data in input parameter IN in BYTE units, or DWORD data in WORD units, and outputs the result to output parameter Q.

SWAP_(type) instructions

Name	Explanation
SWAP_WORD	Swap of WORD data
SWAP_DWORD	Swap of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of logical operation
LEN	Constant	The number of data in the string. 1 ≤ LEN ≤ 256
IN	WORD, DWORD	Input data
ENO	BOOL	Returns the value of EN
Q	WORD, DWORD	Result of operation

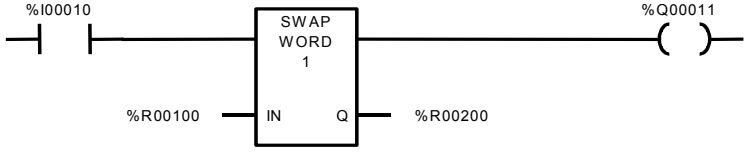
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the data of %R00100 is swapped, and the result is output to %R00200. When %R00100 is 0011100011011011, WORD data 1101101100111000 that has been swapped in BYTE units is output to %R00200.



4.6.3 BLK_CLR_(type)

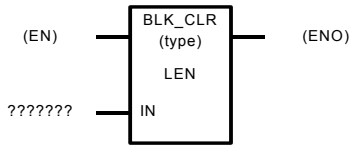
Function

When the Block Clear (BLK_CLR) function receives power flow, it fills the specified block of data with zeros, beginning at the reference specified by IN. The function passes power to the right whenever power is received.

BLK_CLR_(type) instructions

Name	Explanation
BLK_CLR_BYTE	Block clear of BYTE data
BLK_CLR_WORD	Block clear of WORD data
BLK_CLR_DWORD	Block clear of DWORD data
BLK_CLR_SINT	Block clear of SINT data
BLK_CLR_USINT	Block clear of USINT data
BLK_CLR_INT	Block clear of INT data
BLK_CLR_UINT	Block clear of UINT data
BLK_CLR_DINT	Block clear of DINT data
BLK_CLR_UDINT	Block clear of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of logical operation
LEN	Constant	The number of data to be cleared, starting at the IN location. 1 ≤ LEN ≤ 256
IN	BYTE,WORD, DWORD, ANY_INT	Input data
ENO	BOOL	Returns the value of EN

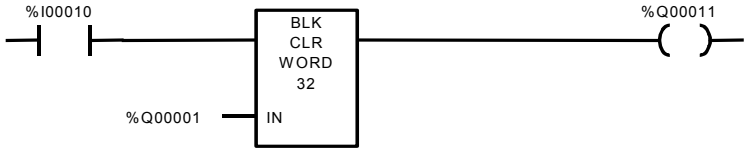
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN			o	o		o		
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

At power-up, 32 words of %Q memory (512 points) beginning at %Q00001 are filled with zeros.



4.6.4 SHFR_(type)

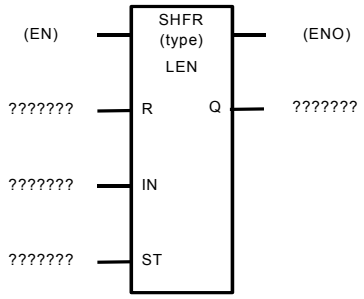
Function

When the Shift Register function (SHFR_BIT, SHFR_BYTE, SHFR_WORD or SHFR_DWORD) receives power and R does not, the function shifts one or more data bits or data words from a reference location into a specified area of memory. A contiguous section of memory serves as a shift register. For example, one word might be shifted into an area of memory with a specified length of five words. As a result of this shift, another word of data would be shifted out of the end of the memory area.
The reset input (R) takes precedence over the function enable input. When the reset is active, all references beginning at the shift register (ST) up to the length specified for the LEN operand, are filled with zeros.
If the function receives power flow and R is not active, each bit or word of the shift register is moved to the next highest reference. The last element in the shift register is shifted into Q. The highest reference of the shift register element of IN is shifted into the vacated element starting at ST.
The function passes power to the right whenever the function receives power flow and the R operand does not. ON.

SHFR_(type) instructions

Name	Explanation
CHFR_BIT	Shift Register (BIT data)
CHFR_BYTE	Shift Register (BYTE data)
CHFR_WORD	Shift Register (WORD data)
CHFR_DWORD	Shift Register (DWORD data)

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
LEN	Constant	Length; the number of data in the shift register. $1 \leq LEN \leq 256$
R	BOOL	Reset
IN	ANY_BIT	The value to be shifted into the first data of the shift register
ST	ANY_BIT	Shift Register
Q	ANY_BIT	The data shifted out of the shift register
ENO	BOOL	Returns the value of EN

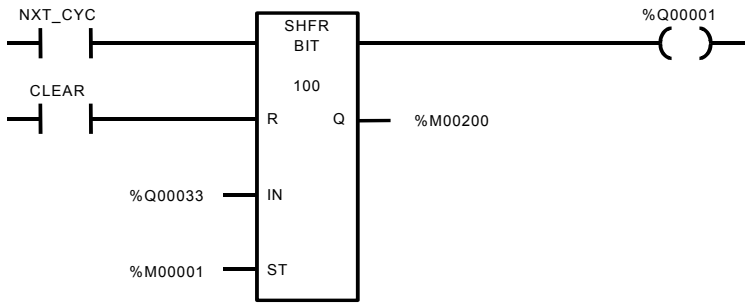
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
R	o	o	o	o	o			
IN		o	o	o	o		o	
ST		o	o	o	o	o		
Q			o	o	o	o		
ENO	o		o	o	Note	o		o

NOTE
Only %SK can be specified.

Example

SHFR_BIT operates on Bool memory locations %M00001 through %M00100. When the reset reference CLEAR is active, the Shift Register function fills %M00001 through %M00100 with zeros. When NXT_CYC is active and CLEAR is not, SHFR_BIT shifts the data in %M00001 to %M00100 down by one bit. The bit in %Q00033 is shifted into %M00001 while the bit shifted out of %M0100 is written to %M00200.



4.7 DATA TABLE FUNCTIONS

Data table functions

Instruction	Function	Parameter	Data type	Controlled I/O
ARRAY_MOVE_(type)	Movement of data array	SR, SNX, DNX, N, DS	SR/DS = ANY_BIT, ANY_INT SNX/DNX = INT N = INT	EN, ENO
SEARCH_EQ_(type)	Search for equal to (= IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD
SEARCH_NE_(type)	Search for not equal to (≠ IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD
SEARCH_GT_(type)	Search for greater than (> IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD
SEARCH_GE_(type)	Search for greater than or equal to (≥ IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD
SEARCH_LT_(type)	Search for less than (< IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD
SEARCH_LE_(type)	Search for less than or equal to (≤ IN)	AR, INX, IN, ONX	AR/IN = ANY_BIT, ANY_INT INX/ONX = WORD	EN, FD

4.7.1 ARRAY_MOVE_(type)

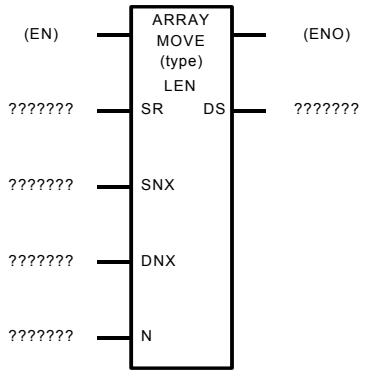
Function

When EN is set to ON and is receiving the power flow, ARRAY_MOVE moves, from the location specified with parameters SR and SNX (SR + SNX - 1) to the location specified with parameters DS and DNX (DS + DNX - 1), the number of array elements specified with input parameter N.

ARRAY_MOVE_(type) instructions

Name	Explanation
ARRAY_MOVE_BOOL	Movement of BOOL data array
ARRAY_MOVE_BYTE	Movement of BYTE data array
ARRAY_MOVE_WORD	Movement of WORD data array
ARRAY_MOVE_DWORD	Movement of DWORD data array
ARRAY_MOVE_SINT	Movement of SINT data array
ARRAY_MOVE_USINT	Movement of USINT data array
ARRAY_MOVE_INT	Movement of INT data array
ARRAY_MOVE_UINT	Movement of UINT data array
ARRAY_MOVE_DINT	Movement of DINT data array
ARRAY_MOVE_UDINT	Movement of UDINT data array

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of transmission
LEN	constant	The length of each memory block (source and destination); the number of registers in each memory block. 1 ≤ LEN ≤ 32767.
SR	ANY_BIT ANY_INT	Head address of array at origin
SNX	INT	Index of array at origin
DNX	INT	Index of array at destination
N	INT	Number of items of data
ENO	BOOL	Normal end flag
DS	ANY_BIT ANY_INT	Head address of destination array

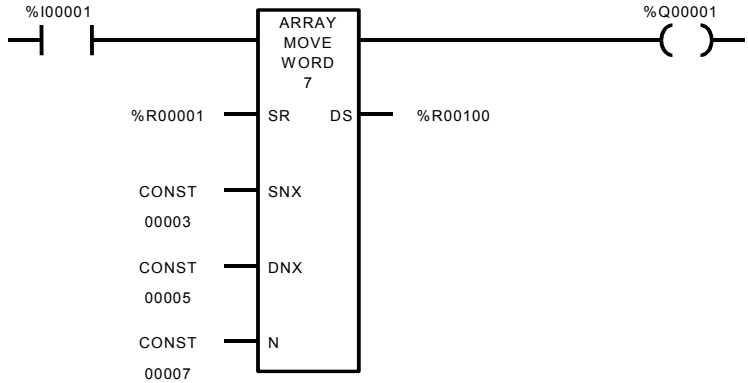
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
SR		o	o	o	o	o		
SNX		o	o	o	o	o	o	
DNX		o	o	o	o	o	o	
N		o	o	o	o	o	o	
ENO	o		o	o	Note			o
DS			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the array elements between %R00003 and %R00009 are moved to between %R00104 and %R00110.



4.7.2 SEARCH_EQ_(type)/SEARCH_NE_(type)/SEARCH_GT_(type)/ SEARCH_GE_(type)/SEARCH_LT_(type)/SEARCH_LE_(type)

Function

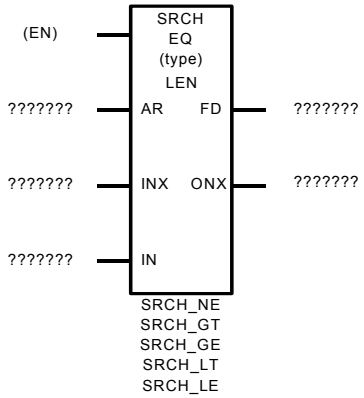
When EN is set to ON and is receiving the power flow, SRCH_EQ/SRCH_NE/SRCH_GT/SRCH_GE/SRCH_LT/SRCH_LE search for, from the start position specified with input parameters AR and INX (AR + INX), those array elements conforming to the data and conditions of input parameter IN (= IN/ ≠ IN/ > IN/ ≥ IN/ < IN/ ≤ IN). Those array elements that satisfy the conditions are output, output parameter FD is set to ON, and the indexes of those array elements are output to output parameter ONX. When there are no array elements that satisfy the conditions, FD is set to OFF, and 0 is output to ONX.

SEARCH_xx_(type) instructions

Name	Explanation
SEARCH_EQ_BOOL	Search for equal to BOOL data array
SEARCH_EQ_BYTE	Search for equal to BYTE data array
SEARCH_EQ_WORD	Search for equal to WORD data array
SEARCH_EQ_DWORD	Search for equal to DWORD data array
SEARCH_EQ_SINT	Search for equal to SINT data array
SEARCH_EQ_USINT	Search for equal to USINT data array
SEARCH_EQ_INT	Search for equal to INT data array
SEARCH_EQ_UINT	Search for equal to UINT data array
SEARCH_EQ_DINT	Search for equal to DINT data array
SEARCH_EQ_UDINT	Search for equal to UDINT data array
SEARCH_NE_BOOL	Search for not equal to BOOL data array
SEARCH_NE_BYTE	Search for not equal to BYTE data array
SEARCH_NE_WORD	Search for not equal to WORD data array
SEARCH_NE_DWORD	Search for not equal to DWORD data array
SEARCH_NE_SINT	Search for not equal to SINT data array
SEARCH_NE_USINT	Search for not equal to USINT data array
SEARCH_NE_INT	Search for not equal to INT data array
SEARCH_NE_UINT	Search for not equal to UINT data array
SEARCH_NE_DINT	Search for not equal to DINT data array
SEARCH_NE_UDINT	Search for not equal to UDINT data array
SEARCH_GT_BYTE	Search for greater than BYTE data array
SEARCH_GT_WORD	Search for greater than WORD data array
SEARCH_GT_DWORD	Search for greater than DWORD data array
SEARCH_GT_SINT	Search for greater than SINT data array
SEARCH_GT_USINT	Search for greater than USINT data array
SEARCH_GT_INT	Search for greater than INT data array
SEARCH_GT_UINT	Search for greater than UINT data array
SEARCH_GT_DINT	Search for greater than DINT data array
SEARCH_GT_UDINT	Search for greater than UDINT data array
SEARCH_GE_BYTE	Search for greater than or equal to BYTE data array
SEARCH_GE_WORD	Search for greater than or equal to WORD data array
SEARCH_GE_DWORD	Search for greater than or equal to DWORD data array
SEARCH_GE_SINT	Search for greater than or equal to SINT data array
SEARCH_GE_USINT	Search for greater than or equal to USINT data array
SEARCH_GE_INT	Search for greater than or equal to INT data array

Name	Explanation
SEARCH_GE_UINT	Search for greater than or equal to UINT data array
SEARCH_GE_DINT	Search for greater than or equal to DINT data array
SEARCH_GE_UDINT	Search for greater than or equal to UDINT data array
SEARCH_LT_BYTE	Search for less than BYTE data array
SEARCH_LT_WORD	Search for less than WORD data array
SEARCH_LT_DWORD	Search for less than DWORD data array
SEARCH_LT_SINT	Search for less than SINT data array
SEARCH_LT_USINT	Search for less than USINT data array
SEARCH_LT_INT	Search for less than INT data array
SEARCH_LT_UINT	Search for less than UINT data array
SEARCH_LT_DINT	Search for less than DINT data array
SEARCH_LT_UDINT	Search for less than UDINT data array
SEARCH_LE_BYTE	Search for less than or equal to BYTE data array
SEARCH_LE_WORD	Search for less than or equal to WORD data array
SEARCH_LE_DWORD	Search for less than or equal to DWORD data array
SEARCH_LE_SINT	Search for less than or equal to SINT data array
SEARCH_LE_USINT	Search for less than or equal to USINT data array
SEARCH_LE_INT	Search for less than or equal to INT data array
SEARCH_LE_UINT	Search for less than or equal to UINT data array
SEARCH_LE_DINT	Search for less than or equal to DINT data array
SEARCH_LE_UDINT	Search for less than or equal to UDINT data array

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of search
LEN	Constant	The number of registers starting at AR that makes up the memory block to search. 1 ≤ LEN ≤ 32767.
AR	ANY_BIT ANY_INT	Head address of array
INX	WORD	Search start index
IN	ANY_BIT ANY_INT	Search value
FD	BOOL	Whether any value satisfies the condition
ONX	WORD	Index of value that satisfies the condition

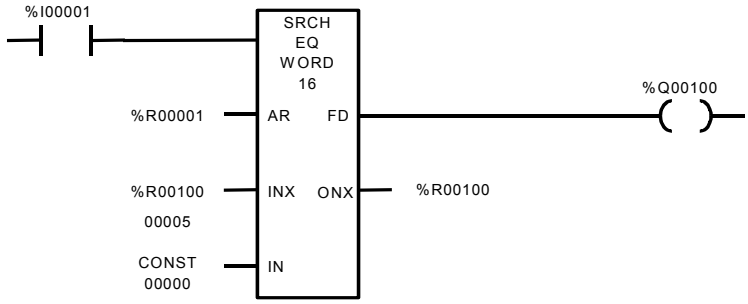
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
AR		o	o	o	o	o		
INX		o	o	o	o	o	o	
IN		o	o	o	o	o	o	
FD	o		o	o	Note	o		o
ONX			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, and the elements of the array between %R00001 and %R00016 are 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0, and 500, setting the initial value of %R00100 to 5 starts a search, starting from %R00006, for those array elements whose value is 0.
In the following example, the first scan reveals %R00007, %Q00100 is set to ON, and %R00100 is set to 7. In the second scan, the search starts from %R00008, %R00015 is revealed, %Q00100 is set to ON, and %R00100 is set to 15. In the next scan, the search starts from %R00016 but, because the value of %R00016 is other than 0, %Q00100 is left set to OFF, and %R00100 is set to 0. In the next scan, the search starts from the head of the array, namely %R00001.



4.8 DATA TYPE CONVERSION

Data type conversion functions

Instruction	Function	Parameter	Data type	Controlled I/O
(type)_TO_BCDx	Conversion from binary to BCD data	IN, Q	IN = ANY_INT Q = WORD, DWORD	EN, ENO
BCDx_TO_(type)	Conversion from BCD to binary data	IN, Q	IN = WORD, DWORD Q = ANY_INT	EN, ENO
(type)_TO_SINT	Conversion to SINT data	IN, Q	IN = USINT, INT, UINT, DINT, UDINT Q = SINT	EN, ENO
(type)_TO_USINT	Conversion to USINT data	IN, Q	IN = SINT, INT, UINT, DINT, UDINT Q = USINT	EN, ENO
(type)_TO_INT	Conversion to INT data	IN, Q	IN = SINT, USINT, UINT, DINT, UDINT Q = INT	EN, ENO
(type)_TO_UINT	Conversion to UINT data	IN, Q	IN = SINT, USINT, INT, DINT , UDINT Q = UINT	EN, ENO
(type)_TO_DINT	Conversion to DINT data	IN, Q	IN = SINT, USINT, INT, UINT, UDINT Q = DINT	EN, ENO
(type)_TO_UDINT	Conversion to UDINT data	IN, Q	IN = SINT, USINT, INT, UINT, DINT Q = DINT	EN, ENO

4.8.1 (type)_TO_BCDx (x = 2, 4, 8)

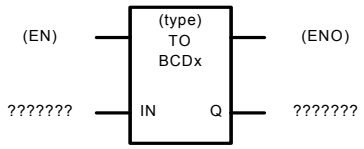
Function

When EN is set to ON and is receiving the power flow, (type)_TO_BCDx converts the integer data of input parameter IN to 2-, 4- or 8- digit BCD, then outputs the result to output parameter Q.
When a large value that can not be expressed in BCD (a value greater than 99 for a 2-digit BCD, greater than 9999 for a 4-digit BCD, and greater than 99999999 for an 8-digit BCD) is input to IN, the maximum value that can be expressed in BCD (99 for a 2-digit BCD, 9999 for a 4-digit BCD, and 99999999 for an 8-digit BCD) is output to Q. In such a case, ENO is not output. Also, if a negative value is input to IN, 0 is output to Q. Again, in such a case ENO is not output.

(type)_TO_BCDx instructions

Name	Explanation
SINT_TO_BCD2	Converts from SINT to 2-digit BCD
USINT_TO_BCD2	Converts from USINT to 2-digit BCD
INT_TO_BCD4	Converts from INT to 4-digit BCD
UINT_TO_BCD4	Converts from UINT to 4-digit BCD
DINT_TO_BCD8	Converts from DINT to 8-digit BCD
UDINT_TO_BCD8	Converts from UDINT to 8-digit BCD

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	ANY_INT	Input data
ENO	BOOL	Normal end flag
Q	WORD, DWORD	Result of conversion

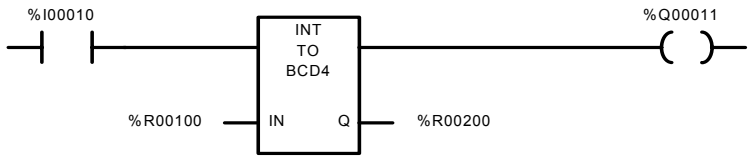
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the INT type data of %R00100 is converted to 4-digit BCD data and output to %R00200.



4.8.2 BCDx_TO_(type) (x = 2, 4, 8)

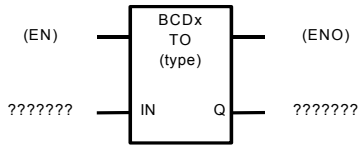
Function

When EN is set to ON and is receiving the power flow, BCDx_TO_(type) converts the 2-, 4-, or 8-digit data input to input parameter IN to integer data and outputs the result to output parameter Q.
If an invalid BCD value is input to IN, the minimum valid value for that data type (-128 for SINT, 0 for USINT) is output to Q. In this case, ENO is not output.

BCDx_TO_(type) instructions

Name	Explanation
BCD2_TO_SINT	Converts from 2-digit BCD to SINT
BCD2_TO_USINT	Converts from 2-digit BCD to USINT
BCD4_TO_INT	Converts from 4-digit BCD to INT
BCD4_TO_UINT	Converts from 4-digit BCD to UINT
BCD8_TO_DINT	Converts from 8-digit BCD to DINT
BCD8_TO_UDINT	Converts from 8-digit BCD to UDINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	WORD, DWORD	Input data
ENO	BOOL	Normal end flag
Q	ANY_INT	Result of conversion

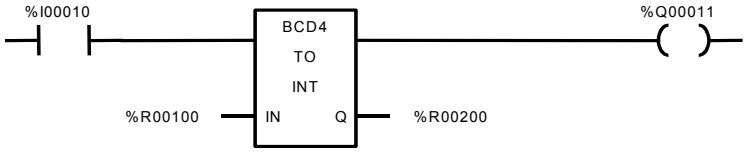
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the 4-digit BCD data of %R00100 is converted to INT type data and output to %R00200.



4.8.3 (type)_TO_SINT

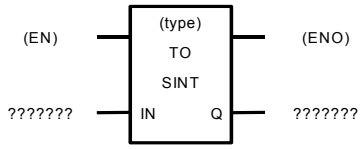
Function

When EN is set to ON and is receiving the power flow, (type)_TO_SINT converts the data of input parameter IN to SINT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_SINT instructions

Name	Explanation
USINT_TO_SINT	Converts from USINT to SINT
INT_TO_SINT	Converts from INT to SINT
UINT_TO_SINT	Converts from UINT to SINT
DINT_TO_SINT	Converts from DINT to SINT
UDINT_TO_SINT	Converts from UDINT to SINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	USINT, INT, UINT, DINT, UDINT	Input data
ENO	BOOL	Normal end flag
Q	SINT	Result of conversion

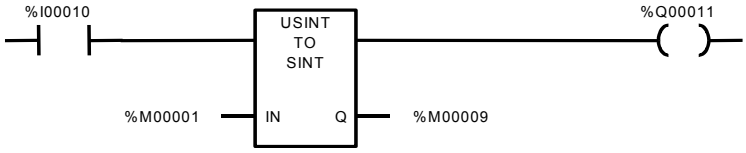
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the USINT data of %M00001 is converted to SINT data and the result is output to %M00009.



4.8.4 (type)_TO_USINT

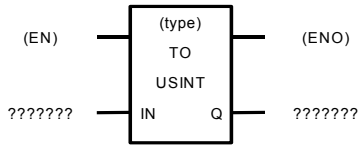
Function

When EN is set to ON and is receiving the power flow, (type)_TO_USINT converts the data of input parameter IN to USINT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_USINT instructions

Name	Explanation
SINT_TO_USINT	Converts from SINT to USINT
INT_TO_USINT	Converts from INT to USINT
UINT_TO_USINT	Converts from UINT to USINT
DINT_TO_USINT	Converts from DINT to USINT
UDINT_TO_USINT	Converts from UDINT to USINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	SINT, INT, UINT, DINT, UDINT	Input data
ENO	BOOL	Normal end flag
Q	USINT	Result of conversion

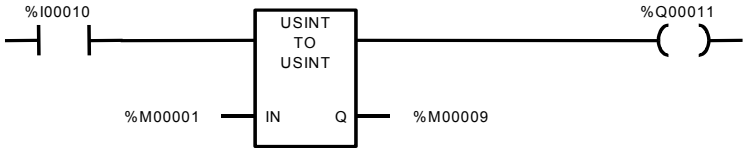
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the SINT data of %M00001 is converted to USINT data and the result is output to %M00009.



4.8.5 (type)_TO_INT

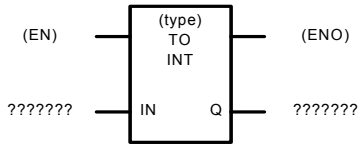
Function

When EN is set to ON and is receiving the power flow, (type)_TO_INT converts the data of input parameter IN to INT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_INT instructions

Name	Explanation
SINT_TO_INT	Converts from SINT to INT
USINT_TO_INT	Converts from USINT to INT
UINT_TO_INT	Converts from UINT to INT
DINT_TO_INT	Converts from DINT to INT
UDINT_TO_INT	Converts from UDINT to INT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	SINT, USINT, UINT, DINT, UDINT	Input data
ENO	BOOL	Normal end flag
Q	INT	Result of conversion

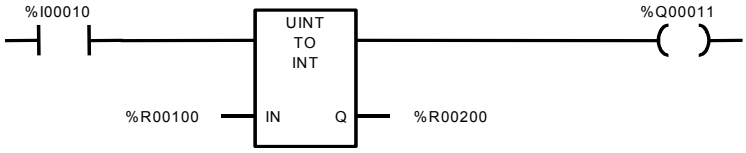
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the UINT data of %R00100 is converted to INT data and the result is output to %R00200.



4.8.6 (type)_TO_UINT

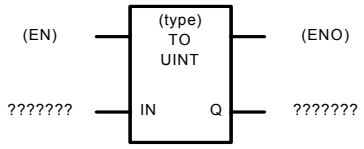
Function

When EN is set to ON and is receiving the power flow, (type)_TO_UINT converts the data of input parameter IN to UINT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_UINT instructions

Name	Explanation
SINT_TO_UINT	Converts from SINT to UINT
USINT_TO_UINT	Converts from USINT to UINT
INT_TO_UINT	Converts from INT to UINT
DINT_TO_UINT	Converts from DINT to UINT
UDINT_TO_UINT	Converts from UDINT to UINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	SINT, USINT, INT, DINT, UDINT	Input data
ENO	BOOL	Normal end flag
Q	UINT	Result of conversion

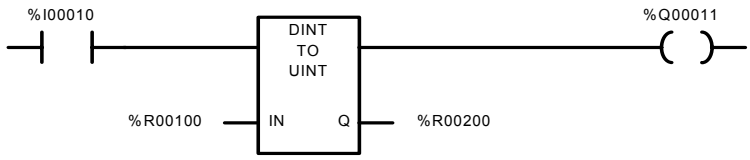
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the DINT data of %R00100 is converted to UINT data and the result is output to %R00200.



4.8.7 (type)_TO_DINT

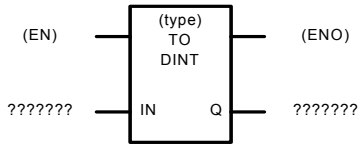
Function

When EN is set to ON and is receiving the power flow, (type)_TO_DINT converts the data of input parameter IN to DINT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_DINT instructions

Name	Explanation
SINT_TO_DINT	Converts from SINT to DINT
USINT_TO_DINT	Converts from USINT to DINT
INT_TO_DINT	Converts from INT to DINT
UINT_TO_DINT	Converts from UINT to DINT
UDINT_TO_DINT	Converts from UDINT to DINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	SINT, USINT, INT, UINT, UDINT	Input data
ENO	BOOL	Normal end flag
Q	DINT	Result of conversion

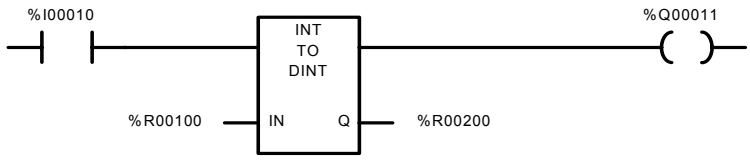
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the INT data of %R00100 is converted to DINT data and the result is output to %R00200.



4.8.8 (type)_TO_UDINT

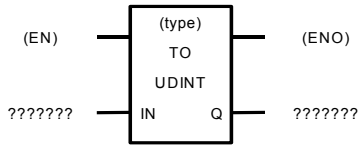
Function

When EN is set to ON and is receiving the power flow, (type)_TO_UDINT converts the data of input parameter IN to UDINT data and outputs the result to output parameter Q.
If the conversion results in an overflow, the maximum value for that data type is output to Q. In such a case, ENO is not output. Furthermore, if the conversion results in an underflow, the minimum value for that type is output to Q. Again, in such a case, ENO is not output.

(type)_TO_UDINT instructions

Name	Explanation
SINT_TO_UDINT	Converts from SINT to UDINT
USINT_TO_UDINT	Converts from USINT to UDINT
INT_TO_UDINT	Converts from INT to UDINT
UINT_TO_UDINT	Converts from UINT to UDINT
DINT_TO_UDINT	Converts from DINT to UDINT

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of conversion
IN	SINT, USINT, INT, UINT, DINT	Input data
ENO	BOOL	Normal end flag
Q	UDINT	Result of conversion

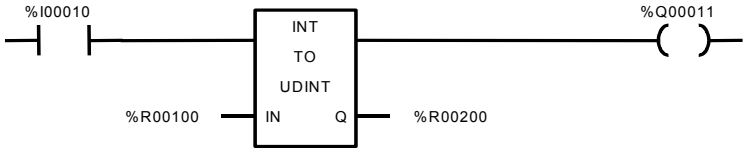
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the INT data of %R00100 is converted to UDINT data and the result is output to %R00200.



4.9 PROGRAM FLOW

Program Flow

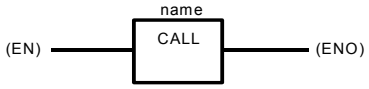
Instruction	Function
CALL	Call IL or LD block
END	Ends execution unconditionally
MCR	Master control(Non-nested)
ENDMCR	Ends master control(Non-nested)
MCRN	Master control(Nested)
ENDMCRN	Ends master control(Nested)
JUMPN	Jump
LABELN	Label of jump destination
COMMENT	Comment

4.9.1 CALL

Function

When the CALL function receives power flow, it causes the scan to go immediately to the designated subroutine IL block or LD block and execute it. After the subroutine block execution is complete, control returns to the point in the logic immediately following the CALL instruction.

Format



Parameters

Parameter	Data type	Meaning
name	-	Name of IL block or LD block
EN	BOOL	Execution of CALL
ENO	BOOL	Normal end flag

Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			o
ENO	o		o	o	Note			o

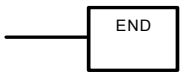
NOTE
Only %SK can be specified.

4.9.2 END

Function

Unconditionally ends the execution of the sequence program for each level. For each level, the program logic subsequent to the END instruction is not executed. After the execution of an END instruction, execution switches to the head address of the program for each label.

Format



Parameters

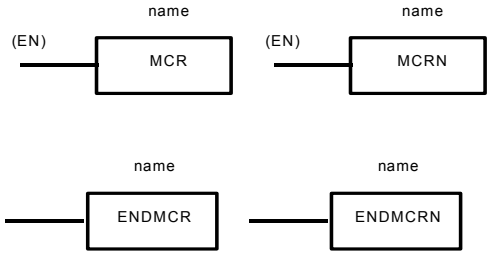
None

4.9.3 MCR/ENDMCR/MCRN/ENDMCRN

Function

When EN is set to ON and is receiving the power flow, MCR(N) executes all the rungs up to the corresponding ENDMCR(N) without applying power to the coil. ENDMCR(N) must be described after the corresponding MCR(N). Furthermore, within the same rung, nothing can be described either before or after ENDMCR(N).
MCR is Non-nested form.
MCRN is Nested form.

Format



Parameters

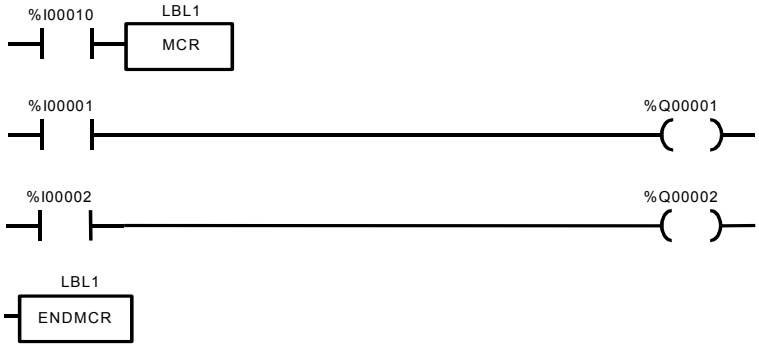
Parameter	Data type	Meaning
EN	BOOL	Execution of master control
name	-	Name

Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			

Example

When %I00010 is set to ON, regardless of the ON/OFF state of %I00001 and %I00002, a negative power flow is established (that is, no power flows), and %Q00001 and %Q00002 are both set to OFF.

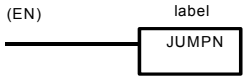


4.9.4 JUMPN

Function

When EN is set to ON and is receiving the power flow, JUMPN causes processing to jump to the corresponding LABELN. All intervening processing is skipped.
All of the coils between the JUMPN and the corresponding LABELN maintain their original state.
LABELN is not necessarily be described after the corresponding JUMPN. Furthermore, within the same rung, nothing can be described either before or after LABELN.

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of jump
label	-	Label

Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			

4.9.5 LABELN

Function

Indicates the destination of a JUMPN instruction.

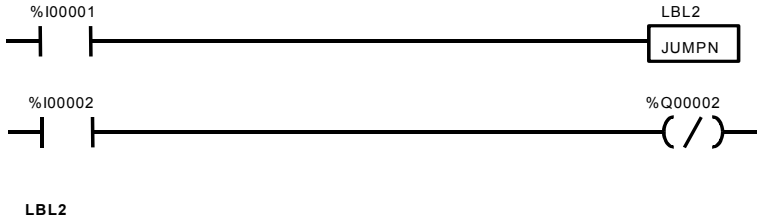
Format

Parameters

Parameter	Data type	Meaning
label	-	Label

Example

When %I00001 is set to ON, the processing up to LBL2 is skipped. (The power flow goes directly from %I00001 to LBL2.)



4.9.6 COMMENT

Function

Enables a comment to be inserted into a program.
Each comment occupies one rung.

Format

Parameters

Parameter	Data type	Meaning
comment	-	comment

4.10 PMC OPERATIONS

PMC operation instructions

Instruction	Function	Parameter	Data type	Controlled I/O
PMC_ADD_BCDx	Addition of BCD data	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
PMC_SUB_BCDx	Subtraction of BCD data	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
PMC_MUL_BCDx	Multiplication of BCD data	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
PMC_DIV_BCDx	Division of BCD data	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
PMC_MOD_BCDx	Remainder of BCD data	IN1, IN2, Q	BYTE, WORD, DWORD	EN, ENO
R_TRIG	Detection of rising edge	--	--	EN, ENO, address
F_TRIG	Detection of falling edge	--	--	EN, ENO, address
PMC_DECODE_(type)	Decode	LEN, IN, BASE, Q	IN/BASE = ANY_INT LEN/Q = WORD	EN, ENO
PMC_EVPAR_(type)	Even parity check	IN	BYTE, WORD, DWORD	EN, ENO, Q
PMC_ODPAR_(type)	Odd parity check	IN	BYTE, WORD, DWORD	EN, ENO, Q
PMC_WINDOW	Read and write of CNC window data	LEN, AR	INT	EN, Q, ERR
PMC_EXIN	External data input	HEAD, CMD, DT	HEAD = INT CMD = WORD DT = DWORD	EN, Q, ERR
PMC_AXCTL	PMC axis control	GRP, CMD, DT1, DT2	GRP = INT CMD = WORD DT1 = DINT DT2 = DINT	EN, R, Q, ERR

4.10.1 PMC_ADD_BCDx/PMC_SUB_BCDx/PMC_MUL_BCDx/ PMC_DIV_BCDx (x = 2, 4, 8)

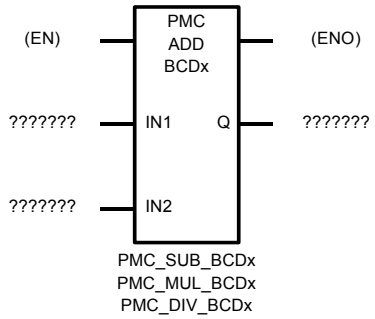
Function

When EN is set to ON and is receiving the power flow, PMC_ADD_BCDx/PMC_SUB_BCDx/PMC_MUL_BCDx/PMC_DIV_BCDx perform addition, subtraction, multiplication, and division on the BCD data of input parameters IN1 and IN2, and output the result to output parameter Q.
IN1, IN2, and Q must all be of the same data type.
If the operation produces an overflow, the maximum possible value for that data type is set in the output reference. If invalid BCD data is input to IN, 0 is output to Q. When the operation ends without an overflow, ENO is set to ON.

Addition/subtraction/multiplication/division of BCD data

Name	Explanation
PMC_ADD_BCD2	Addition of 2-digit BCD data
PMC_ADD_BCD4	Addition of 4-digit BCD data
PMC_ADD_BCD8	Addition of 8-digit BCD data
PMC_SUB_BCD2	Subtraction of 2-digit BCD data
PMC_SUB_BCD4	Subtraction of 4-digit BCD data
PMC_SUB_BCD8	Subtraction of 8-digit BCD data
PMC_MUL_BCD2	Multiplication of 2-digit BCD data
PMC_MUL_BCD4	Multiplication of 4-digit BCD data
PMC_MUL_BCD8	Multiplication of 8-digit BCD data
PMC_DIV_BCD2	Division of 2-digit BCD data
PMC_DIV_BCD4	Division of 4-digit BCD data
PMC_DIV_BCD8	Division of 8-digit BCD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
IN1	BYTE, WORD, DWORD	Input data 1
IN2	BYTE, WORD, DWORD	Input data 2
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	BYTE, WORD, DWORD	Result of operation

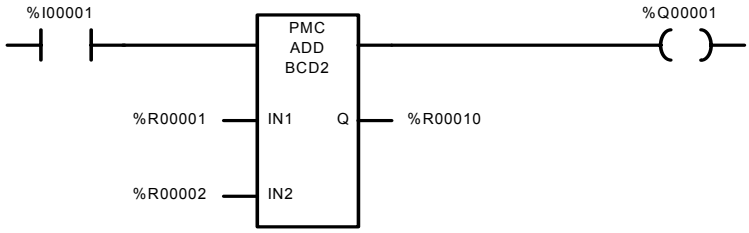
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the 2-digit BCD data in %R00001 is added to that in %R00002, and the result is output to %R00010.
When %Q00001 is set to ON, it indicates that the operation ended with no overflow.



4.10.2 PMC_MOD_BCDx (x = 2, 4, 8)

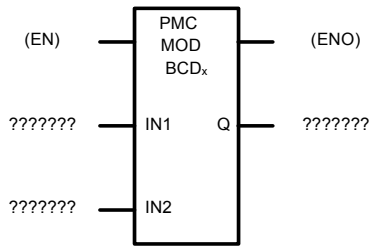
Function

When EN is set to ON and is receiving the power flow, PMC_MOD_BCDx divides the BCD data of input parameter IN1 by that of IN2 (IN1 ÷ IN2) and outputs the remainder to output parameter Q.
IN1, IN2, and Q must all be of the same data type.
If invalid BCD data is input to IN, 0 is output to Q. When the operation ends with no overflow (such as that caused by division by zero), ENO is set to ON.

Remainder of BCD data

Name	Explanation
PMC_MOD_BCD2	Remainder of 2-digit BCD data
PMC_MOD_BCD4	Remainder of 4-digit BCD data
PMC_MOD_BCD8	Remainder of 8-digit BCD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
IN1	BYTE, WORD, DWORD	Input data 1
IN2	BYTE, WORD, DWORD	Input data 2
ENO	BOOL	Normal end flag (set to OFF upon an overflow)
Q	BYTE, WORD, DWORD	Result of operation

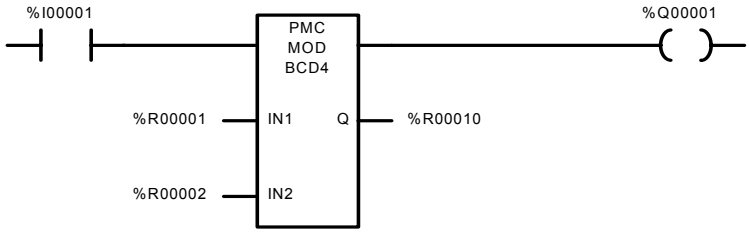
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN1		o	o	o	o	o	o	
IN2		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the 4-digit BCD data in %R00001, IN1, is divided by that in %R00002 (IN2) (IN1 ÷ IN2), and the remainder is output to %R00010.
When %Q00001 is set to ON, it indicates that the operation ended without an overflow.

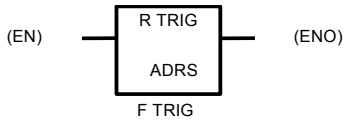


4.10.3 R_TRIG/F_TRIG

Function

When EN is rising (going from OFF to ON) or falling (going from ON to OFF), R_TRIG/F_TRIG sets ENO to ON for one sweep only.

Format



Parameters

Parameter	Data type	Meaning
ADRS	BOOL	Detection memory
EN	BOOL	Detection signal input
ENO	BOOL	Detection result output

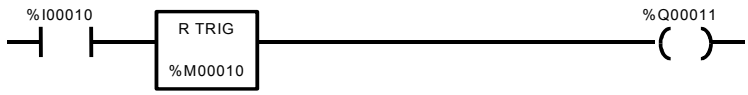
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
ADRS				o				
EN	o	o	o	o	o			
ENO	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00010 goes from OFF to ON, %Q00011 is set to ON for one sweep only.



4.10.4 PMC_DECODE_(type)

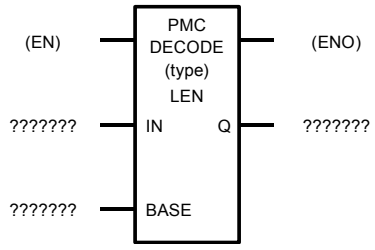
Function

When EN is set to ON and is receiving the power flow, PMC_DECODE extracts continuous 8 x n (depending on the array size of output parameter Q) integer data from the integer data of input parameter IN and that of BASE, decodes it, then sets the corresponding bit of output parameter Q.

PMC_DECODE_(type) instructions

Name	Explanation
PMC_DECODE_SINT	Decode of SINT data
PMC_DECODE_USINT	Decode of USINT data
PMC_DECODE_INT	Decode of INT data
PMC_DECODE_UINT	Decode of UINT data
PMC_DECODE_DINT	Decode of DINT data
PMC_DECODE_UDINT	Decode of UDINT data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of operation
LEN	Constant	Length of Q array(WORD)
IN	ANY_INT	Input data
BASE	ANY_INT	Decode indication number
ENO	BOOL	Returns the value of EN
Q	WORD	Result of decoding

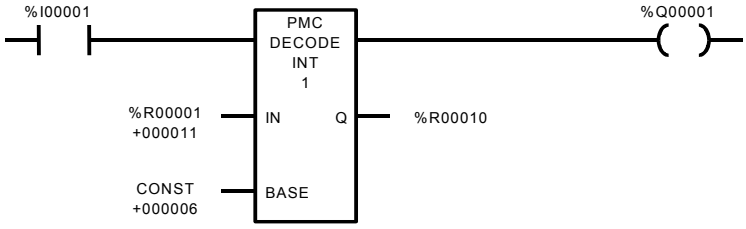
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
BASE		o	o	o	o	o	o	
ENO	o		o	o	Note			o
Q			o	o	Note	o		

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the value of %R00001 is decoded into sixteen (when the array size of output address %00010 is 1) continuous integer values, starting from 6.
The result of decoding, 00000000 00100000, is output to %R00010.



4.10.5 PMC_EVPAR_(type)/PMC_ODPAR_(type)

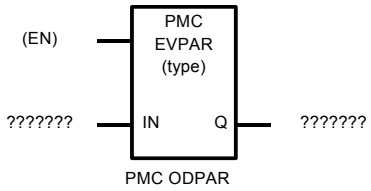
Function

When EN is set to ON and is receiving the power flow, PMC_EVPAR/PMC_ODPAR check the even/odd parity of input parameter IN and, when the result is true, set output parameter Q to ON.

PMC_EVPAR_(type)/PMC_ODPAR_(type) instructions

Name	Explanation
PMC_EVPAR_BYTE	Check the even parity of BYTE data
PMC_EVPAR_WORD	Check the even parity of WORD data
PMC_EVPAR_DWORD	Check the even parity of DWORD data
PMC_ODPAR_BYTE	Check the odd parity of BYTE data
PMC_ODPAR_WORD	Check the odd parity of WORD data
PMC_ODPAR_DWORD	Check the odd parity of DWORD data

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of parity check
IN	BYTE, WORD, DWORD	Input data
Q	BOOL	Result of check

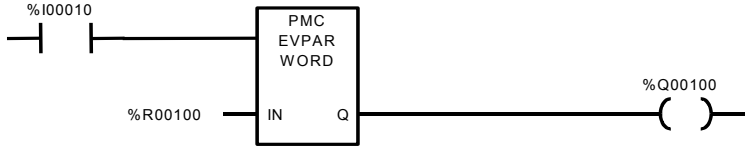
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
IN		o	o	o	o	o	o	
Q	o		o	o	Note			o

NOTE
Only %SK can be specified.

Example

When %I00010 is set to ON, the data of %R00100 is subjected to a parity check. When the parity is even, %Q00100 is set to ON.

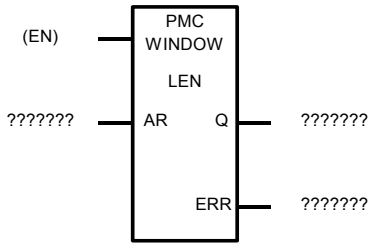


4.10.6 PMC_WINDOW

Function

The CNC ←→ PMC window function is used to read and write the CNC data.

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	WINDOW execution
LEN	Constant	Length of AR(INT) array
AR	INT	AR array Top address +0 Function code ...INT +2 Completion code ...INT +4 Data length ...INT +6 Data number ...INT +8 Data attribute ...INT +10 Data (max. 32 bytes) ...ANY_NUM (Values with each function)
Q	BOOL	Completion flag
ERR	BOOL	Error output

For details of the data contents set in the AR array, refer to the specified sections of the Appendix A ”WINDOW FUNCTIONS.”

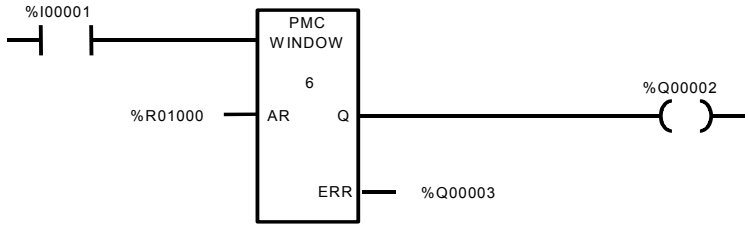
Memory Type

	Flow	Memory %I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
AR			o	o		o		
Q	o		o	o	Note			o
ERR			o	o	Note			

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, NC parameter No. 1815 (N = +001815) is read (FNC = +000017) for all axes (ATR = -000001).
The completion flag is output to %Q00002, while the error flag is output to %Q00003.



%R01000	17
%R01001	-
%R01002	-
%R01003	1815
%R01004	-1
%R01005	-

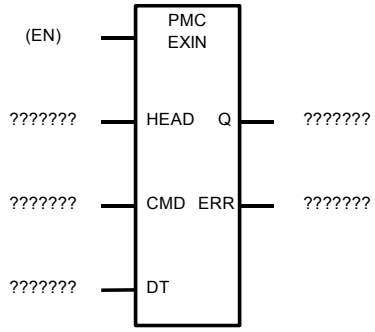
NOTE
%R address is 16-bit size (word type).

4.10.7 PMC_EXIN

Function

Issues commands for the CNC external data input function

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of EXIN
HEAD	INT	HEAD NO.
CMD	WORD	Command code
DT	DWORD	Command data
Q	BOOL	Completion flag
ERR	BOOL	Error output

For details of the data contents set in the parameters, refer to the specified sections of the following manuals.

- FANUC PMC-MODEL PA1/PA3/SA1/SA2/SA3/SA5/SB/SB2/ SB3/SB4/SB5/SB6/SB7/SC/SC3/SC4/NB/NB2/NB6 Ladder Language Programming Manual (B-61863E) 5.45 EXIN (External Data Input)
- FANUC Series 16*i*/18*i*/21*i*/160*i*/180*i*/210*i*-MODEL B Connection Manual (Functions) (B-63523EN-1) 15.2 External Data Input

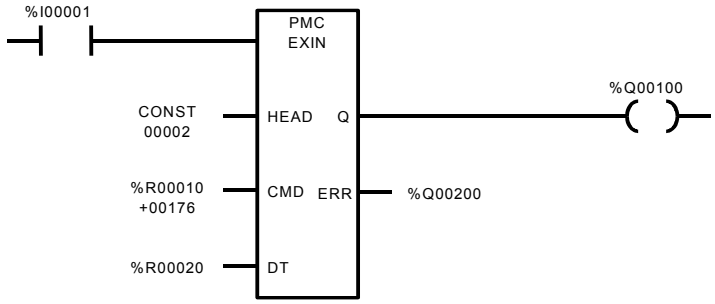
Memory Type

		Memory						
	Flow	%I*	%Q*	%M*	%S*	%R	Constant	Omissible
EN	o	o	o	o	o			
HEAD		o	o	o	o	o	o	
CMD		o	o	o	o	o		
DT		o	o	o	o	o		
Q	o		o	o	Note			o
ERR			o	o	Note			

NOTE
Only %SK can be specified.

Example

For a 16*i*-TB two-path control system, in the example shown below, when %I00001 is set to ON, the shift amount for %R00020 is input for the first axis of the second path, and the machine coordinates are shifted.
The completion flag is output to %Q00100 and the error flag is output to %Q00200.

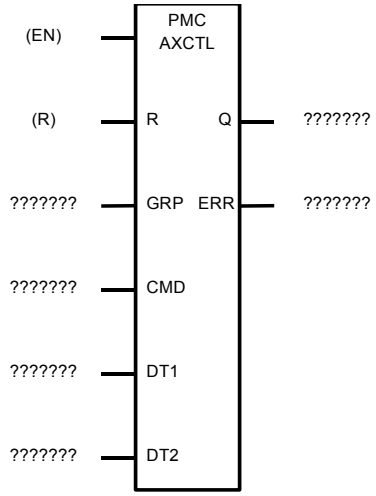


4.10.8 PMC_AXCTL

Function

Specifies the PMC axis control function of the CNC.

Format



Parameters

Parameter	Data type	Meaning
EN	BOOL	Execution of AXCTL
R	BOOL	Reset
GRP	INT	DI/DO signal group
CMD	WORD	Axis control command
DT1	DINT	Instruction data 1
DT2	DINT	Instruction data 2
Q	BOOL	Completion flag
ERR	BOOL	Error output

For details of the data contents set in the parameters, refer to the specified section of the following manual.

- FANUC PMC-MODEL PA1/PA3/SA1/SA2/SA3/SA5/SB/SB2/ SB3/SB4/SB5/SB6/SB7/SC/SC3/SC4/NB/NB2/NB6
  Ladder Language Programming Manual (B-61863E)
  5.72 AXCTL (PMC Axis Control Instruction)

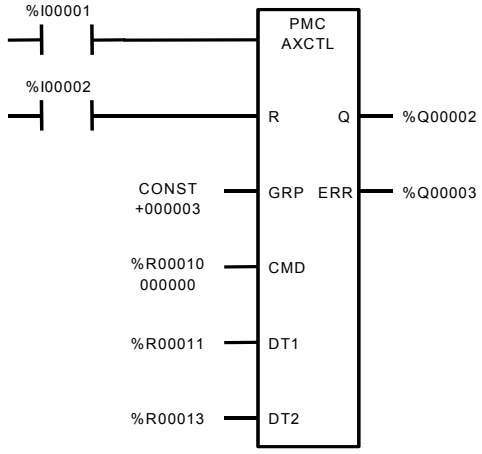
Memory Type

	Flow	Memory					Constant	Omissible
		%I*	%Q*	%M*	%S*	%R		
EN	o	o	o	o	o			
R	o	o	o	o	o			
GRP		o	o	o	o	o	o	
CMD		o	o	o	o	o	o	
DT1		o	o	o	o	o	o	
DT2		o	o	o	o	o	o	
Q			o	o	Note			
ERR			o	o	Note			

NOTE
Only %SK can be specified.

Example

When %I00001 is set to ON, the axis specified in the C group (GRP = +000003) is subjected to high-speed positioning (CMD = 000000) at the speed specified with %R00011, over the distance specified with %R00013.
The completion flag is output to %Q00002 and the error code is output to %Q00003.
In addition, the system can be reset by setting %I00002 to ON.



4.11 NOTE ON LD PROGRAMMING

Some functional instructions may cause the ladder program to take a long time to stop or make it unable to stop, if their EN or R condition remains on for no apparent reason. If the ladder program does not stop, all operations aimed at changing the ladder program will take longer to end or will never end.
To avoid such problems, when you code a ladder program using functional instructions, you need to design the ladder structure based on a thorough understanding of the control conditions of the individual instructions you use.
Listed below are typical cases in which the ladder program will not stop.

- A low‐speed window function is used for a PMC_WINDOW functional instruction, and its EN condition remains on.
- In a PMC_EXIN or PMC_AXCTL instruction, not only the EN condition but also the R condition remain on.

If the ladder program takes long to stop or does not stop for any of these reasons, the following operations will be affected.

1. Stopping the ladder program using a soft key on the screen.
2. Reading a new ladder program from a memory card or other medium, by using the data input and output screen.
3. Updating the ladder program with changes made using the FANUC LADDER-IIIC.

If any of the above phenomena occurs, the functional instruction causing the problem needs to be fixed. Check the functional instructions mentioned above to see whether there is any EN or R condition remaining on, and correct the ladder program according to the following procedure.

1. Put the machine in safe condition and turn off the power of the NC.
2. Turn on the power of the NC while holding down the "CAN" and "Z" keys simultaneously, to restart the NC with the ladder program halted.
3. In the FANUC LADDER-IIIC, redesign the logic associated with the problematic functional instruction. When done, set the EN or R condition to off.
4. Write the resulting logic to flash ROM using the I/O screen.
5. Run the ladder program.

If the ladder program does not stop or cannot be changed even after you make the correction, there may be other functional instructions that have the same condition settings.
Check for other functional instructions having the same condition settings, besides the one you have corrected, and repeat the above procedure to correct them all.

5 IL INSTRUCTIONS

IL logic instructions and functions are grouped according to the type of operation performed. The instruction groups are:

Number	Class
1	Basic IL Instructions
2	Timers & Counters
3	Math Functions
4	Relational Functions
5	Bit Functions
6	Data Move Functions
7	Data Table Functions
8	Conversion Functions
9	PMC Operations

5.1 BASIC INSTRUCTIONS

The basic IL instruction set supports operations on BOOL, 8-bit (BYTE), 16-bit (INT, WORD, UINT) and 32-bit (DINT, DWORD) variables. These instructions generally operate on an accumulator's content to generate a new value for the same accumulator. The following instructions (sorted by functional group) are supported:

(a) Accumulator Instructions (Basic IL)

Mnemonic	Description
LD_BOOL	Load a value into the boolean accumulator.
LDN_BOOL	Load the inverse of a value into the boolean accumulator.
LD_INT	Load a single-precision integer value into the integer accumulator.
LD_ENO	Load the enable output state, from the previously executed function, into the boolean accumulator.
ST_BOOL	Store the content of the boolean accumulator to a BOOL variable.
STN_BOOL	Store the inverse of the boolean accumulator to a BOOL variable.
ST_SINT	Store the content of the integer accumulator to an 8-bit integer (SINT) variable.
ST_USINT	Store the content of the integer accumulator to an unsigned 8-bit integer (USINT) variable.
ST_INT	Store the content of the integer accumulator to a 16-bit integer (INT) variable.
ST_UINT	Store the content of the integer accumulator to an unsigned 16-bit integer (UINT) variable.
ST_DINT	Store the content of the integer accumulator to a 32-bit integer (DINT) variable.
ST_UDINT	Store the content of the integer accumulator to an unsigned 32-bit integer (UDINT) variable.
ST_BYTE	Store the content of the integer accumulator to a BYTE variable.
ST_WORD	Store the content of the integer accumulator to a WORD variable.
ST_DWORD	Store the content of the integer accumulator to a DWORD variable.

(b) Coil Instructions (Basic IL)

Mnemonic	Description
S	Store the content of the boolean accumulator to the Set Coil variable.
R	Store the content of the boolean accumulator to the Reset Coil variable.

(c) Boolean Instructions (Basic IL)

Mnemonic	Description
NOT	Invert the state of the boolean accumulator.
AND	Perform a logical AND between an operand and the boolean accumulator.
ANDN	Perform a logical AND between the boolean accumulator and the inverse of an operand.
AND()	Perform a logical AND between a boolean expression and the boolean accumulator.
OR	Perform a logical OR between an operand and the boolean accumulator.
ORN	Perform a logical OR between the boolean accumulator and the inverse of an operand.
OR()	Perform a logical OR between a boolean expression and the boolean accumulator.
XOR	Perform a logical XOR between an operand and the boolean accumulator.
XOR()	Perform a logical XOR between a boolean expression and the boolean accumulator.
XORN	Perform a logical XOR between the boolean accumulator and the inverse of an operand.
XORN()	Perform a logical XORN between a boolean expression and the boolean accumulator.

(d) Math Instructions (Basic IL)

Mnemonic	Description
ADD	Add the content of the integer accumulator to the single-precision integer operand.
SUB	Subtract the single-precision integer operand from the content of the integer accumulator.
MUL	Multiply the content of the integer accumulator with the single-precision integer operand.
DIV	Divide the content of the integer accumulator by the single-precision integer operand.
MOD	Divide the content of the integer accumulator by the single-precision integer operand and store the remainder in the accumulator.

(e) Relational Instructions (Basic IL)

Mnemonic	Description
GT	Set the boolean accumulator to True if the content of the integer accumulator is greater than the single-precision integer operand.
GE	Set the boolean accumulator to True if the content of the integer accumulator is greater than or equal to the single-precision integer operand.
EQ	Set the boolean accumulator to True if the content of the integer accumulator is equal to the single-precision integer operand.
NE	Set the boolean accumulator to True if the content of the integer accumulator is not equal to the single-precision integer operand.
LE	Set the boolean accumulator to True if the content of the integer accumulator is less than or equal to the single-precision integer operand.
LT	Set the boolean accumulator to True if the content of the integer accumulator is less than the single-precision integer operand.

(f) Program Flow Instructions (Basic IL)

Mnemonic	Description
CAL	Call subroutine block.
CALC	Call subroutine block if boolean accumulator is set to True.
CALCN	Call subroutine block if boolean accumulator is set to False.
END	Marks the absolute end of section of IL logic.
ENDMCRN	Marks the end of a section of IL logic that will not be solved. The section begins with a MCRN instruction with the same name assigned.
JMP	Transfer execution to label.
JMPC	Transfer execution to label if boolean accumulator is set to True.
JMPCN	Transfer execution to label if boolean accumulator is set to False.
MCRN	Marks the beginning of a section of IL logic that will be bypassed. The end of the section is marked with an END_MCRN instruction assigned the same name.
RET	Return from subroutine.
RETC	Return from subroutine if boolean accumulator is set to True.
RETCN	Return from subroutine if boolean accumulator is set to False.

5.2 IL TIMERS & COUNTERS

Timers

Use these IL functions to time intervals from instruction list logic.

Off Delay Timer
On Delay Stopwatch Timer
On Delay Timer

Counters

Use these IL functions to count events occurring in a process under control.

Up Counter
Down Counter

5.2.1 Off Delay Timer

5.2.1.1 OFDT_THOUS(*address, pv*)

The off-delay timer increments every thousandths of a second while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in thousandths of second.

5.2.1.2 OFDT_HUNDS(*address, pv*)

The off-delay timer increments every hundredth of a second while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in hundredths of second.

5.2.1.3 OFDT_TENTHS(*address, pv*)

The off-delay timer increments every tenth of a second while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in tenths of second.

5.2.1.4 OFDT_SECS(*address, pv*)

The off-delay timer increments every one second while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in one second.

5.2.1.5 OFDT_TENSEC(*address, pv*)

The off-delay timer increments every ten seconds while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in ten seconds.

5.2.1.6 OFDT_MIN(*address, pv*)

The off-delay timer increments every one minute while power flow is off, and resets to 0 when there is power flow. The timer passes power until the specified interval '*pv*' has elapsed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in one minute.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.2.2 On Delay Stopwatch Timer

5.2.2.1 ONDTR_THOUS(*address, r, pv*)

The on-delay stop watch timer increments every thousandth of a second while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in thousandths of second.
r: BOOL variable; resets '*cv*' to zero when True.

5.2.2.2 ONDTR_HUNDS(*address, r, pv*)

The on-delay stop watch timer increments every hundredth of a second while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in hundredths of second.
r: BOOL variable; resets '*cv*' to zero when True.

5.2.2.3 ONDTR_TENTHS(*address, r, pv*)

The on-delay stop watch timer increments every tenth of a second while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in tenth of second.
r: BOOL variable; resets '*cv*' to zero when True.

5.2.2.4 ONDTR_SECS(*address, r, pv*)

The on-delay stop watch timer increments every one second while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in one second.
r: BOOL variable; resets '*cv*' to zero when True.

5.2.2.5 ONDTR_TENSEC(*address, r, pv*)

The on-delay stop watch timer increments every ten seconds while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in ten seconds.
r: BOOL variable; resets '*cv*' to zero when True.

5.2.2.6 ONDTR_MIN(*address, r, pv*)

The on-delay stop watch timer increments every one minute while an accumulator is present, and holds its value when the accumulator is removed.
address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.
pv: INT variable or constant (0 - 32,767); preset value in one minute.
r: BOOL variable; resets '*cv*' to zero when True.

For more details, see LD Functions.
Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.2.3 On Delay Timer

5.2.3.1 TMR_THOUS(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in thousandths of second.

5.2.3.2 TMR_HUNDS(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in hundredths of second.

5.2.3.3 TMR_TENTHS(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in tenths of second.

5.2.3.4 TMR_SECS(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in one second.

5.2.3.5 TMR_TENSEC(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in ten seconds.

5.2.3.6 TMR_MIN(*address, pv*)

The standard timer increments while there is an accumulator present and resets to 0 when the accumulator is removed.

address: The first of three consecutive word registers. Word 1 - Current Value '*cv*', Word 2 - Preset Value '*pv*', Word 3 - Control Word.

pv: INT variable or constant (0 - 32,767); preset value in one minute.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.2.4 Up Counter

5.2.4.1 UPCTR(*address, r, pv*)

Each time UPCTR executes, it checks the current value of the Boolean accumulator. If that value has transitioned to ON, then UPCTR increments the current value 'CV'; otherwise, CV remains the same. When the 'PV' value is reached, the enable output turns ON and stays on until '*r*' input becomes True to reset 'CV' to zero.

address: WORD variable; the beginning address of a three-word WORD array. Word 1 - Current Value 'CV', Word 2 - Preset Value 'PV', Word 3 - Control Word.

r: BOOL variable; resets the counter when True.

pv: INT variable or constant (0 - 32,767); preset value.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.2.5 Down Counter

5.2.5.1 DNCTR(*address, r, pv*)

Each time DNCTR executes, it checks the current value of the Boolean accumulator. If that value has transitioned to ON, then DNCTR decrements the current value 'CV'; otherwise, CV remains the same. When the 'CV' value decrements to zero or less, the enable output turns ON and stays on until the 'r' input becomes True to reset 'CV' to the 'PV'.

address: WORD variable; the beginning address of a three-word WORD array. Word 1 - Current Value 'CV', Word 2 - Preset Value 'PV', Word 3 - Control Word.

r: BOOL variable; resets the counter when True.

pv: INT variable or constant (0 - 32,767); preset value.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3 IL MATH FUNCTIONS

Math Functions

Use these IL instructions to perform basic mathematical operations on data in instruction list logic.

Add
Subtract
Multiply
Divide
Modulus
Absolute
Square root

5.3.1 Add

5.3.1.1 ADD_SINT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two short integers and store the sum in the integer accumulator.

in1: SINT variable or constant

in2: SINT variable or constant

5.3.1.2 ADD_USINT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two unsigned short integers and store the sum in the integer accumulator.

in1: USINT variable or constant

in2: USINT variable or constant

5.3.1.3 ADD_INT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two single-precision integers and store the sum in the integer accumulator.

in1: INT variable or constant

in2: INT variable or constant

5.3.1.4 ADD_UINT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two single-precision unsigned integers and store the sum in the integer accumulator.

in1: UINT variable or constant

in2: UINT variable or constant

5.3.1.5 ADD_DINT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two double-precision integers and store the sum in the integer accumulator.

in1: DINT variable or constant

in2: DINT variable or constant

5.3.1.6 ADD_UDINT(*in1, in2*)

accumulator ← *in1* + *in2*

Add two double-precision unsigned integers and store the sum in the integer accumulator.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.2 Subtract

5.3.2.1 SUB_SINT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one short integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: SINT variable or constant

in2: SINT variable or constant

5.3.2.2 SUB_USINT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one unsigned short integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: USINT variable or constant

in2: USINT variable or constant

5.3.2.3 SUB_INT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one single-precision integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: INT variable or constant

in2: INT variable or constant

5.3.2.4 SUB_UINT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one single-precision unsigned integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: UINT variable or constant

in2: UINT variable or constant

5.3.2.5 SUB_DINT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one double-precision integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: DINT variable or constant

in2: DINT variable or constant

5.3.2.6 SUB_UDINT(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one double-precision unsigned integer from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.3 Multiply

5.3.3.1 MUL_SINT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one short integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: SINT variable or constant

in2: SINT variable or constant

5.3.3.2 MUL_USINT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one unsigned short integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: USINT variable or constant

in2: USINT variable or constant

5.3.3.3 MUL_INT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one single-precision integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: INT variable or constant

in2: INT variable or constant

5.3.3.4 MUL_UINT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one single-precision unsigned integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: UINT variable or constant

in2: UINT variable or constant

5.3.3.5 MUL_DINT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one double-precision integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: DINT variable or constant

in2: DINT variable or constant

5.3.3.6 MUL_UDINT(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one double-precision unsigned integer by another and store the product in the accumulator. The accumulator type is unchanged.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.4 Divide

5.3.4.1 DIV_SINT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one short integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: SINT variable or constant; the dividend

in2: SINT variable or constant; the divisor

5.3.4.2 DIV_USINT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one unsigned short integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: USINT variable or constant; the dividend

in2: USINT variable or constant; the divisor

5.3.4.3 DIV_INT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one single-precision integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: INT variable or constant; the dividend

in2: INT variable or constant; the divisor

5.3.4.4 DIV_UINT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one single-precision unsigned integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: UINT variable or constant; the dividend

in2: UINT variable or constant; the divisor

5.3.4.5 DIV_DINT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one double-precision integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: DINT variable or constant; the dividend

in2: DINT variable or constant; the divisor

5.3.4.6 DIV_UDINT(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one double-precision unsigned integer (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: UDINT variable or constant; the dividend

in2: UDINT variable or constant; the divisor

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.5 Modulus

5.3.5.1 MOD_SINT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one short integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: SINT variable or constant

in2: SINT variable or constant

5.3.5.2 MOD_USINT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one unsigned short integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: USINT variable or constant

in2: USINT variable or constant

5.3.5.3 MOD_INT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one single-precision integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: INT variable or constant

in2: INT variable or constant

5.3.5.4 MOD_UINT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one single-precision unsigned integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: UINT variable or constant

in2: UINT variable or constant

5.3.5.5 MOD_DINT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one double-precision integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: DINT variable or constant

in2: DINT variable or constant

5.3.5.6 MOD_UDINT(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]

Divide one double-precision unsigned integer by another and place the remainder in the accumulator. The accumulator type is unchanged.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.6 Absolute

5.3.6.1 ABS_SINT(*in*)

accumulator ← ABS(*in*)

Absolute short integers and store the result in the integer accumulator.

in: SINT variable or constant

5.3.6.2 ABS_INT(*in*)

accumulator ← ABS(*in*)

Absolute single-precision integers and store the result in the integer accumulator.

in: INT variable or constant

5.3.6.3 ABS_DINT(*in*)

accumulator ← ABS(*in*)

Absolute double-precision integers and store the result in the integer accumulator.

in: DINT variable or constant

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.3.7 Square Root

5.3.7.1 SQRT_SINT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a short integer and store the result in the integer accumulator.

in: SINT variable or constant

5.3.7.2 SQRT_USINT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a unsigned short integer and store the result in the integer accumulator.

in: USINT variable or constant

5.3.7.3 SQRT_INT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a single-precision integer and store the result in the integer accumulator.

in: INT, WORD variable or constant

5.3.7.4 SQRT_UINT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a single-precision unsigned integer and store the result in the integer accumulator.

in: UINT, WORD variable or constant

5.3.7.5 SQRT_DINT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a double-precision integer and store the result in the integer accumulator.

in: DINT variable or constant

5.3.7.6 SQRT_UDINT(*in*)

accumulator ← $\sqrt{\text{operand}}$

Take the square root of a double-precision unsigned integer and store the result in the integer accumulator.

in: UDINT variable or constant

For more details, see "4.3.4 SQRT_(type)"

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4 IL RELATIONAL FUNCTIONS

Relational Functions

Use these IL functions to compare the values of two operands without altering them or generating a sum or difference.

Equal
Not Equal
Greater Than
Greater or Equal
Less Than
Less or Equal
Range

5.4.1 Equal

5.4.1.1 EQ_SINT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare the two short integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: SINT variable or constant

in2: SINT variable or constant

5.4.1.2 EQ_USINT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare the two unsigned short integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: USINT variable or constant

in2: USINT variable or constant

5.4.1.3 EQ_INT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare the two single-precision integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: INT variable or constant

in2: INT variable or constant

5.4.1.4 EQ_UINT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare the two single-precision unsigned integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: UINT variable or constant

in2: UINT variable or constant

5.4.1.5 EQ_DINT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare two double-precision integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.1.6 EQ_UDINT(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare two double-precision unsigned integers and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: UDINT variable or constant

in2: UDINT variable or constant

5.4.1.7 EQ_BYTE(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare two BYTE data and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: BYTE variable or constant

in2: BYTE variable or constant

5.4.1.8 EQ_WORD(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare two WORD data and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: WORD variable or constant

in2: WORD variable or constant

5.4.1.9 EQ_DWORD(*in1, in2*)

If *in1* = *in2*, boolean accumulator ← 1

If *in1* ≠ *in2*, boolean accumulator ← 0

Compare two DWORD data and set the boolean accumulator if they are equal. Otherwise, clear the boolean accumulator.

in1: DWORD variable or constant

in2: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.2 Not Equal

5.4.2.1 NE_SINT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1
If *in1* = *in2*, boolean accumulator ← 0
Compare the values of two short integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.
in1: SINT variable or constant
in2: SINT variable or constant

5.4.2.2 NE_USINT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1
If *in1* = *in2*, boolean accumulator ← 0
Compare the values of two unsigned short integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.
in1: USINT variable or constant
in2: USINT variable or constant

5.4.2.3 NE_INT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1
If *in1* = *in2*, boolean accumulator ← 0
Compare the values of two single-precision integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.
in1: INT variable or constant
in2: INT variable or constant

5.4.2.4 NE_UINT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1
If *in1* = *in2*, boolean accumulator ← 0
Compare the values of two single-precision unsigned integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.
in1: UINT variable or constant
in2: UINT variable or constant

5.4.2.5 NE_DINT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1

If *in1* = *in2*, boolean accumulator ← 0

Compare the values of two double-precision integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.2.6 NE_UDINT(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1

If *in1* = *in2*, boolean accumulator ← 0

Compare the values of two double-precision unsigned integers and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UDINT variable or constant

in2: UDINT variable or constant

5.4.2.7 NE_BYTE(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1

If *in1* = *in2*, boolean accumulator ← 0

Compare the values of two BYTE data and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: BYTE variable or constant

in2: BYTE variable or constant

5.4.2.8 NE_WORD(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1

If *in1* = *in2*, boolean accumulator ← 0

Compare the values of two WORD data and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: WORD variable or constant

in2: WORD variable or constant

5.4.2.9 NE_DWORD(*in1, in2*)

If *in1* ≠ *in2*, boolean accumulator ← 1

If *in1* = *in2*, boolean accumulator ← 0

Compare the values of two DWORD data and set the boolean accumulator to 1 if they are not equal. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: DWORD variable or constant

in2: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.3 Greater Than

5.4.3.1 GT_SINT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two short integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: SINT variable or constant

in2: SINT variable or constant

5.4.3.2 GT_USINT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two unsigned short integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: USINT variable or constant

in2: USINT variable or constant

5.4.3.3 GT_INT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two single-precision integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: INT variable or constant

in2: INT variable or constant

5.4.3.4 GT_UINT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two single-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UINT variable or constant

in2: UINT variable or constant

5.4.3.5 GT_DINT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two double-precision integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.3.6 GT_UDINT(*in1, in2*)

If *in1* > *in2*, boolean accumulator ← 1

If *in1* ≤ *in2*, boolean accumulator ← 0

Compare the value of two double-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically greater than *in2*. Otherwise, set the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.4 Greater or Equal

5.4.4.1 GE_SINT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two short integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: SINT variable or constant

in2: SINT variable or constant

5.4.4.2 GE_USINT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two unsigned short integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: USINT variable or constant

in2: USINT variable or constant

5.4.4.3 GE_INT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two single-precision integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: INT variable or constant

in2: INT variable or constant

5.4.4.4 GE_UINT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two single-precision unsigned integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: UINT variable or constant

in2: UINT variable or constant

5.4.4.5 GE_DINT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two double-precision integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.4.6 GE_UDINT(*in1, in2*)

If *in1* ≥ *in2*, boolean accumulator ← 1

If *in1* < *in2*, boolean accumulator ← 0

Compare two double-precision unsigned integers. If *in1* is greater than or equal to *in2*, set the boolean accumulator. If *in1* is less than *in2*, clear the boolean accumulator.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.5 Less Than

5.4.5.1 LT_SINT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two short integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: SINT variable or constant

in2: SINT variable or constant

5.4.5.2 LT_USINT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two unsigned short integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: USINT variable or constant

in2: USINT variable or constant

5.4.5.3 LT_INT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two single-precision integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: INT variable or constant

in2: INT variable or constant

5.4.5.4 LT_UINT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two single-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UINT variable or constant

in2: UINT variable or constant

5.4.5.5 LT_DINT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two double-precision integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.5.6 LT_UDINT(*in1, in2*)

If *in1* < *in2*, boolean accumulator ← 1

If *in1* ≥ *in2*, boolean accumulator ← 0

Compare two double-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically less than *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.6 Less or Equal

5.4.6.1 LE_SINT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two short integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: SINT variable or constant

in2: SINT variable or constant

5.4.6.2 LE_USINT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two unsigned short integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: USINT variable or constant

in2: USINT variable or constant

5.4.6.3 LE_INT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two single-precision integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: INT variable or constant

in2: INT variable or constant

5.4.6.4 LE_UINT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two single-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UINT variable or constant

in2: UINT variable or constant

5.4.6.5 LE_DINT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two double-precision integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: DINT variable or constant

in2: DINT variable or constant

5.4.6.6 LE_UDINT(*in1, in2*)

If *in1* ≤ *in2*, boolean accumulator ← 1

If *in1* > *in2*, boolean accumulator ← 0

Compare the content of two double-precision unsigned integers and set the boolean accumulator to 1 if *in1* is numerically less than or equal to *in2*. Otherwise, clear the boolean accumulator to 0. The accumulator type is set to boolean.

in1: UDINT variable or constant

in2: UDINT variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.4.7 Range

5.4.7.1 RANGE_SINT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: SINT variable or constant; the lower limit of the range.

I2: SINT variable or constant; the upper limit of the range.

in: SINT variable or constant; the value under test

5.4.7.2 RANGE_USINT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: USINT variable or constant; the lower limit of the range.

I2: USINT variable or constant; the upper limit of the range.

in: USINT variable or constant; the value under test

5.4.7.3 RANGE_INT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: INT variable or constant; the lower limit of the range.

I2: INT variable or constant; the upper limit of the range.

in: INT variable or constant; the value under test

5.4.7.4 RANGE_UINT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: UINT variable or constant; the lower limit of the range.

I2: UINT variable or constant; the upper limit of the range.

in: UINT variable or constant; the value under test.

5.4.7.5 RANGE_DINT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: DINT variable or constant; the lower limit of the range.

I2: DINT variable or constant; the upper limit of the range.

in: DINT variable or constant; the value under test.

5.4.7.6 RANGE_UDINT(*I1, I2, in*)

If *I1* < *in* < *I2*, boolean accumulator ← 1

Otherwise, boolean accumulator ← 0

Test if a value is within a range limited by two other values.

I1: UDINT variable or constant; the lower limit of the range.

I2: UDINT variable or constant; the upper limit of the range.

in: UDINT variable or constant; the value under test.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5 IL BIT OPERATIONS

Bit Functions

Use these IL functions to perform bit-wise logical operations on binary strings (WORD/DWORD data).

Logical AND
Logical OR
Logical XOR
Logical NOT
Shift Bits Right, Shift Bits Left
Rotate Bits Right, Rotate Bits Left
Bit Test
Bit Set, Bit Clear
Bit Position
Bit Sequencer
Masked Compare

5.5.1 Logical AND

5.5.1.1 AND_BYTE(*in1, in2*)

accumulator ← *in1* AND *in2*
Perform a logical AND between each bit of *in1* and the same bit of *in2*. The corresponding bit of the integer accumulator is set or cleared depending on the result.
Operands: Both *in1* and *in2* can be either a BYTE variable or constant.

5.5.1.2 AND_WORD(*in1, in2*)

accumulator ← *in1* AND *in2*
Perform a logical AND between each bit of *in1* and the same bit of *in2*. The corresponding bit of the integer accumulator is set or cleared depending on the result.
Operands: Both *in1* and *in2* can be either a WORD variable or constant.

5.5.1.3 AND_DWORD(*in1, in2*)

accumulator ← *in1* AND *in2*
Perform a logical AND between each bit of *in1* and the same bit of *in2*. The corresponding bit of the integer accumulator is set or cleared depending on the result.
Operands: Both *in1* and *in2* can be either a DWORD variable or constant.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.2 Logical OR

5.5.2.1 OR_BYTE(*in1, in2*)

accumulator ← *in1* OR *in2*

Perform a logical OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either BYTE variables or constants.

5.5.2.2 OR_WORD(*in1, in2*)

accumulator ← *in1* OR *in2*

Perform a logical OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either WORD variables or constants.

5.5.2.3 OR_DWORD(*in1, in2*)

accumulator ← *in1* OR *in2*

Perform a logical OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either DWORD variables or constants.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.3 Logical XOR

5.5.3.1 XOR_BYTE(*in1, in2*)

accumulator ← *in1* XOR *in2*

Perform a logical exclusive OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either BYTE variables or constants.

5.5.3.2 XOR_WORD(*in1, in2*)

accumulator ← *in1* XOR *in2*

Perform a logical exclusive OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either WORD variables or constants.

5.5.3.3 XOR_DWORD(*in1, in2*)

accumulator ← *in1* XOR *in2*

Perform a logical exclusive OR between the bits of *in1* and *in2* and store the result in the integer accumulator.

Operands: Both *in1* and *in2* can be either DWORD variables or constants.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.4 Logical NOT

5.5.4.1 NOT_BYTE(*operand*)

accumulator ← *inverted operand*

Invert every bit of the operand and store the result in the integer accumulator.

Operand: BYTE variable

5.5.4.2 NOT_WORD(*operand*)

accumulator ← *inverted operand*

Invert every bit of the operand and store the result in the integer accumulator.

Operand: WORD variable

5.5.4.3 NOT_DWORD(*operand*)

accumulator ← *inverted operand*

Invert every bit of the operand and store the result in the integer accumulator.

Operand: DWORD variable

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.5 Shift Bits

5.5.5.1 SHIFTL_BYTE(*in, n, length, b1, q*)

Shift the bits in a BYTE, or string of BYTEs, left by a specified number of positions.
in: BYTE variable; the first BYTE in a string of BYTEs.
n: INT variable or constant; the number of positions to rotate.
length: Constant; the number (1 to 256) of BYTEs in the string.
b1: BOOL variable; the bits to shift into the string.
q: BYTE variable; the first BYTE of the shifted string.

5.5.5.2 SHIFTL_WORD(*in, n, length, b1, q*)

Shift the bits in a word, or string of words, left by a specified number of positions.
in: WORD variable; the first word in a string of words.
n: INT variable or constant; the number of positions to rotate.
length: Constant; the number (1 to 256) of words in the string.
b1: BOOL variable; the bits to shift into the string.
q: WORD variable; the first word of the shifted string.

5.5.5.3 SHIFTL_DWORD(*in, n, length, b1, q*)

Shift the bits in a DWORD, or string of DWORDs, left by a specified number of positions.
in: DWORD variable; the first DWORD in a string of DWORDs.
n: INT variable or constant; the number of positions to rotate.
length: Constant; the number (1 to 256) of DWORDs in the string.
b1: BOOL variable; the bits to shift into the string.
q: DWORD variable; the first DWORD of the shifted string.

5.5.5.4 SHIFTR_BYTE(*in, n, length, b1, q*)

Shift the bits in a BYTE, or string of BYTEs, right by a specified number of positions.
in: BYTE variable; the first BYTE in a string of BYTEs.
n: INT variable or constant; the number of positions to rotate.
length: Constant; the number of BYTEs in the string (1 to 256).
b1: BOOL variable; the bits to shift into the string.
q: BYTE variable; the first BYTE of the shifted string.

5.5.5.5 SHIFTR_WORD(*in, n, length, b1, q*)

Shift the bits in a word, or string of words, right by a specified number of positions.

in: WORD variable; the first word in a string of words.

n: INT variable or constant; the number of positions to rotate.

length: Constant; the number of words in the string (1 to 256).

b1: BOOL variable; the bits to shift into the string.

q: WORD variable; the first word of the shifted string.

5.5.5.6 SHIFTR_DWORD(*in, n, length, b1, q*)

Shift the bits in a DWORD, or string of DWORDs, right by a specified number of positions.

in: DWORD variable; the first DWORD in a string of DWORDs.

n: INT variable or constant; the number of positions to rotate.

length: Constant; the number of DWORDs in the string (1 to 256).

b1: BOOL variable; the bits to shift into the string.

q: DWORD variable; the first DWORD of the shifted string.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.6 Rotate Bits

5.5.6.1 ROL_BYTE(*in, n, length, q*)

Rotates the bits in a string of BYTEs *n* positions to the left. The most significant bit moves to the least significant bit.

in: BYTE variable; the first BYTE in a string of BYTEs.

n: INT variable or constant; the number of positions to rotate

length: Constant; the number of BYTEs in the string (1 to 256).

q: BYTE variable; the first BYTE of the rotated string.

5.5.6.2 ROL_WORD(*in, n, length, q*)

Rotates the bits in a string of words *n* positions to the left. The most significant bit moves to the least significant bit.

in: WORD variable; the first word in a string of words.

n: INT variable or constant; the number of positions to rotate

length: Constant; the number of words in the string (1 to 256).

q: WORD variable; the first word of the rotated string.

5.5.6.3 ROL_DWORD(*in, n, length, q*)

Rotates the bits in a string of DWORDs *n* positions to the left. The most significant bit moves to the least significant bit.

in: DWORD variable; the first DWORD in a string of DWORDs.

n: INT variable or constant; the number of positions to rotate

length: Constant; the number of DWORDs in the string (1 to 256).

q: DWORD variable; the first DWORD of the rotated string.

5.5.6.4 ROR_BYTE(*in, n, length, q*)

Rotates the bits in a string of BYTEs *n* positions to the right. The least significant bit moves to the most significant bit.

in: BYTE variable; the first BYTE in a string of BYTEs.

n: INT variable or constant; the number of positions to rotate.

length: Constant; the number of BYTEs in the string (1 to 256).

q: BYTE variable; the first BYTE of the rotated string.

5.5.6.5 ROR_WORD(*in, n, length, q*)

Rotates the bits in a string of words *n* positions to the right. The least significant bit moves to the most significant bit.

in: WORD variable; the first word in a string of words.

n: INT variable or constant; the number of positions to rotate.

length: Constant; the number of words in the string (1 to 256).

q: WORD variable; the first word of the rotated string.

5.5.6.6 ROR_DWORD(*in, n, length, q*)

Rotates the bits in a string of DWORDs *n* positions to the right. The least significant bit moves to the most significant bit.

in: DWORD variable; the first DWORD in a string of DWORDs.

n: INT variable or constant; the number of positions to rotate.

length: Constant; the number of DWORDs in the string (1 to 256).

q: DWORD variable; the first DWORD of the rotated string.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.7 Bit Test

5.5.7.1 BIT_TEST_BYTE(*in, bit, length*)

accumulator ← bit state

Returns the state of a specific bit within a string of BYTEs (consecutive registers) to the boolean accumulator.

in: BYTE variable; the first item in the data to be operated on.

bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (8*length)$.

length: Constant; the number of BYTEs comprising the data string to operate on (1 to 255).

5.5.7.2 BIT_TEST_WORD(*in, bit, length*)

accumulator ← bit state

Returns the state of a specific bit within a string of words (consecutive registers) to the boolean accumulator.

in: WORD variable; the first item in the data to be operated on.

bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (16*length)$.

length: Constant; the number of words comprising the data string to operate on (1 to 255).

5.5.7.3 BIT_TEST_DWORD(*in, bit, length*)

accumulator ← bit state

Returns the state of a specific bit within a string of DWORDs (consecutive registers) to the boolean accumulator.

in: DWORD variable; the first item in the data to be operated on.

bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (32*length)$.

length: Constant; the number of DWORDs comprising the data string to operate on (1 to 255).

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.8 Bit Set, Clear

5.5.8.1 BIT_SET_BYTE(*in, bit, length*)

Set a specific bit within a string (consecutive registers) of BYTEs by setting that bit to 1.
in: BYTE variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (8*length)$.
length: Constant; the number of BYTEs comprising the data string to operate on (1 to 255).

5.5.8.2 BIT_SET_WORD(*in, bit, length*)

Set a specific bit within a string (consecutive registers) of words by setting that bit to 1.
in: WORD variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (16*length)$.
length: Constant; the number of words comprising the data string to operate on (1 to 255).

5.5.8.3 BIT_SET_DWORD(*in, bit, length*)

Set a specific bit within a string (consecutive registers) of DWORDs by setting that bit to 1.
in: DWORD variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be set. $1 \leq bit \leq (32*length)$.
length: Constant; the number of DWORDs comprising the data string to operate on (1 to 255).

5.5.8.4 BIT_CLR_BYTE(*in, bit, length*)

Clear a specific bit within a string (consecutive registers) of BYTEs by setting that bit to 0.
in: BYTE variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be cleared. $1 \leq bit \leq (8*length)$.
length: Constant; the number (1 to 255) of BYTEs comprising the data string to operate on.

5.5.8.5 BIT_CLR_WORD(*in, bit, length*)

Clear a specific bit within a string (consecutive registers) of words by setting that bit to 0.
in: WORD variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be cleared. $1 \leq bit \leq (16*length)$.
length: Constant; the number (1 to 255) of words comprising the data string to operate on.

5.5.8.6 BIT_CLR_DWORD(*in, bit, length*)

Clear a specific bit within a string (consecutive registers) of DWORDs by setting that bit to 0.
in: DWORD variable; the first item in the data to be operated on.
bit: INT variable or constant; the bit number of *in* that will be cleared. $1 \leq bit \leq (32*length)$.
length: Constant; the number (1 to 255) of DWORDs comprising the data string to operate on.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.9 Bit Position

5.5.9.1 BIT_POS_BYTE(*in, length*)

accumulator ← bit number

Returns to the integer accumulator the bit number of the first bit found within a string of BYTEs (consecutive registers) set to 1.

in: BYTE variable; the first data item in the string to be operated on.

length: Constant; the number (1 of 255) of BYTEs comprising the data string to operate on.

5.5.9.2 BIT_POS_WORD(*in, length*)

accumulator ← bit number

Returns to the integer accumulator the bit number of the first bit found within a string of words (consecutive registers) set to 1.

in: WORD variable; the first data item in the string to be operated on.

length: Constant; the number (1 of 255) of words comprising the data string to operate on.

5.5.9.3 BIT_POS_DWORD(*in, length*)

accumulator ← bit number

Returns to the integer accumulator the bit number of the first bit found within a string of DWORDs (consecutive registers) set to 1.

in: DWORD variable; the first data item in the string to be operated on.

length: Constant; the number (1 of 255) of DWORDs comprising the data string to operate on

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.10 Bit Sequencer

5.5.10.1 BIT_SEQ(*address, r, dir, n, st, length*)

acummulator ← latest sequence value

Sequences a string of bit values, starting at *st*.

address: WORD variable; the first of three control words for the BIT_SEQ function (word 1:current step number, word 2: length of sequence in bits, word 3: control word).

r: BOOL variable; when *r* is true, the bit sequencer's step number is set to the value in *st* (default = 1), and the bit sequencer is filled with zeros, except for the current step number bit.

dir: BOOL variable; when *dir* is true, the bit sequencer's step number is incremented prior to the shift. Otherwise, it is decremented.

n: UINT, WORD variable or constant; when *r* is energized, the step number is set to this value (Optional).

st: BYTE; the first word of the bit sequencer (Optional).

length: Constant; the number of bits (starting at *st*) that this instruction will step through. Acceptable values range from 1-256.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.5.11 Masked Compare

5.5.11.1 MASK_COMP_BYTE(*in1, in2, m, bit, length, q, bn*)

Compare the bits in two strings of BYTEs (consecutive registers). The compare starts at the bit number *bit* and continues until a mismatch is found. When a mismatch is found the corresponding bit of *m* is set and then *q* is updated with the value of *m*.

in1: BYTE variable; the first data item in a string of BYTES.

in2: BYTE variable; the first data item in another string of BYTES.

m: BYTE variable; the mask containing the ongoing status of the compare.

bit: UINT variable or constant; the bit number where the compare begins.

length: Constant; the number (1 to 8192) of BYTEs in either compared string.

q: BYTE variable; the output copy of the compare mask.

bn: UINT variable; the number of the bit where the latest miscompare occurred.

5.5.11.2 MASK_COMP_WORD(*in1, in2, m, bit, length, q, bn*)

Compare the bits in two strings of words (consecutive registers). The compare starts at the bit number *bit* and continues until a mismatch is found. When a mismatch is found the corresponding bit of *m* is set and then *q* is updated with the value of *m*.

in1: WORD variable; the first data item in a string of WORDS.

in2: WORD variable; the first data item in another string of WORDS.

m: WORD variable; the mask containing the ongoing status of the compare.

bit: UINT variable or constant; the bit number where the compare begins.

length: Constant; the number (1 to 4096) of WORDs in either compared string.

q: WORD variable; the output copy of the compare mask.

bn: UINT variable; the number of the bit where the latest miscompare occurred.

5.5.11.3 MASK_COMP_DWORD(*in1, in2, m, bit, length, q, bn*)

Compare the bits in two strings of DWORDS (consecutive registers). The compare starts at the bit number *bit* and continues until a mismatch is found. When a mismatch is found the corresponding bit of *m* is set and then *q* is updated with the value of *m*.

in1: DWORD variable; the first data item in a string of DWORDS.

in2: DWORD variable; the first data item in another string of DWORDS.

m: DWORD variable; the mask containing the ongoing status of the compare.

bit: UINT variable or constant; the bit number where the compare begins.

length: Constant; the number (1 to 2048) of DWORDs in either compared string.

q: DWORD variable; the output copy of the compare mask.

bn: UINT variable; the number of the bit where the latest miscompare occurred.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.6 IL DATA MOVE FUNCTIONS

Data Move Functions

Use these IL functions to transfer data between various memory locations from instruction list logic.

Move Data
Swap Data
Block Clear
Shift Register

5.6.1 Move Data

5.6.1.1 MOVE_SINT(*in, length, q*)

Move a block of SINT data from one location in PMC memory to another.
in: SINT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: SINT variable; the location of the first destination data item.

5.6.1.2 MOVE_USINT(*in, length, q*)

Move a block of USINT data from one location in PMC memory to another.
in: USINT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: USINT variable; the location of the first destination data item.

5.6.1.3 MOVE_INT(*in, length, q*)

Move a block of INT data from one location in PMC memory to another.
in: INT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: INT variable; the location of the first destination data item.

5.6.1.4 MOVE_UINT(*in, length, q*)

Move a block of UINT data from one location in PMC memory to another.
in: UINT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: UINT variable; the location of the first destination data item.

5.6.1.5 MOVE_DINT(*in, length, q*)

Move a block of DINT data from one location in PMC memory to another.
in: DINT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: DINT variable; the location of the first destination data item.

5.6.1.6 MOVE_UDINT(*in, length, q*)

Move a block of UDINT data from one location in PMC memory to another.
in: UDINT variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: UDINT variable; the location of the first destination data item.

5.6.1.7 MOVE_BOOL(*in, length, q*)

Move a block of boolean data from one location in PMC memory to another.
in: BOOL variable; the location of the first source data item.
length: Constant; the number of data items to move.
q: BOOL variable; the location of the first destination data item.

5.6.1.8 MOVE_BYTE(*in, length, q*)

Move a block of BYTE data from one location in PMC memory to another.
in: BYTE variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: BYTE variable; the location of the first destination data item.

5.6.1.9 MOVE_WORD(*in, length, q*)

Move a block of WORD data from one location in PMC memory to another.
in: WORD variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: WORD variable; the location of the first destination data item.

5.6.1.10 MOVE_DWORD(*in, length, q*)

Move a block of DWORD data from one location in PMC memory to another.
in: DWORD variable; the location of the first source data item.
length: Constant; the number of data items to move (1 - 256).
q: DWORD variable; the location of the first destination data item.

For more details, see LD Functions.
Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.6.2 Swap Data

5.6.2.1 SWAP_WORD(*in, length, q*)

Swap a block of WORD data from one location in PMC memory to another.

in: WORD variable; the location of the first source data item.

length: Constant; the number of data items to move (1 - 256).

q: WORD variable; the location of the first destination data item.

5.6.2.2 SWAP_DWORD(*in, length, q*)

Swap a block of DWORD data from one location in PMC memory to another.

in: DWORD variable; the location of the first source data item.

length: Constant; the number of data items to move (1 - 256).

q: DWORD variable; the location of the first destination data item.

5.6.3 Block Clear

5.6.3.1 BLK_CLR_SINT(*in, length*)

Clear a block of memory to 0.

in: SINT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.2 BLK_CLR_USINT(*in, length*)

Clear a block of memory to 0.

in: USINT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.3 BLK_CLR_INT(*in, length*)

Clear a block of memory to 0.

in: INT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.4 BLK_CLR_UINT(*in, length*)

Clear a block of memory to 0.

in: UINT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.5 BLK_CLR_DINT(*in, length*)

Clear a block of memory to 0.

in: DINT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.6 BLK_CLR_UDINT(*in, length*)

Clear a block of memory to 0.

in: UDINT variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.7 BLK_CLR_BYTE(*in, length*)

Clear a block of memory to 0.

in: BYTE variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.8 BLK_CLR_WORD(*in, length*)

Clear a block of memory to 0.

in: WORD variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

5.6.3.9 BLK_CLR_DWORD(*in, length*)

Clear a block of memory to 0.

in: DWORD variable; starting location of memory block to be cleared. The length field for this variable must be between 1 and 256 words.

length: Constant; the number of words (starting at *in*) that will be cleared. Acceptable values range from 1-256.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.6.4 Shift Register

5.6.4.1 SHFR_BIT(*r, in, st, length, q*)

If the Boolean Accumulator is set to ON (the transition state does not matter), SHFR_BIT shifts the content in a block of BOOL memory from one register to the next. The contiguous "length" number of bits starting at section of memory "st" serves as a shift register.

r: BOOL variable; when true (ON), the shift register is filled with zeroes (is set to false, OFF).

in: BOOL, WORD variable or constant; the bit value to be shifted into the shift register. Only the first bit of a WORD is used.

st: BOOL, WORD variable; the address of the first bit of the shift register.

length: Constant; the number (1 - 256) of bits in the shift register.

q: BOOL, INT, WORD variable; the value of the bit shifted out of the shift register. If q is a WORD variable, only the first bit is used.

5.6.4.2 SHFR_BYTE(*r, in, st, length, q*)

Shift the content in a block of BYTE memory, from one register to the next. The contiguous section of memory serves as a shift register.

r: BOOL variable; when True the shift register is filled with zeroes.

in: BYTE variable or constant; the value to be shifted into the shift register.

st: BYTE variable; the first element of the shift register.

length: Constant; the number (1 - 256) of elements in the shift register.

q: BYTE variable; the value shifted out of the shift register.

5.6.4.3 SHFR_WORD(*r, in, st, length, q*)

Shift the content in a block of WORD memory, from one register to the next. The contiguous section of memory serves as a shift register.

r: BOOL variable; when True the shift register is filled with zeroes.

in: WORD variable or constant; the value to be shifted into the shift register.

st: WORD variable; the first element of the shift register.

length: Constant; the number (1 - 256) of elements in the shift register.

q: WORD variable; the value shifted out of the shift register.

5.6.4.4 SHFR_DWORD(*r, in, st, length, q*)

Shift the content in a block of double WORD memory, from one register to the next. The contiguous section of memory serves as a shift register.

r: BOOL variable; when True the shift register is filled with zeroes.

in: DWORD variable or constant; the value to be shifted into the shift register.

st: DWORD variable; the first element of the shift register.

length: Constant; the number (1 - 256) of elements in the shift register.

q: DWORD variable; the value shifted out of the shift register.

5.7 IL DATA TABLE FUNCTIONS

Data Table Functions

Use these IL functions to copy a specified number of data elements from a source array to a destination array.

Array Move
Search for values in a Memory Block

5.7.1 Array Move

5.7.1.1 ARRAY_MOVE_SINT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: SINT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: SINT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.2 ARRAY_MOVE_USINT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: USINT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: USINT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.3 ARRAY_MOVE_INT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: INT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: INT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.4 ARRAY_MOVE_UINT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: UINT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: UINT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.5 ARRAY_MOVE_DINT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: DINT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: DINT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.6 ARRAY_MOVE_UDINT(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: UDINT variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: UDINT variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.7 ARRAY_MOVE_BOOL(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: BOOL, variable; the first element of the source array. For an Array Move with the data type BOOL, any reference may be used; it does not need to be byte aligned.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: BOOL, variable; the first element of the destination array. For an Array Move with the data type BOOL, any reference may be used. It does not need to be byte aligned.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.8 ARRAY_MOVE_BYTE(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: BYTE variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: BYTE variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.9 ARRAY_MOVE_WORD(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: WORD variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: WORD variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

5.7.1.10 ARRAY_MOVE_DWORD(*sr, snx, dnx, n, length, ds*)

When the function receives power flow, the number of data elements in the count indicator '*n*' is extracted from the input array starting with the indexed location ('*sr*' + '*snx*' – 1). The data elements are written to the output array starting with the indexed location ('*ds*' + '*dnx*' – 1).

sr: DWORD variable; the first element of the source array.

snx: UINT, variable or constant; the index into the source array.

dnx: UINT, variable or constant; the index into the destination array.

n: UINT, variable or constant; count indicator.

ds: DWORD variable; the first element of the destination array.

length: Constant; the number (1 - 32767) of elements (starting at '*sr*' and '*ds*') that make up each array.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.7.2 Search for Values in a Memory Block

5.7.2.1 Mnemonics

The Search function supports 56 mnemonics:

Mnemonic	Description
SEARCH_EQ_BOOL SEARCH_EQ_BYTE SEARCH_EQ_WORD SEARCH_EQ_DWORD SEARCH_EQ_SINT SEARCH_EQ_USINT SEARCH_EQ_INT SEARCH_EQ_UINT SEARCH_EQ_DINT SEARCH_EQ_UDINT	Searches memory block for all values equal to a specified value.
SEARCH_NE_BOOL SEARCH_NE_BYTE SEARCH_NE_WORD SEARCH_NE_DWORD SEARCH_NE_SINT SEARCH_NE_USINT SEARCH_NE_INT SEARCH_NE_UINT SEARCH_NE_DINT SEARCH_NE_UDINT	Searches memory block for all values not equal to a specified value.
SEARCH_GT_BYTE SEARCH_GT_WORD SEARCH_GT_DWORD SEARCH_GT_SINT SEARCH_GT_USINT SEARCH_GT_INT SEARCH_GT_UINT SEARCH_GT_DINT SEARCH_GT_UDINT	Searches memory block for all values greater than a specified value.
SEARCH_GE_BYTE SEARCH_GE_WORD SEARCH_GE_DWORD SEARCH_GE_SINT SEARCH_GE_USINT SEARCH_GE_INT SEARCH_GE_UINT SEARCH_GE_DINT SEARCH_GE_UDINT	Searches memory block for all values greater than or equal to a specified value
SEARCH_LT_BYTE SEARCH_LT_WORD SEARCH_LT_DWORD SEARCH_LT_SINT SEARCH_LT_USINT SEARCH_LT_INT SEARCH_LT_UINT SEARCH_LT_DINT SEARCH_LT_UDINT	Searches memory block for all values less than a specified value.

Mnemonic	Description
SEARCH_LE_BYTE SEARCH_LE_WORD SEARCH_LE_DWORD SEARCH_LE_SINT SEARCH_LE_USINT SEARCH_LE_INT SEARCH_LE_UINT SEARCH_LE_DINT SEARCH_LE_UDINT	Searches memory block for all values less than or equal to a specified value.

5.7.2.2 Operation

The Search function searches the specified memory block for a value. Searching begins at ***ar***+***inx***, where ***ar*** is the starting address and ***inx*** is the index value into the memory block. The search continues either until a register equal to the search object (***in***) is found or until the end of the memory block is reached.

If a register is found, the boolean accumulator is set ON and the Output Index (***onx***) is set to the relative position of this register within the block.

If no register is found before the end of the block is reached, the boolean accumulator is set OFF and ***onx*** is set to zero.

inx is zero-based, that is, 0 the means first reference, whereas ***onx*** is one-based, that is, 1 means the first reference.

The valid values for ***inx*** are 0 to (***length*** - 1). The valid values for ***onx*** are 1 to ***length***.

inx should be set to zero to begin searching at the memory block's first register. This value increments by one at the time of execution. If the value of ***inx*** is out-of-range, (< 0 or > ***length***-1), ***inx*** is set to the default value of zero.

5.7.2.3 Operands

Operand	Data Type	Memory Area	Description
Length	Constant		The number of registers starting at ***ar*** that make up the memory block to search. 1 ≤ ?? ≤ 32767.
ar (must be the same data type as ***in***)	BOOL	%I*, %Q*,%M*,%S*	The starting address of the memory block to search; the address of the first register in the memory block.
	BYTE	%I*, %Q*,%M*,%S*,%R	
	WORD	%I*, %Q*,%M*,%S*,%R	
	DWORD	%I*, %Q*,%M*,%S*,%R	
	SINT	%I*, %Q*,%M*,%S*,%R	
	USINT	%I*, %Q*,%M*,%S*,%R	
	INT	%I*, %Q*,%M*,%S*,%R	
	UINT	%I*, %Q*,%M*,%S*,%R	
	DINT	%I*, %Q*,%M*,%S*,%R	
	UDINT	%I*, %Q*,%M*,%S*,%R	
Inx	UINT variable or constant	%I*, %Q*,%M*,%S*,%R	The zero-based index into the memory block at which to begin the search. Zero points to the first reference. Valid range: 0 ≤ ***inx*** ≤ (***length***-1). If ***inx*** is out of range, it is set to the default value of 0.
in (must be the same data type as ***ar***)	BOOL variable or constant	%I*, %Q*,%M*,%S*	The object of the search; the value to search for.
	BYTE variable or constant	%I*, %Q*,%M*,%S*,%R	
	WORD variable or constant	%I*, %Q*,%M*,%S*,%R	
	DWORD variable or constant	%I*, %Q*,%M*,%S*,%R	
	SINT variable or constant	%I*, %Q*,%M*,%S*,%R	
	USINT variable or constant	%I*, %Q*,%M*,%S*,%R	
	INT variable or constant	%I*, %Q*,%M*,%S*,%R	
	UINT variable or constant	%I*, %Q*,%M*,%S*,%R	
	DINT variable or constant	%I*, %Q*,%M*,%S*,%R	
	UDINT variable or constant	%I*, %Q*,%M*,%S*,%R	
onx	UINT variable	%I*, %Q*,%M*,%S*,%R	The one-based position within the memory block of the search target. A value of 1 points to the first reference. Valid range: 1 ≤ ***onx*** ≤ ***length***.

5.8 IL CONVERSIONS

Conversion Functions

Use these IL functions to convert data types, number formats and units in instruction list logic.

(type)_TO_BCDx(x=2,4,8)
BCDx_TO_(type)(x=2,4,8)
(type)_TO_SINT
(type)_TO_USINT
(type)_TO_INT
(type)_TO_UINT
(type)_TO_DINT
(type)_TO_UDINT

5.8.1 (type)_TO_BCDx(x=2,4,8)

5.8.1.1 SINT_TO_BCD2(*operand*)

accumulator ← converted value

Convert a short integer to binary coded decimal format.

Operand: SINT variable; the value to be converted.

5.8.1.2 USINT_TO_BCD2(*operand*)

accumulator ← converted value

Convert an unsigned short integer to binary coded decimal format.

Operand: USINT variable; the value to be converted.

5.8.1.3 INT_TO_BCD4(*operand*)

accumulator ← converted value

Convert a single-precision integer to binary coded decimal format.

Operand: INT variable; the value to be converted.

5.8.1.4 UINT_TO_BCD4(*operand*)

accumulator ← converted value

Convert a single-precision unsigned integer to binary coded decimal format.

Operand: UINT variable; the value to be converted.

5.8.1.5 DINT_TO_BCD8(*operand*)

accumulator ← converted value

Convert a double-precision integer to binary coded decimal format.

Operand: DINT variable; the value to be converted.

5.8.1.6 UDINT_TO_BCD8(*operand*)

accumulator ← converted value

Convert a double-precision unsigned integer to binary coded decimal format.

Operand: UDINT variable; the value to be converted.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.2 BCDx_TO_(type)(x=2,4,8)

5.8.2.1 BCD2_TO_SINT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to a short integer value and store the result in the integer accumulator.

Operand: BYTE variable; the BCD value for conversion.

5.8.2.2 BCD2_TO_USINT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to an unsigned short integer value and store the result in the integer accumulator.

Operand: BYTE variable; the BCD value for conversion.

5.8.2.3 BCD4_TO_INT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to a single-precision integer value and store the result in the integer accumulator.

Operand: WORD variable; the BCD value for conversion.

5.8.2.4 BCD4_TO_UINT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: WORD variable; the BCD value for conversion.

5.8.2.5 BCD8_TO_DINT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to a double-precision integer value and store the result in the integer accumulator.

Operand: DWORD variable; the BCD value for conversion.

5.8.2.6 BCD8_TO_UDINT(*operand*)

accumulator ← converted value

Convert a binary coded decimal value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: DWORD variable; the BCD value for conversion.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.3 (type)_TO_SINT

5.8.3.1 USINT_TO_SINT(*operand*)

accumulator ← converted value

Convert an unsigned short integer value to a short integer value and store the result in the integer accumulator.

Operand: USINT variable;

5.8.3.2 INT_TO_SINT(*operand*)

accumulator ← converted value

Convert a single-precisio integer value to a short integer value and store the result in the integer accumulator.

Operand: INT variable;

5.8.3.3 UINT_TO_SINT(*operand*)

accumulator ← converted value

Convert a single-precision unsigned integer value to a short integer value and store the result in the integer accumulator.

Operand: UINT variable;

5.8.3.4 DINT_TO_SINT(*operand*)

accumulator ← converted value

Convert a double-precision integer value to a short integer value and store the result in the integer accumulator.

Operand: DINT variable;

5.8.3.5 UDINT_TO_SINT(*operand*)

accumulator ← converted value

Convert a double-precision unsigned integer value to a short integer value and store the result in the integer accumulator.

Operand: UDINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.4 (type)_TO_USINT

5.8.4.1 SINT_TO_USINT(*operand*)

accumulator ← converted value
Convert a short integer value to an unsigned short integer value and store the result in the integer accumulator.
Operand: SINT variable;

5.8.4.2 INT_TO_USINT(*operand*)

accumulator ← converted value
Convert a single-precision integer value to an unsigned short integer value and store the result in the integer accumulator.
Operand: INT variable;

5.8.4.3 UINT_TO_USINT(*operand*)

accumulator ← converted value
Convert a single-precision unsigned integer value to an unsigned short integer value and store the result in the integer accumulator.
Operand: UINT variable;

5.8.4.4 DINT_TO_USINT(*operand*)

accumulator ← converted value
Convert a double-precision integer value to an unsigned short integer value and store the result in the integer accumulator.
Operand: DINT variable;

5.8.4.5 UDINT_TO_USINT(*operand*)

accumulator ← converted value
Convert a double-precision unsigned integer value to an unsigned short integer value and store the result in the integer accumulator.
Operand: UDINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.5 (type)_TO_INT

5.8.5.1 SINT_TO_INT(*operand*)

accumulator ← converted value

Convert a short integer value to a single-precision integer value and store the result in the integer accumulator.

Operand: SINT variable;

5.8.5.2 USINT_TO_INT(*operand*)

accumulator ← converted value

Convert an unsigned short integer value to a single-precision integer value and store the result in the integer accumulator.

Operand: USINT variable;

5.8.5.3 UINT_TO_INT(*operand*)

accumulator ← converted value

Convert a single-precision unsigned integer value to a single-precision integer value and store the result in the integer accumulator.

Operand: UINT variable;

5.8.5.4 DINT_TO_INT(*operand*)

accumulator ← converted value

Convert a double-precision integer value to a single-precision integer value and store the result in the integer accumulator.

Operand: DINT variable;

5.8.5.5 UDINT_TO_INT(*operand*)

accumulator ← converted value

Convert a double-precision unsigned integer value to a single-precision integer value and store the result in the integer accumulator.

Operand: UDINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.6 (type)_TO_UINT

5.8.6.1 SINT_TO_UINT(*operand*)

accumulator ← converted value

Convert a short integer value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: SINT variable;

5.8.6.2 USINT_TO_UINT(*operand*)

accumulator ← converted value

Convert an unsigned short integer value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: USINT variable;

5.8.6.3 INT_TO_UINT(*operand*)

accumulator ← converted value

Convert a single-precision integer value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: INT variable;

5.8.6.4 DINT_TO_UINT(*operand*)

accumulator ← converted value

Convert a double-precision integer value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: DINT variable;

5.8.6.5 UDINT_TO_UINT(*operand*)

accumulator ← converted value

Convert a double-precision unsigned integer value to a single-precision unsigned integer value and store the result in the integer accumulator.

Operand: UDINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.7 (type)_TO_DINT

5.8.7.1 SINT_TO_DINT(*operand*)

accumulator ← converted value

Convert a short integer value to a double-precision integer value and store the result in the integer accumulator.

Operand: SINT variable;

5.8.7.2 USINT_TO_DINT(*operand*)

accumulator ← converted value

Convert an unsigned short integer value to a double-precision integer value and store the result in the integer accumulator.

Operand: USINT variable;

5.8.7.3 INT_TO_DINT(*operand*)

accumulator ← converted value

Convert a single-precision integer value to a double-precision integer value and store the result in the integer accumulator.

Operand: INT variable;

5.8.7.4 UINT_TO_DINT(*operand*)

accumulator ← converted value

Convert a single-precision unsigned integer value to a double-precision integer value and store the result in the integer accumulator.

Operand: UINT variable;

5.8.7.5 UDINT_TO_DINT(*operand*)

accumulator ← converted value

Convert a double-precision unsigned integer value to a double-precision integer value and store the result in the integer accumulator.

Operand: UDINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.8.8 (type)_TO_UDINT

5.8.8.1 SINT_TO_UDINT(*operand*)

accumulator ← converted value

Convert a short integer value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: SINT variable;

5.8.8.2 USINT_TO_UDINT(*operand*)

accumulator ← converted value

Convert an unsigned short integer value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: USINT variable;

5.8.8.3 INT_TO_UDINT(*operand*)

accumulator ← converted value

Convert a single-precision integer value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: INT variable;

5.8.8.4 UINT_TO_UDINT(*operand*)

accumulator ← converted value

Convert a single-precision unsigned integer value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: UINT variable;

5.8.8.5 DINT_TO_UDINT(*operand*)

accumulator ← converted value

Convert a double-precision integer value to a double-precision unsigned integer value and store the result in the integer accumulator.

Operand: DINT variable;

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9 IL PMC OPERATIONS

PMC Operations

Use these IL functions to operate PMC.

Add BCD
Subtract BCD
Multiply BCD
Divide BCD
Modulus BCD
Trigger
Decode
Even Parity
Odd Parity
Window
EXIN
AXCTL

5.9.1 Add BCD

5.9.1.1 PMC_ADD_BCD2(*in1, in2*)

accumulator ← *in1* + *in2*

Add two BCD2 and store the sum in the integer accumulator.

in1: BYTE variable or constant

in2: BYTE variable or constant

5.9.1.2 PMC_ADD_BCD4(*in1, in2*)

accumulator ← *in1* + *in2*

Add two BCD4 and store the sum in the integer accumulator.

in1: WORD variable or constant

in2: WORD variable or constant

5.9.1.3 PMC_ADD_BCD8(*in1, in2*)

accumulator ← *in1* + *in2*

Add two BCD8 and store the sum in the integer accumulator.

in1: DWORD variable or constant

in2: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.2 Subtract BCD

5.9.2.1 PMC_SUB_BCD2(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one BCD2 from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: BYTE variable or constant

in2: BYTE variable or constant

5.9.2.2 PMC_SUB_BCD4(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one BCD4 from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: WORD variable or constant

in2: WORD variable or constant

5.9.2.3 PMC_SUB_BCD8(*in1, in2*)

accumulator ← *in1* - *in2*

Subtract one BCD8 from another and store the result in the accumulator. The value and type of the accumulator are unchanged.

in1: DWORD variable or constant

in1: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.3 Multiply BCD

5.9.3.1 PMC_MUL_BCD2(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one BCD2 by another and store the product in the accumulator. The accumulator type is unchanged.

in1: BYTE variable or constant

in2: BYTE variable or constant

5.9.3.2 PMC_MUL_BCD4(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one BCD4 by another and store the product in the accumulator. The accumulator type is unchanged.

in1: WORD variable or constant

in2: WORD variable or constant

5.9.3.3 PMC_MUL_BCD8(*in1, in2*)

accumulator ← *in1* * *in2*

Multiply one BCD8 by another and store the product in the accumulator. The accumulator type is unchanged.

in1: DWORD variable or constant

in2: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.4 Divide BCD

5.9.4.1 PMC_DIV_BCD2(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one BCD2 (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: BYTE variable or constant; the dividend

in2: BYTE variable or constant; the divisor

5.9.4.2 PMC_DIV_BCD4(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one BCD4 (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: WORD variable or constant; the dividend

in2: WORD variable or constant; the divisor

5.9.4.3 PMC_DIV_BCD8(*in1, in2*)

accumulator ← *in1*/*in2*

Divide one BCD8 (*in1*) by another (*in2*) and store the quotient in the integer accumulator.

in1: DWORD variable or constant; the dividend

in2: DWORD variable or constant; the divisor

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.5 Modulus BCD

5.9.5.1 PMC_MOD_BCD2(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]
Divide one BCD2 by another and place the remainder in the accumulator. The accumulator type is unchanged
in1: BYTE variable or constant
in2: BYTE variable or constant

5.9.5.2 PMC_MOD_BCD4(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]
Divide one BCD4 by another and place the remainder in the accumulator. The accumulator type is unchanged
in1: WORD variable or constant
in2: WORD variable or constant

5.9.5.3 PMC_MOD_BCD8(*in1, in2*)

accumulator ← REMAINDER [*in1*/*in2*]
Divide one BCD8 by another and place the remainder in the accumulator. The accumulator type is unchanged
in1: DWORD variable or constant
in2: DWORD variable or constant

For more details, see LD Functions.
Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.6 Trigger

5.9.6.1 R_TRIG(*in*)

When accumulator is rising (going from OFF to ON), R_TRIG sets ENO to ON for one sweep only.
in: BOOL variable for detection.

5.9.6.2 F_TRIG(*in*)

When accumulator is falling (going from ON to OFF), F_TRIG sets ENO to ON for one sweep only.
in: BOOL variable for detection.

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.7 Decode

5.9.7.1 PMC_DECODE_SINT(*in, base, n, q*)

PMC_DECODE_SINT extracts continuous 8 x n (depending on the array size of output parameter *q*) integer data from the integer data of input parameter *in* and that of *base*, decodes it, then sets the corresponding bit of output parameter *q*.

in: SINT variable or constant

base: SINT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

5.9.7.2 PMC_DECODE_USINT(*in, base, n, q*)

PMC_DECODE_USINT extracts continuous 8 x n (depending on the array size of output parameter *q*) integer data from the integer data of input parameter *in* and that of *base*, decodes it, then sets the corresponding bit of output parameter *q*.

in: USINT variable or constant

base: USINT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

5.9.7.3 PMC_DECODE_INT(*in, base, n, q*)

PMC_DECODE_INT extracts continuous 16 x n (depending on the array size of output parameter *q*) integer data from the integer data of input parameter *in* and that of *base*, decodes it, then sets the corresponding bit of output parameter *q*.

in: INT variable or constant

base: INT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

5.9.7.4 PMC_DECODE_UINT(*in, base, n, q*)

PMC_DECODE_UINT extracts continuous 16 x n (depending on the array size of output parameter ***q***) integer data from the integer data of input parameter ***in*** and that of ***base***, decodes it, then sets the corresponding bit of output parameter ***q***.

in: UINT variable or constant

base: UINT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

5.9.7.5 PMC_DECODE_DINT(*in, base, n, q*)

PMC_DECODE_DINT extracts continuous 32 x n (depending on the array size of output parameter ***q***) integer data from the integer data of input parameter ***in*** and that of ***base***, decodes it, then sets the corresponding bit of output parameter ***q***.

in: DINT variable or constant

base: DINT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

5.9.7.6 PMC_DECODE_UDINT(*in, base, n, q*)

PMC_DECODE_UDINT extracts continuous 32 x n (depending on the array size of output parameter ***q***) integer data from the integer data of input parameter ***in*** and that of ***base***, decodes it, then sets the corresponding bit of output parameter ***q***.

in: UDINT variable or constant

base: UDINT variable or constant; Decode indication number

n: constant

q: WORD [n] variable; Result of decoding

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.8 Check Even Parity

5.9.8.1 PMC_EVPAR_BYTE(*in1*)

accumulator ← Result of check

Check the even parity of BYTE data.

in1: BYTE variable or constant

5.9.8.2 PMC_EVPAR_WORD(*in1*)

accumulator ← Result of check

Check the even parity of WORD data.

in1: WORD variable or constant

5.9.8.3 PMC_EVPAR_DWORD(*in1*)

accumulator ← Result of check

Check the even parity of DWORD data.

in1: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.9 Check Odd Parity

5.9.9.1 PMC_ODPAR_BYTE(*in1*)

accumulator ← Result of check

Check the odd parity of BYTE data.

in1: BYTE variable or constant

5.9.9.2 PMC_ODPAR_WORD(*in1*)

accumulator ← Result of check

Check the odd parity of WORD data.

in1: WORD variable or constant

5.9.9.3 PMC_ODPAR_DWORD(*in1*)

accumulator ← Result of check

Check the odd parity of DWORD data.

in1: DWORD variable or constant

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.10 Window

5.9.10.1 PMC_WINDOW(*ar, n, err*)

accumulator ← Completion flag

The CNC ←→ PMC window function is used to read and write the CNC data if boolean accumulator is set to True.

ar: INT variable; AR array

n: constant. Length of ar(INT) array.

err: BOOL variable; Error output

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.11 EXIN

5.9.11.1 PMC_EXIN(head, cmd, dt, err)

accumulator ← Completion flag

Issues commands for the CNC external data input function if boolean accumulator is set to True.

head: INT variable or constant; HEAD NO.

cmd: WORD variable; Command code

dt: DWORD variable; Command data

err: BOOL variable; Error output

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.9.12 AXCTL

5.9.12.1 PMC_AXCTL(r, grp, cmd, dt1, dt2, err)

accumulator ← Completion flag

Specifies the PMC axis control function of the CNC if boolean accumulator is set to True.

r: BOOL variable; Reset

grp: INT variable or constant; DI/DO signal group

cmd: WORD variable or constant; Axis control command

dt1: DINT variable or constant; Instruction data 1

dt2: DINT variable or constant; Instruction data 2

err: BOOL variable or constant; Error output

For more details, see LD Functions.

Note: IL and LD functions are generally the same, with some differences in data types for operands.

5.10 NOTE ON IL PROGRAMMING

Some functional instructions may cause the ladder program to take a long time to stop or make it unable to stop, if their boolean accumulator or r (Reset) condition remains on for no apparent reason. If the ladder program does not stop, all operations aimed at changing the ladder program will take longer to end or will never end.
To avoid such problems, when you code a ladder program using functional instructions, you need to design the ladder structure based on a thorough understanding of the control conditions of the individual instructions you use.
Listed below are typical cases in which the ladder program will not stop.

- A low‐speed window function is used for a PMC_WINDOW functional instruction, and its boolean accumulator condition remains on.
- In a PMC_EXIN or PMC_AXCTL instruction, not only the boolean accumulator condition but also the r (Reset) condition remain on.

If the ladder program takes long to stop or does not stop for any of these reasons, the following operations will be affected.

1. Stopping the ladder program using a soft key on the screen.
2. Reading a new ladder program from a memory card or other medium, by using the data input and output screen.
3. Updating the ladder program with changes made using the FANUC LADDER-IIIC.

If any of the above phenomena occurs, the functional instruction causing the problem needs to be fixed. Check the functional instructions mentioned above to see whether there is any boolean accumulator or r (Reset) condition remaining on, and correct the ladder program according to the following procedure.

1. Put the machine in safe condition and turn off the power of the NC.
2. Turn on the power of the NC while holding down the "CAN" and "Z" keys simultaneously, to restart the NC with the ladder program halted.
3. In the FANUC LADDER-IIIC, redesign the logic associated with the problematic functional instruction. When done, set the boolean accumulator or r (Reset) condition to off.
4. Write the resulting logic to flash ROM using the I/O screen.
5. Run the ladder program.

If the ladder program does not stop or cannot be changed even after you make the correction, there may be other functional instructions that have the same condition settings.
Check for other functional instructions having the same condition settings, besides the one you have corrected, and repeat the above procedure to correct them all.

6 OPERATION

6.1 OVERVIEW

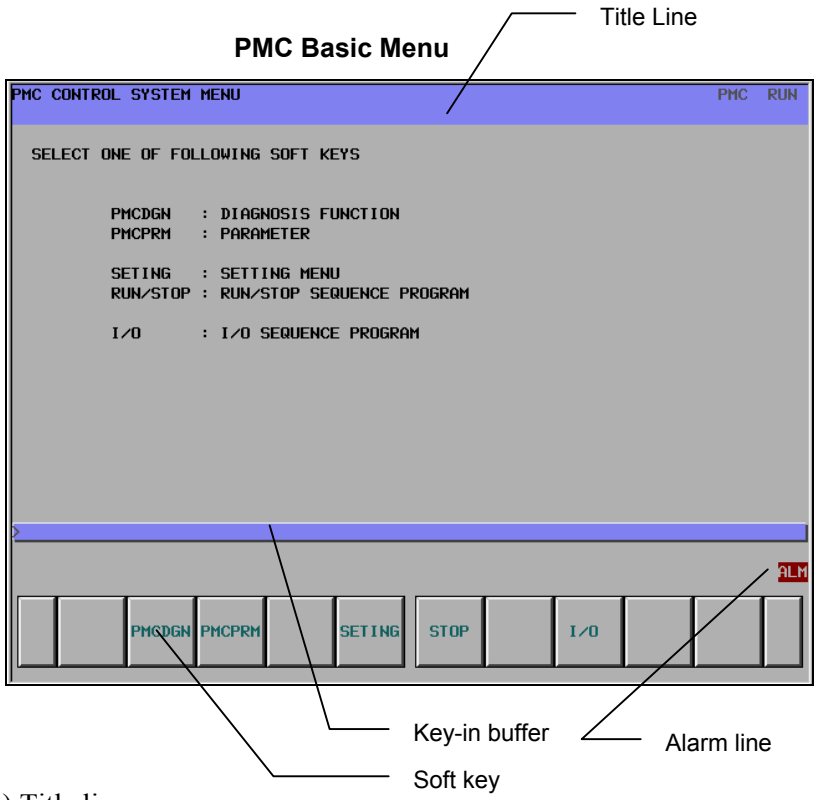
In the NC system, pressing the [PMC] soft key on the [SYSTEM] menu enables the setting and display of data related to the PMC. The following screens are used to specify and display the PMC-related data.

(1) Displaying PMC input/output signals and internal relay (PMCDGN)
  (a) Title data screen
  (b) Status screen
  (c) Alarm screen
  (d) Trace screen
  (e) I/O Link Monitor screen
(2) PMC data setting and display (PMCPRM)
  (f) Keep relay (%MK, %SK)
  (g) Data Register Table (%R)
(3) Specifying PMC setting data (SETING)
  (h) General setting data
  (i) Setting data related to editing and debugging
  (j) Online monitor parameter
(4) Writing, reading, and collating sequence programs and PMC parameters (I/O)

6.2 SOFT KEY-BASED PMC MENU SELECTION PROCEDURE

After the <SYSTEM> function key on the LCD/MDI is held down, pressing the [PMC] soft key displays the following PMC basic menu.

6.2.1 PMC Basic Menu



(1) Title line

This line displays the title of each PMC system screen.
It also displays the status of the PMC system at the right-hand end.

RUN STOP . . . Whether the sequence program is running is indicated. [RUN] means that the sequence program is running. [STOP] means that the sequence program is not running.

(2) Key-in buffer

This area displays the data that was typed in.

(3) Soft key

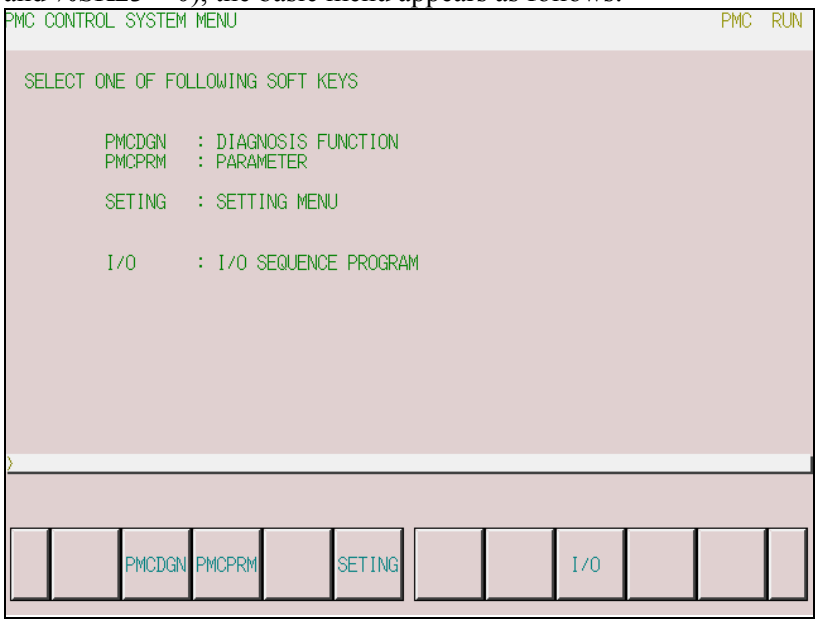
The soft key field consists of a soft key at both ends and 10 soft keys or 5 soft keys in between. The left-end key has the following meaning:

[<] Return key .Pressing this key returns you to the previous screen.

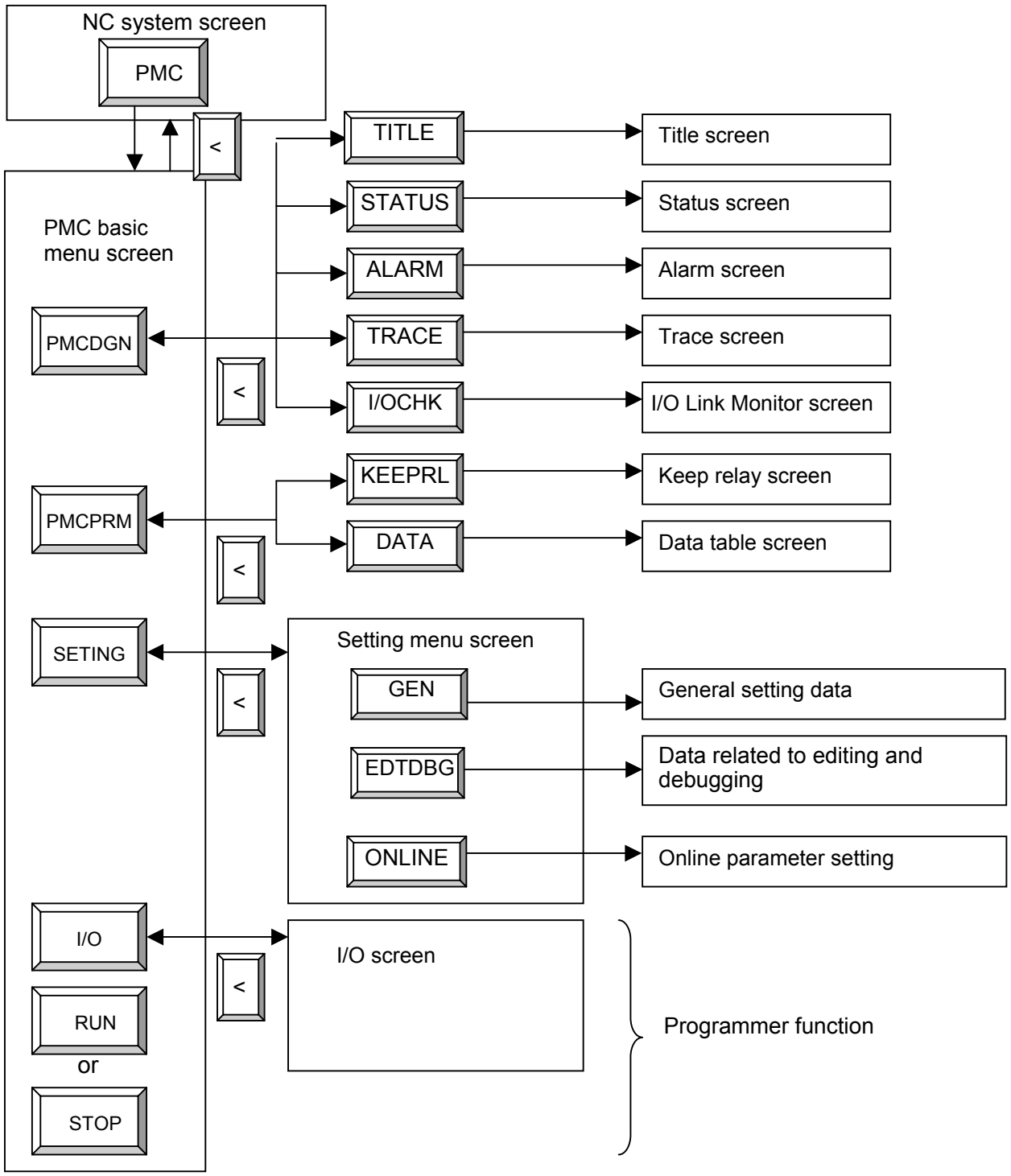
(4) Alarm line

ALM . . . . . This character string appears if a PMC alarm has occurred.

If the built-in debug function is disabled (%SK2 = 0, %SK19 = 0 and %SK23 = 0), the basic menu appears as follows:



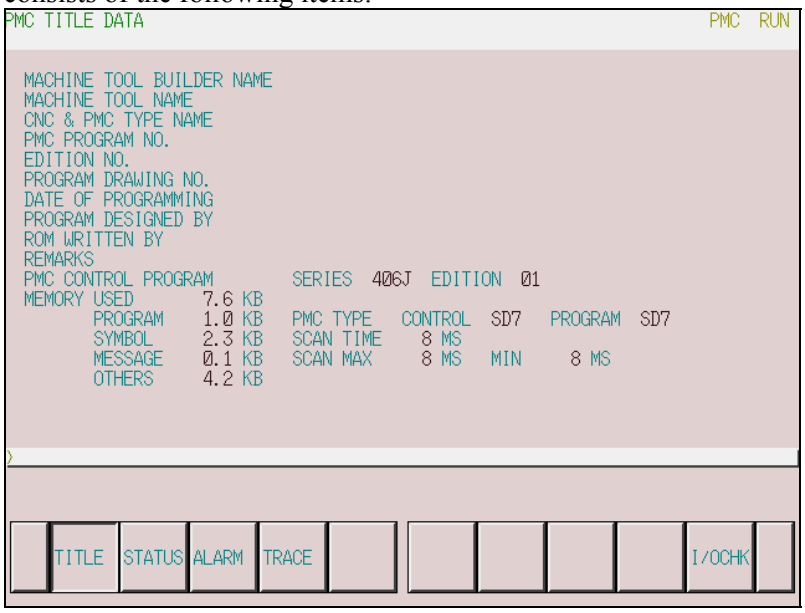
6.2.2 PMC Screen Transition and Related Soft Keys



6.3 DISPLAYING PMC INPUT/ OUTPUT SIGNALS AND INTERNAL RELAY (PMCDGN)

6.3.1 Title Data Display (TITLE)

The title data corresponds to the title of a sequence program. It consists of the following items:



- Machine tool builder name (32 characters)
- Machine name (32 characters)
- CNC/PMC type (32 characters)
- Sequence program number (4 characters)
- Edition (2 characters)
- Sequence program drawing (32 characters)
- Sequence program creation date (16 characters)
- Sequence program creator name (32 characters)
- ROM writer operator name (32 characters)
- Comment (32 characters)

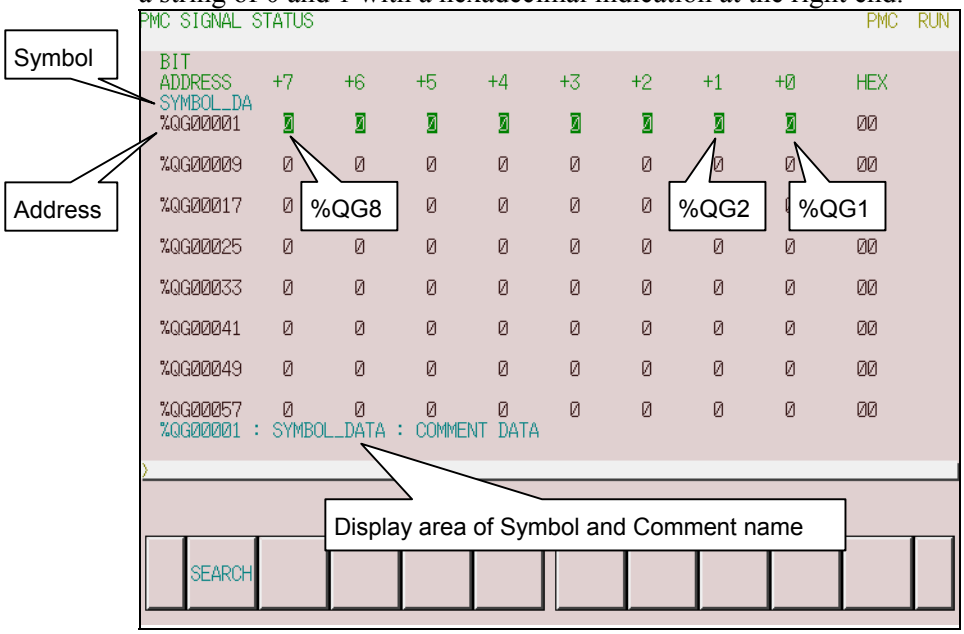
In addition, the following data is displayed:

- PMC basic software series and edition
- Amount of memory occupied by each set of sequence data
- PMC basic software type and sequence program PMC type
- Current, maximum, and minimum execution time of the ladder program

On the programming FANUC LADDER-IIIC, these title items can be edited in property inspector by selecting PMC target node.

6.3.2 Signal Status Display (STATUS)

This screen displays the contents at all the addresses (%QG, %IF, %Q, %I, %MA, %SA, %M, %ST, %S, %ME, %MK, %SK, %R, %UF and %UI) specified in programs. Each content display is a string of 0 and 1 with a hexadecimal indication at the right end.



Operating procedure

(1) Press the [STATUS] soft key. The screen shown above appears.
(2) Specify without % of the desired address, then press the [SEARCH] soft key. Example %QG1 -> QG1
(3) A sequence of data starting at the specified address is displayed as a bit pattern.
(4) To specify another address for display, press a cursor key, page key, or the [SEARCH] soft key.
(5) When you want to change the status of the signals, press [FORCE] soft key. So, the forced I/O screen appears.

NOTE
The [FORCE] soft key is displayed only when the following conditions are satisfied.
In the GENERAL screen of PMC setting:
Set “RAM WRITE ENABLE” to “YES”.
Or
Set “PROGRAMMER ENABLE” to “YES”.

(6) The signals of %I and %Q addresses which are set into overridden mode are marked with ">" on the left in the forced I/O screen. That shows the setting of override signals

NOTE

Names given to variables on the programming FANUC LADDER-IIIC can be displayed in symbol field on this screen.
Strings entered on the programmer are translated into symbols when compiling according to the following rules.

1. In the string of the bit symbol that can be displayed, there is a limitation of maximum 6 characters.
2. If the symbol (uint type etc.) except for the bit symbol is displayed, there is limitation of maximum 9 characters.
3. The limitation to input SYMBOL name is as the following.
   - The small letter cannot be inputted. Input the capital.
   - The '_' can not be input by key input. Input the '-'.
   - The '$' can not be input by key input. Input the '&'.
4. When the symbol name is searched, the small letter isn't distinguished from the capital letter.

Forced Input/ Output Function

- Overview

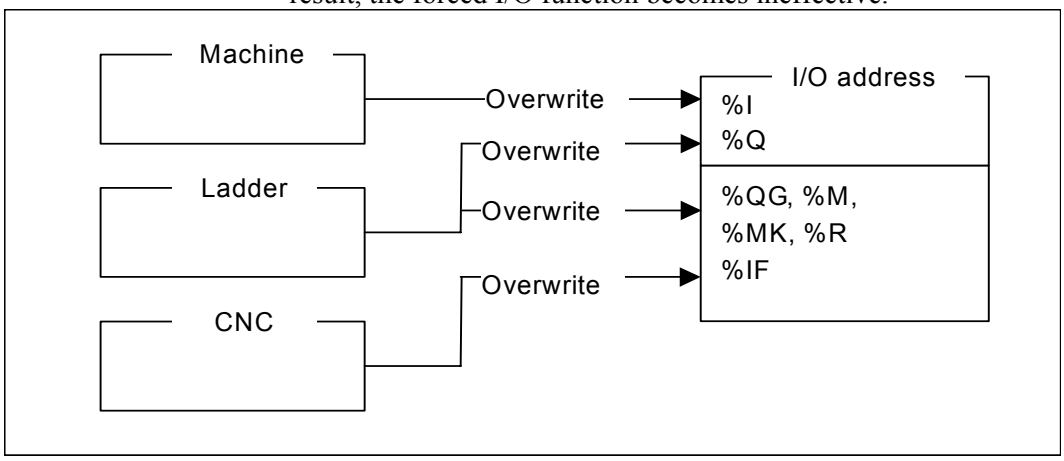
This is the function to change the status of the signals at arbitrary PMC address forcibly. If the input "%I" is forced with this function, the sequence program can be debugged without the machine. If the output "%Q" is forced, the wiring for the machine can be checked without the sequence program.

- Input mode

This function has two kinds of input modes. These modes can be used properly according to the purpose.

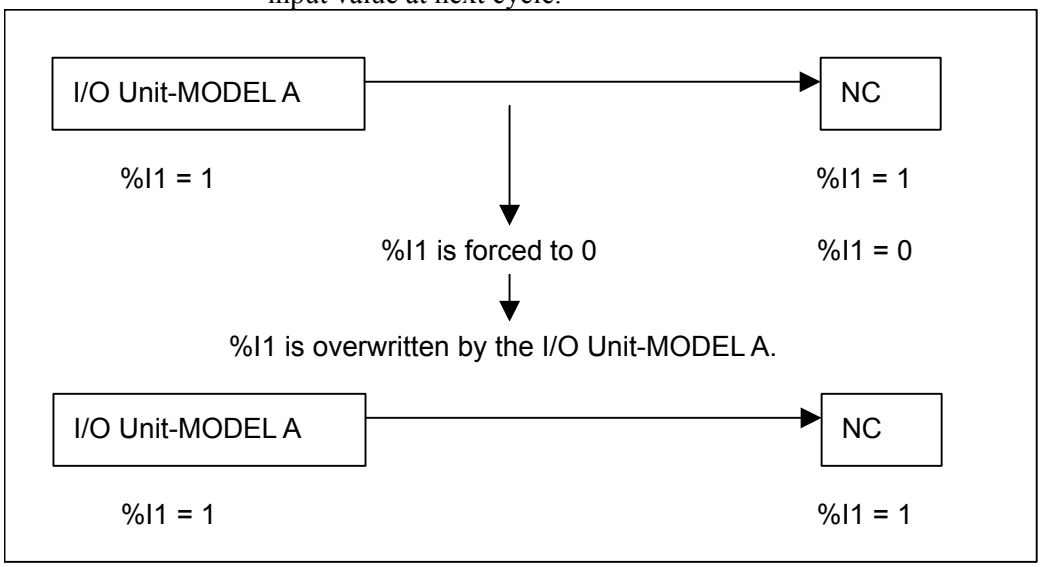
(1) Forcing mode

This mode is applied to all of PMC addresses. However, the signals changed by this function are overwritten when the sequence program or operator input in the same address. As a result, the forced I/O function becomes ineffective.



Ex1: I/O Unit-MODEL A is connected with %I1. And %I1 is forced to be changed.

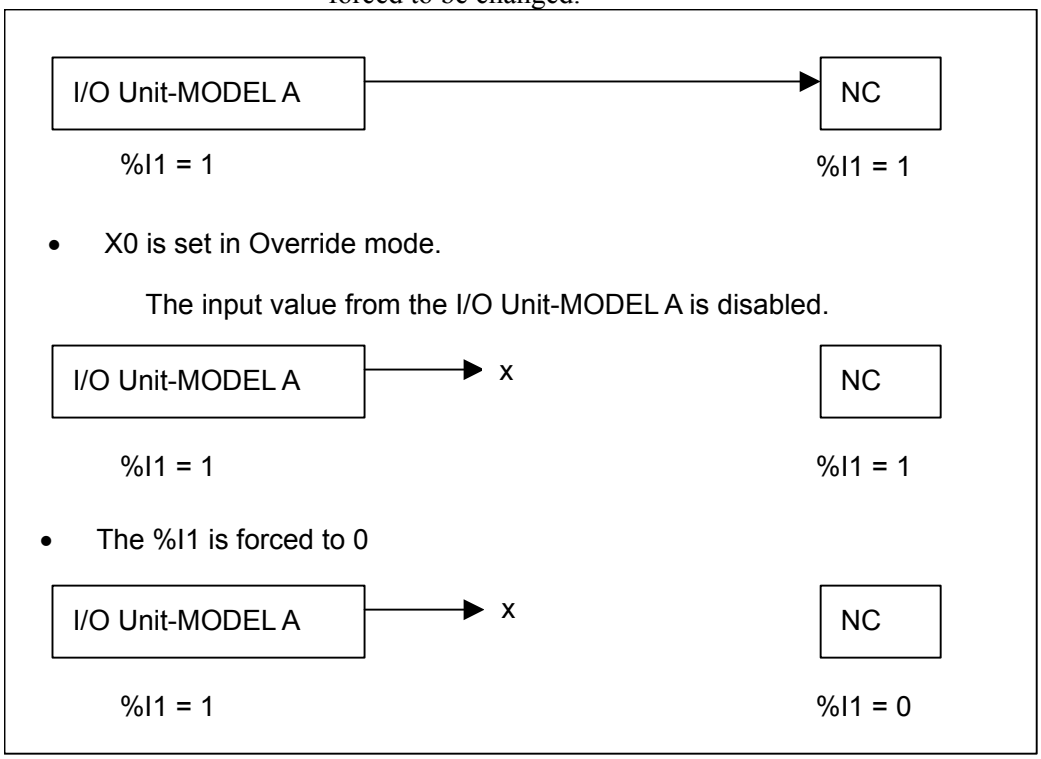
The input value from a I/O Unit-MODEL A is periodically transferred to %I1. Therefore, even if you change the value of %I1 forcedly, the I/O Unit-MODEL A would overwrite the input value at next cycle.



The cyclic transfer might be performed to the addresses at which any I/O device is not assigned. So, you should use the forced I/O function in the forcing mode without I/O devices when debugging any %I address. As for the debug of the ladder program with I/O devices, use the forced I/O function in Override mode explained in next paragraph.

(2) Override Mode
In this mode, the ladder program and the I/O devices does not overwrite the signals which are forced to be changed. You can specify the override mode into arbitrary %I/%Q addresses. The %I/%Q addresses which is not set the override mode is in the forcing mode.

Ex) I/O Unit-MODEL A is connected with %I1. And %I1 is forced to be changed.



Therefore, the forced I/O function for %I addresses in the Override mode can be used to debug the ladder program with some I/O devices. If you set the override mode into %Q addresses, The value forced to be changed is output to the I/O devices.

NOTE

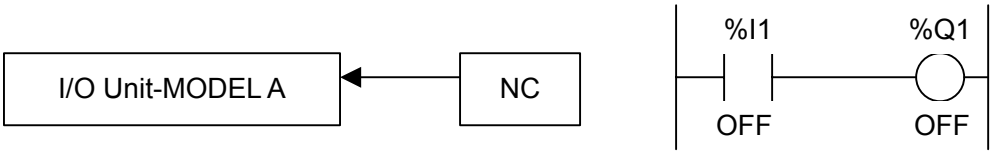
1 In the Override mode, the refresh cycle of the I/O signal is 8msec, which is synchronized with the 1st level of the ladder program. Usually, the refresh cycle of the I/O link is 2msec. So, the timing of the signals for I/O devices is delayed. Pay attention the change of work of the ladder program which depends on the timing of the I/O devices.

2 The processing time of the 2nd level of the ladder program might get longer in the override mode.

3 The relay coils which are set in the override mode are displayed as the processing result of the ladder program in the ladder monitor screen. The output value for the I/O devices is forced to be changed. Pay attention that the values on the ladder monitor screen does not correspond with the output values for the I/O devices.

Ex.) : I/O Unit-MODEL A is connected with %Q1. And %Q1 is forced to be changed.

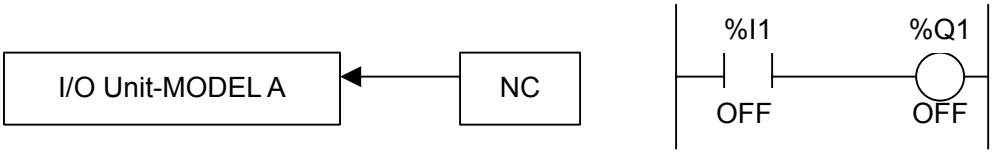
Usually, the display values on the ladder screen correspond with the output values for the I/O Unit-MODEL A as follows.



%Q1=OFF

Ladder Monitor

If you set “1” into the %Q1 in the override mode, the value is output to the I/O UNIT-MODEL A as follows.



%Q1=ON

Ladder Monitor

⚠ **WARNING**

When you change the signals by the forcing I/O function, you have to pay special attention. Inappropriate use of this function may cause unexpected reaction of machine. You have to make it sure that nobody is near the machine when you use this function.

Operations to Enable the Forced I/O Function

The following setting is necessary to enable each mode.

- Setting of forcing mode
  In the GENERAL screen of PMC setting:
  Set “RAM WRITE ENABLE” to “YES”.
  Or
  Set “PROGRAMMER ENABLE” to “YES”.

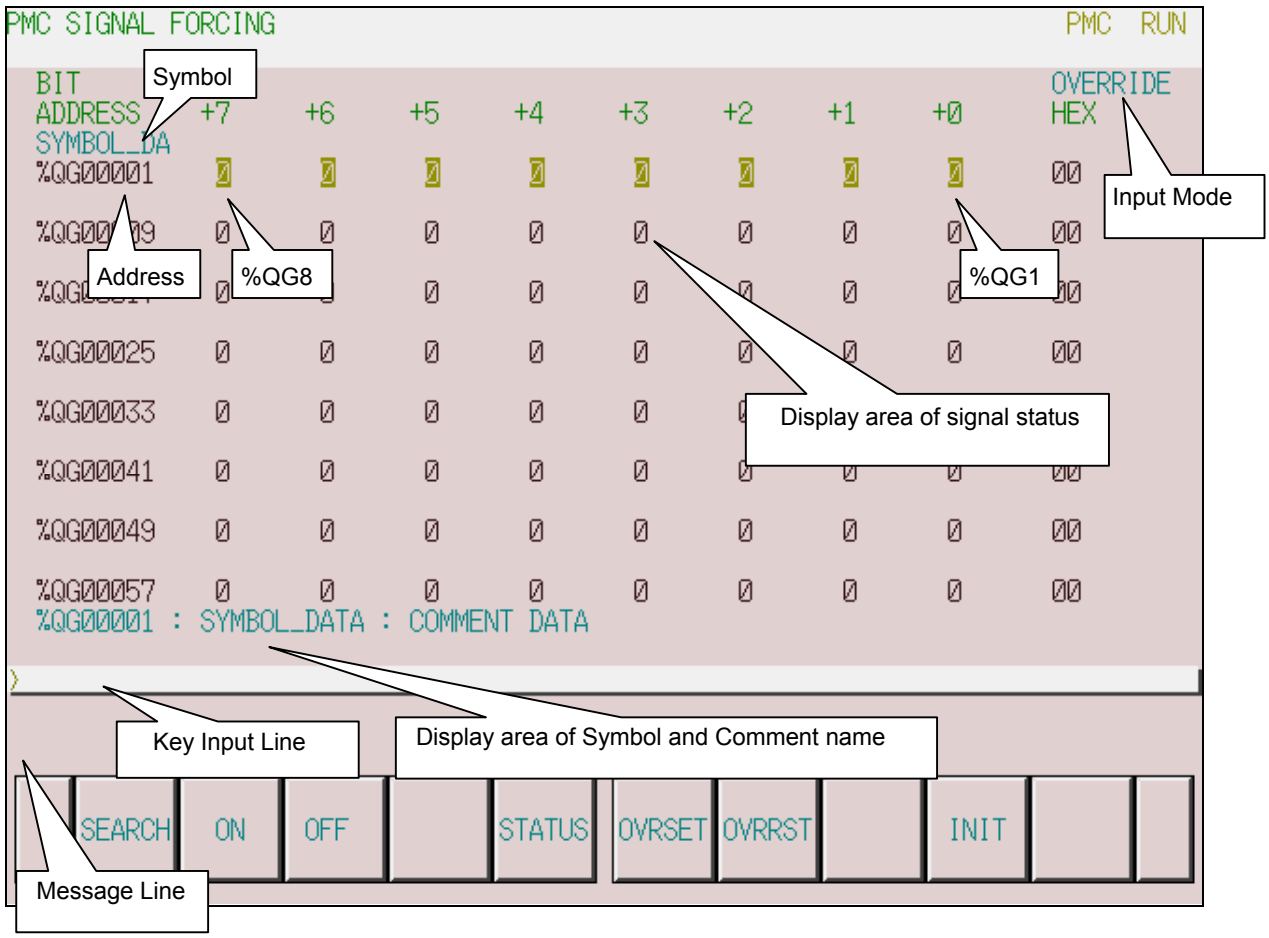
- Setting of override mode
  In the GENERAL screen of PMC setting:
  Set “RAM WRITE ENABLE” to “YES”.
  Or
  Set “PROGRAMMER ENABLE” to “YES”.
  And, In the EDIT/DEUG screen of PMC setting:
  Set “OVERRIDE ENABLE” to “YES”.
  Then, turn on the power supply again.

NOTE
1 Please disable the forced I/O function when the machine is shipped.
2 The setting “OVERRIDE ENABLE” is set to “NO” automatically and the override function becomes disabled when the setting "PROGRAM ENABLE = NO", “RAM WRITE ENABLE = NO” and turning off the power supply.
3 The settings of the override mode are not kept after turning off the power supply. All bits of %I and %Q are not in the override mode just after turning on the power supply.

Forced Input/ Output Screen

In this screen, you can change the value of arbitrary address forcibly. You can use the two input modes, i.e. the forcing mode and the override mode. To display this screen, press the soft key [FORCE] in the signal status screen. The following operations are available in this screen.

[SEARCH] Search of address
[ON] Signal on
[OFF] Signal off
[STATUS] Display of signal status screen
[OVR SET] Setting of override signal (Effective only when override mode)
[OVRRST] Release from override signal (Effective only when override mode)
[INIT] Release form all override signal (Effective only when override mode)



(1) When the override mode, The string ”OVERRIDE" is displayed above the screen.

(2) The status of each signal is displayed in the area of the signal status.

(3) Each bit of %I/%Q which is set into the override is displayed on the screen as follows.

%I address:

(Input signal from the I/O devices)->(Input signal to the ladder)

The hexadecimal numbers on the right of the screen is the overridden input signals to ladder.

ADDRESS	7	6	5	4	3	2	1	0	HEX
%I00001	0	0›1	0	1	1›0	0	0›0	0	50

%Q address:

(Output signal from the ladder)->(Output signal to the I/O devices)

The hexadecimal numbers on the right of the screen is the overridden output signals from the ladder.

ADDRESS	7	6	5	4	3	2	1	0	HEX
%Q00001	0	0›0	0	0›1	0	0›1	0	0	14

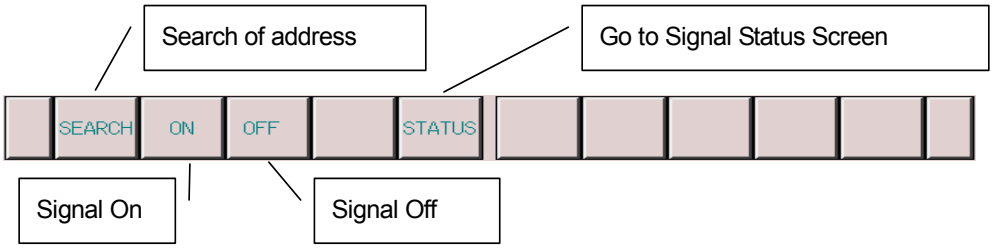
(4) In the message line, some system messages like error messages are displayed.

Operations

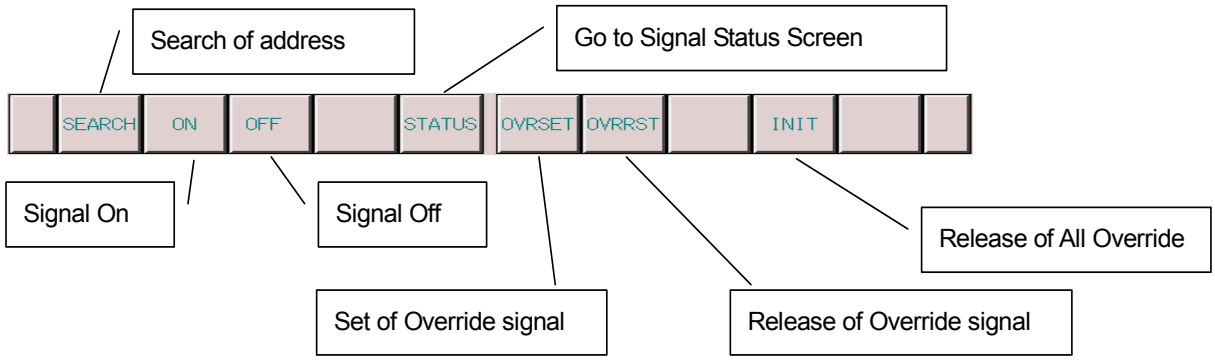
(1) Operations by the soft keys

The soft-keys in this screen

Forcing Mode



Override Mode



[SEARCH] Search of address
This searches the bit address or the word address to display on the screen.

[ON] Signal On
The signal on the cursor is sets into on.

[OFF] Signal Off
The signal on the cursor is sets into off.

[STATUS] Return to the signal status screen
This returns the signal status screen.

[OVRSET] Setting or Override signal
This sets the %I/%Q address on the cursor into the override signal.
As for the override signal, the status of the override signal is set the status of the signal just before pressing the soft key [OVRSET].
The bit set the override is added the ">" display on the screen. The actual signals before overridden are also displayed.
%I address:
(Actual input signal from the I/O devices)-> (Overridden Input signal to the ladder)

ADDRESS	7	6	5	4	3	2	1	0	HEX
%I00001	0	0>1	0	1	1>0	0	0>0	0	50

%Q address:
(Actual output signal from the ladder)->(Overridden output signal to the I/O devices)

ADDRESS	7	6	5	4	3	2	1	0	HEX
%Q00001	0	0›0	0	0›1	0	0›1	0	0	14

[OVRRST] Release of override signal
This releases the setting of override signal about the %I/%Q address on the cursor.
The status of signal returns to the status before overridden by the operation.

[INIT] Release of all the overridden signals
These releases the setting of all override signals in %I/%Q addresses.

6.3.3 Alarm Screen (ALARM)

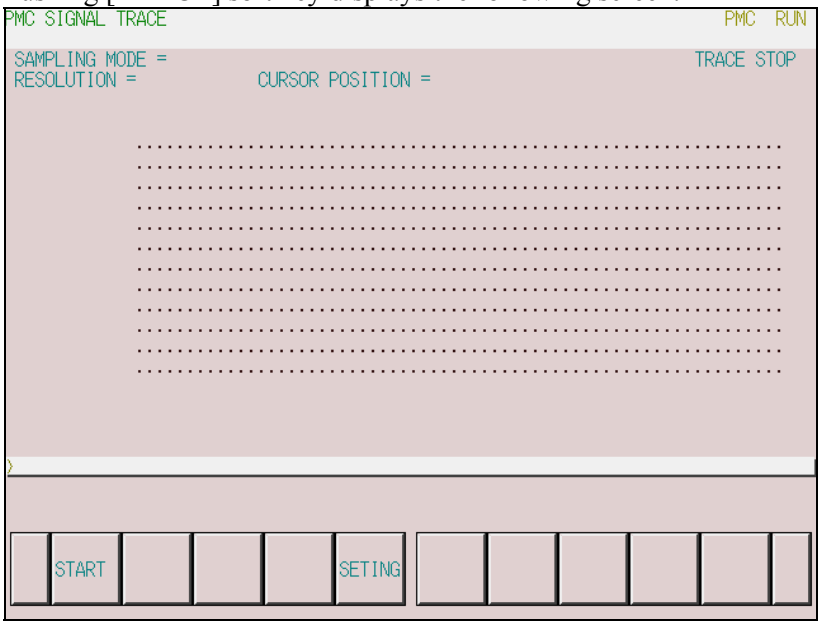
If an alarm condition occurs in the PMC, pressing the [PMC] soft key from the NC system displays the following alarm message instead of the PMC basic menu. The soft keys displayed on this screen remain the same as on the PMC basic module screen. In addition, character string “ALM” appears.
If the alarm condition is fatal, no sequence program will be executed.

For an explanation of the alarm messages displayed on this screen, see APPENDIX C, “ALARM MESSAGE LIST”.

6.3.4 Trace Screen (TRACE)

Overview

On the trace screen, you can trace specified signals. The result of the trace is displayed as the time chart of signals.
There are two tracing modes. One is the “Time Cycle” mode that samples the state of the signals at every specified cycle time. The other is the “Signal Transition” mode that samples the status of the signals when the signals that are watched at every specified time are changed.
Pushing [TRACE] soft key displays the following screen.

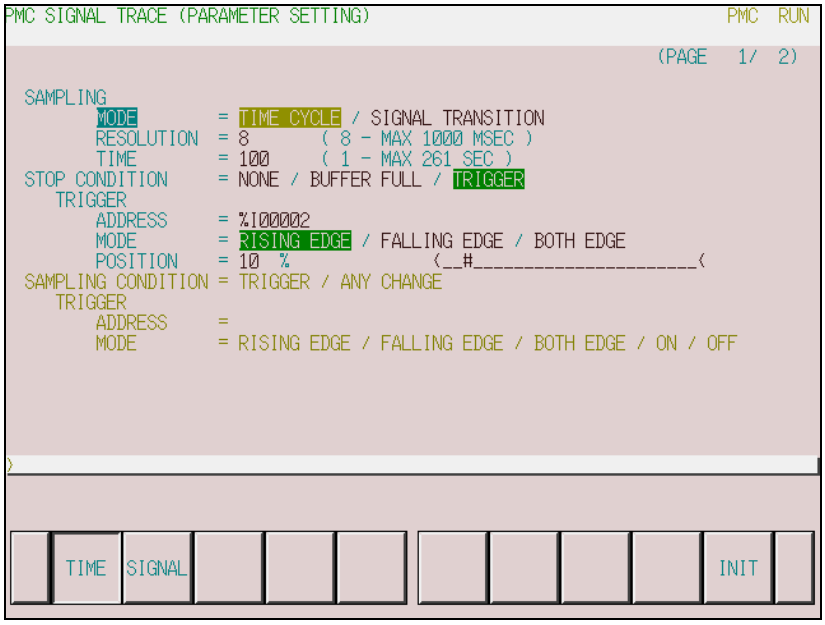


To execute tracing, setting of the trace parameter is necessary. Pushing the [SETING] soft key displays “Parameter Setting” screen. The trace function is able to run automatically by the setting on “PMC Setting” screen when the power is switched on. In this case, setting of the trace parameter is necessary in advance.

NOTE
As for the setting of automatic running for the trace function, please refer to “Screen for Displaying General Settings (GENERAL)” in Subsection 6.4.4.

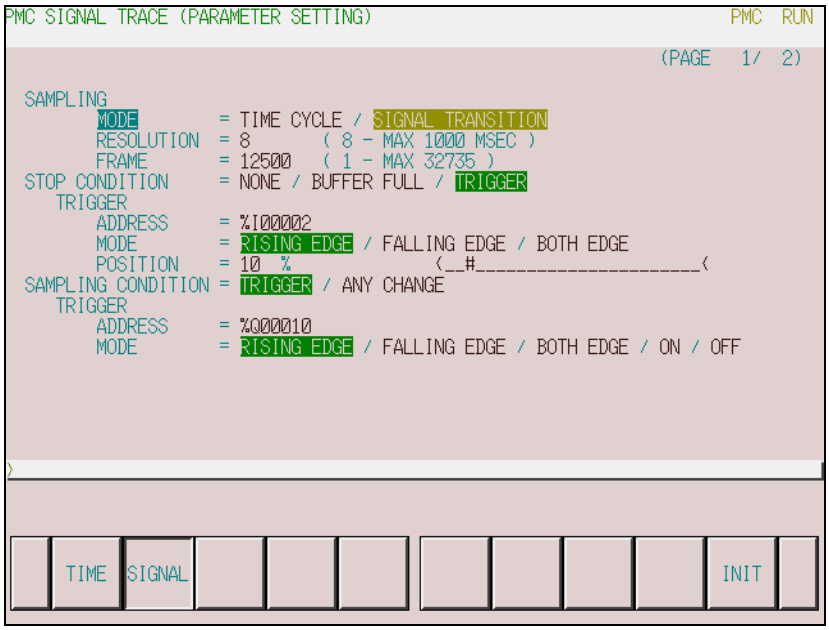
Setting of Trace Parameter

Pushing the [SETING] soft key displays the “Parameter Setting” screen.
The following is the screen example of the trace execution by “TIME CYCLE” mode.



“Parameter Setting” has two screens. The page number is displayed on up right of the screen. The page key changes these screens.
Pushing [<] soft key displays “PMC Signal Trace” screen.

The following is the screen example of the trace execution by “SIGNAL TRANSITION” mode.



• SAMPLING MODE

Determines the sampling mode. Select one by cursor key or soft key.

Soft keys display when the cursor is put on “SAMPLING MODE”.

Explanation:

TIME CYCLE	Samples at every specified cycle time.
SIGNAL TRANSITION	Samples when the signal changes.
INIT	Initializes all the settings. (This soft key is always displayed in page 1.)

• SAMPLING RESOLUTION

The resolution of sampling is inputted. The default value is the minimum resolution (8msec). The range of the value is from 8msec to 1000msec. Inputted value is rounded down to the multiple of 8msec.

• SAMPLING TIME

This parameter is displayed when “TIME CYCLE” is set on “SAMPLING MODE”. The execution time of trace is inputted. The value of “SAMPLING SESOLUTION” or the number of specified signal address changes the range of the value that is able to input. The range is displayed on the right of the edit box.

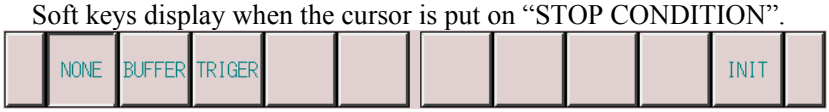
• SAMPLING FRAME

This parameter is displayed when “SIGNAL TRANSITION” is set on “SAMPLING MODE”. The number of sampling is inputted. The value of “SAMPLING SESOLUTION” or the number of specified signal addresses changes the range of the value that is able to input. The range is displayed on the right of the edit box.

• STOP CONDITION

Determines the condition to stop the trace. Select one by cursor key or soft key.

Soft keys display when the cursor is put on “STOP CONDITION”.



Explanation:

NONE	Does not stop the tracing automatically.
BUFFER FULL	Stops the tracing when the buffer becomes full.
TRIGGER	Stops the tracing by trigger

• STOP CONDITION TRIGGER ADDRESS

When "TRIGGER" is set on "STOP CONDITION", this parameter is enabled. Input signal address or symbol name as stop trigger.

Only bit address or corresponding symbols can be inputted. Byte address cannot be inputted.

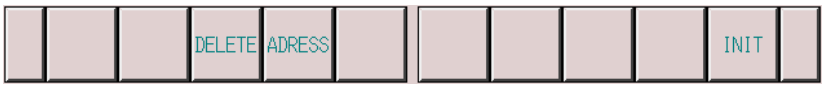
Soft keys display when the cursor is put on "STOP CONDITION TRIGGER ADDRESS".

Explanation:

DELETE Clears the value in the edit box.

SYMBOL Changes the address display to the symbol name display and changes the soft key to [ADDRESS]. After that, following soft keys are displayed.

NOTE
In the string of the symbol that can be displayed, there is a limitation of maximum 16 characters. The SYMBOL name is displayed only in case of the bit symbol.

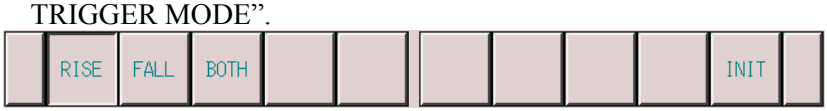


ADRESS Changes the symbol name display to the address display and changes the soft key to [SYMBOL].

• STOP CONDITION TRIGGER MODE

When "TRIGGER" is set on "STOP CONDITION", this parameter is enabled. Input trigger mode when the trace is stopped. Select one by cursor key or soft key.

Soft keys display when the cursor is put on "STOP CONDITION TRIGGER MODE".



Explanation:

RISE Stops the tracing automatically by rising up of the trigger signal.

FALL Stops the tracing automatically by falling down of the trigger signal.

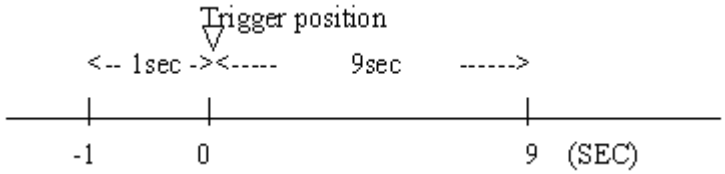
BOTH Stops the tracing automatically by rising up or falling down of the trigger signal.

• STOP CONDITION TRIGGER POSITION

When "TRIGGER" is set on "STOP CONDITION", this parameter is enabled. Input the ratio of the sampling time or number which specifies the position where specified trigger condition is on.

If you would like to examine the transitions of the signal before the trigger condition, you should set a big value in this parameter. If you would like to examine the transitions of the signal after the trigger condition, you should set a small value in this parameter.

Example: The case that sampling time is 10 second and trigger position is set as "10%".



The graph is displayed on the right of the edit box. The edge of the left hand is as 0% and the edge of the right hand is as 100%. The position indicated by the input value is displayed as a gauge.

• SAMPLING CONDITION

When "SIGNAL TRANSITION" is set on "TRACE MODE", this parameter is enabled. Select one by cursor key or soft key.

Soft keys display when the cursor is put on "SAMPLING CONDITION".

Explanation:

TRIGER Samples the status of specified signals when the specified sampling condition is on.

CHANGE Samples the status of specified signals when the signals change

The address of the signals that should be sampled is set on Parameter Setting screen page2.

• SAMPLING CONDITION TRIGGER ADDRESS

When "SIGNAL TRANSITION" is set on "TRACE MODE", and "TRIGGER" is set on "SAMPLING CONDITION", this parameter is enabled. Input signal address or symbol name as sampling trigger.

Only bit address or corresponding symbol can be inputted. Byte address cannot be inputted.

Soft keys display when the cursor is put on "SAMPLING CONDITION TRIGGER ADDRESS".

The contents of the soft keys are same as "STOP CONDITION TRIGGER ADDRESS".

• SAMPLING CONDITION TRIGGER MODE

When "SIGNAL TRANSITION" is set on "TRACE MODE", and "TRIGGER" is set on "SAMPLING CONDITION", this parameter is enabled. Input trigger mode that determines the condition of specified trigger.

Soft keys display when the cursor is put on "SAMPLING CONDITION TRIGGER MODE".

Explanation:

RISE Samples the status of specified signals by rising up of the trigger signal.

FALL Samples the status of specified signals by falling down of the trigger signal.

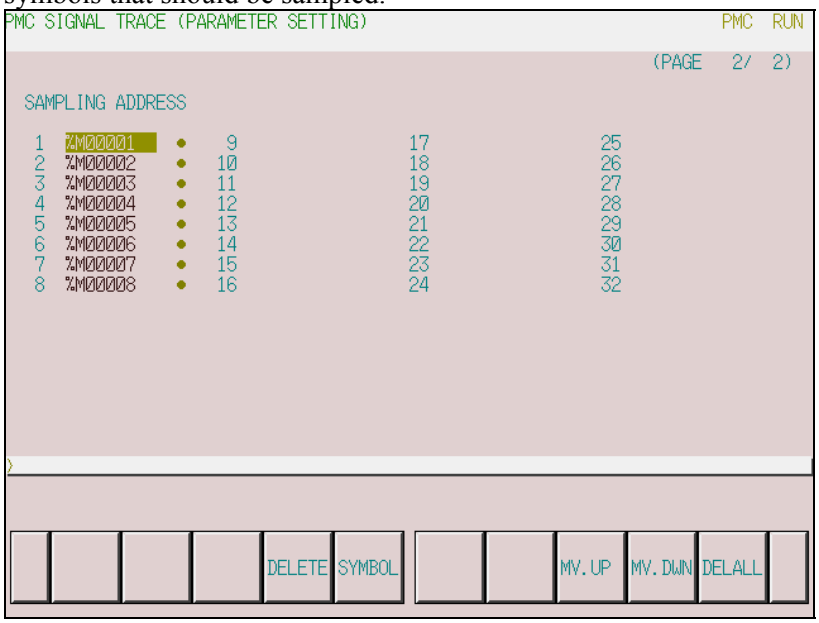
BOTH Samples the status of specified signals by rising up or falling down of the trigger signal.

ON Samples the status of specified signals during the trigger signal is on.

OFF Samples the status of specified signals during the trigger signal is off.

• SAMPLING ADDRESS

In page 2 of Parameter Setting screen, you can set the addresses or symbols that should be sampled.



Move the cursor into edit box and input PMC signal address or symbol.

In case of inputting discrete bit addresses, any bit address can be inputted. Moreover, when you input byte address, all bits of the address (bit0-bit7) are set automatically.

Maximum 32 points of signal address can be inputted. Increasing the number of the signal address changes the capacity of "SAMPLING TIME" or "SAMPLING FRAME" in page1. If the capacity is changed, the warning message is displayed

Example of warning message:
In case of “TIME CYCLE” mode
“SAMPLING TIME IS REDUCED TO n SEC.”
In case of “SIGNAL TRANSITION” mode
“SAMPLING FRAME IS REDUCED TO n COUNT.”

The “n” means the maximum value that is able to input.

Explanation of the soft keys:

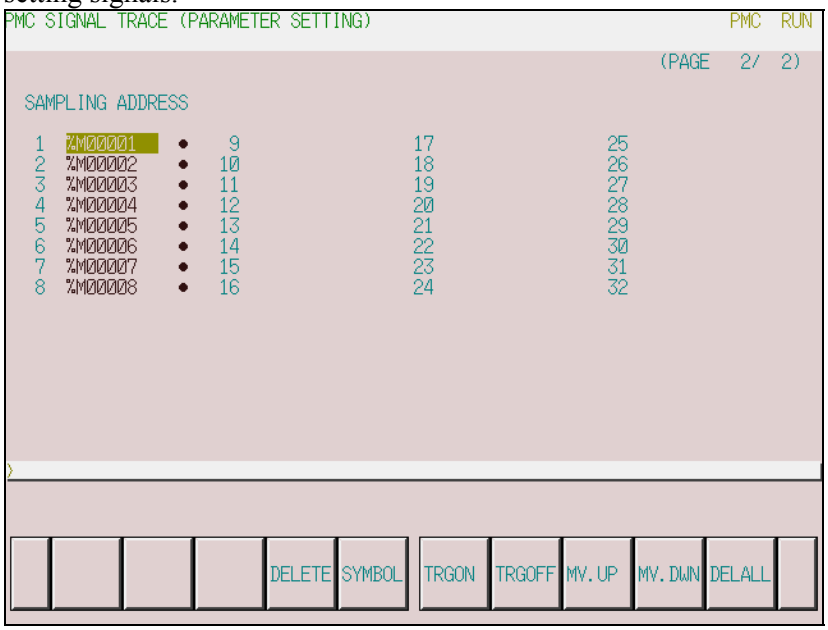
DELETE	Clears the value of the edit box on the cursor.
SYMBOL	Changes the address display to the symbol display. However, display of the address that is not defined the symbol does not change. This soft key also changes to “ADDRESS”. The following soft keys are displayed.
MV.UP	Exchanges the signal indicated the cursor for the signal above one line.
MV.DWN	Exchanges the signal indicated the cursor for the signal below one line.
DELALL	Clears all of the value of the edit box.

ADDRESS	Changes the symbol display into the address display and changes the soft key to “SYMBOL”.

NOTE
In the string of the symbol that can be displayed, there is a limitation of maximum 10 characters. The SYMBOL name is displayed only in case of the bit symbol.

- TRIGGER SETTING OF THE SAMPLING SIGNALS

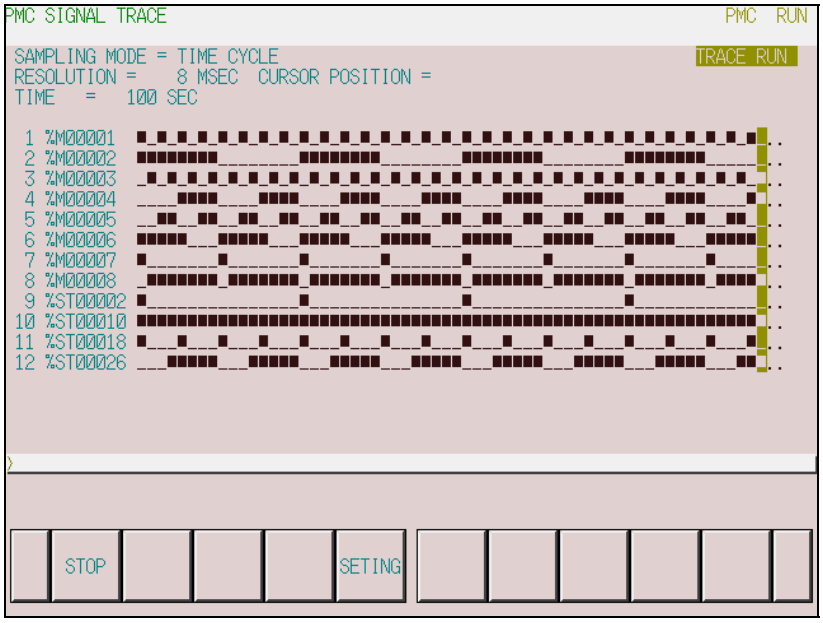
  When “SIGNAL TRANSITION” is set on “TRACE MODE” and “ANY CHANGE” is set on “SAMPLING CONDITION”, the check boxes on the right of the sampling address or symbols are displayed as follows. Check the signals that should trigger the sampling in the setting signals.



Pushing [TRIGGER ON] soft key sets the trigger on. Pushing [TRIGGER OFF] soft key sets the trigger off. The default setting is trigger on for all signals. The contents of other soft keys are same as “SAMPLING ADDRESS”.

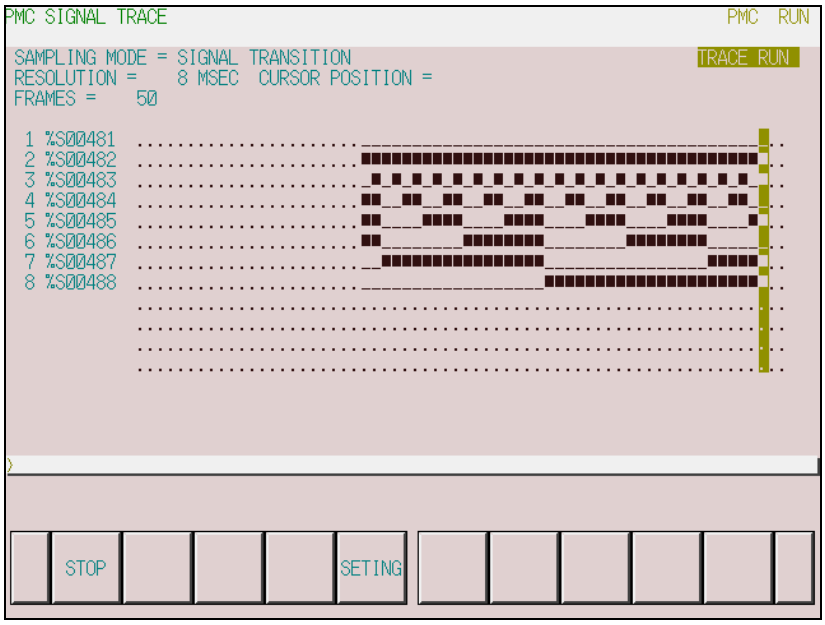
Execution of Trace

On trace screen, pushing [START] soft key starts the execution of trace after you set the trace parameter correctly. The following is the screen example of the trace execution by “TIME CYCLE” mode.



The result of trace is immediately displayed during execution of the trace. When the stop condition that is set in parameter setting screen is satisfied the execution is finished. Pushing [STOP] soft key aborts the execution.

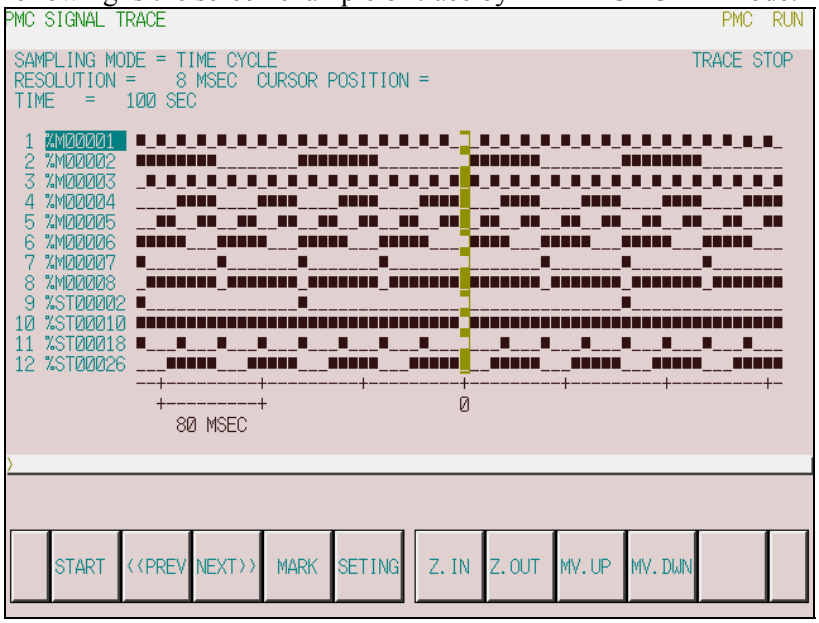
The following is the screen example of the trace execution by “SIGNAL TRANSITION” mode.



In “SIGNAL TRANSITION” mode, graphic display is not refreshed until any signal for sampling trigger changes.

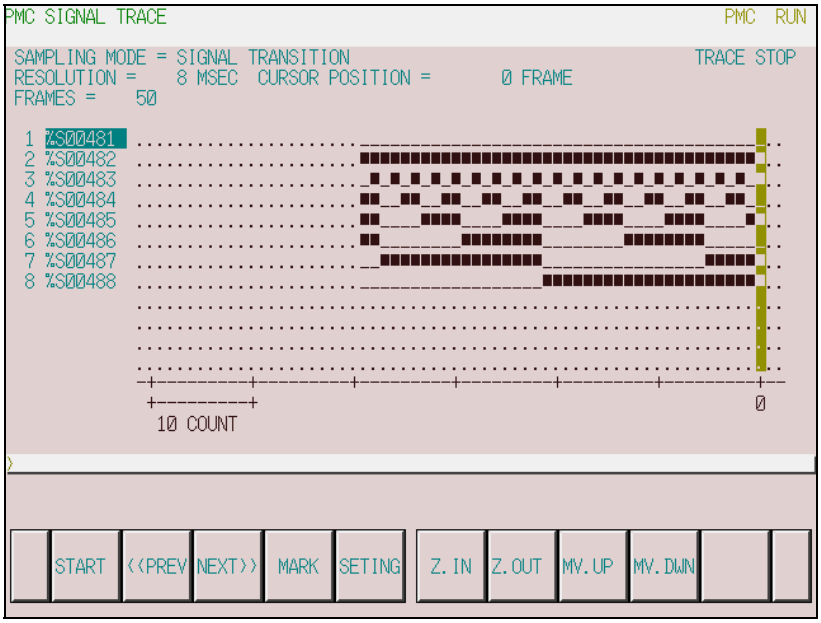
Operation after Execution of Trace

When the execution is finished, the result of trace is displayed. The following is the screen example of trace by “TIME CYCLE” mode.



The cursor indicating current position is initially displayed on the original point (0 point).
The position of the cursor is displayed in “CURSOR POSITION” in the upper of the screen. The cursor can move horizontally.

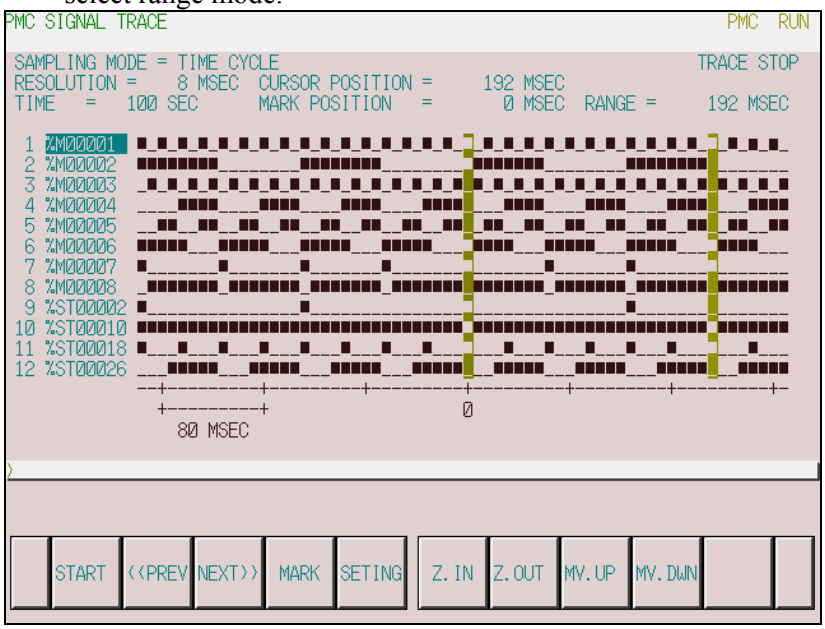
The following is the screen example of trace by “SIGNAL TRANSITION” mode.



After the execution, following operation is enabled.

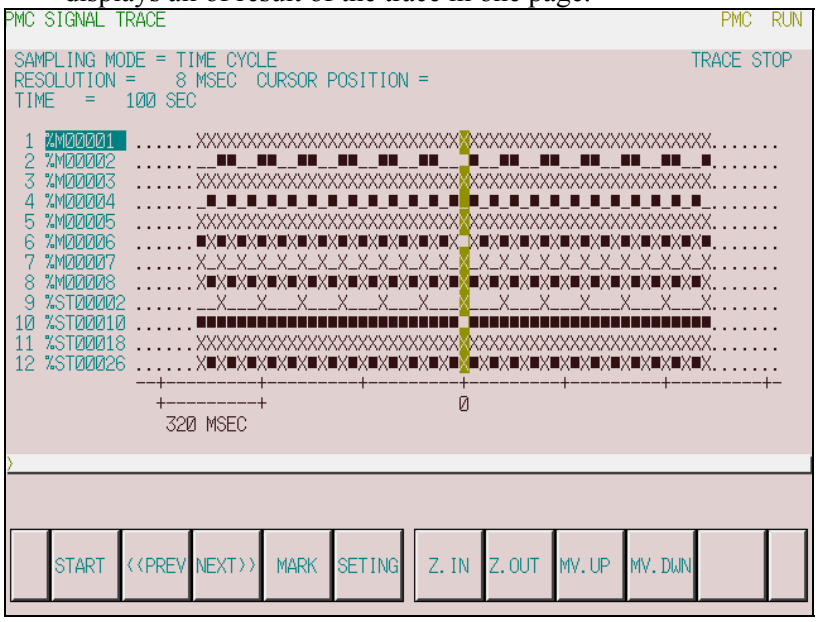
(1) SCROLL OF SCREEN
Using cursor up/down key and page up/down key enables the vertical scroll for the specified signal. Using cursor right/left key, [NEXT>>] soft key and [<<PREV] soft key enables the horizontal scroll of the graph.

(2) AUTOMATIC CALCULATION OF THE SELECTED RANGE
Pushing [MARK] soft key marks the current position and displays the mark cursor. If the mark cursor duplicates with the current position cursor, the current position cursor has priority of display. The “MARK POSITION” that shows the position of the mark cursor and “SELECT RANGE” that shows the range between the mark cursor and the current position cursor are displayed in the upper of screen. Moving the current position cursor changes these values. Pushing [MARK] again releases the select range mode.



(3) ZOOM IN/ZOOM OUT OF WAVEFORM

Pushing [Z.IN] soft key magnifies the display of chart. Pushing [Z.OUT] soft key reduces the display of chart. Pushing these soft keys also change the scale value of the graduation on the graph. When trace is just finished, the default zooming level was the most magnified level. In [Z.OUT] mode, gray box is displayed as following screen example when the transitions of signal cannot be expressed accurately enough. The limitation of [Z.OUT] displays all of result of the trace in one page.

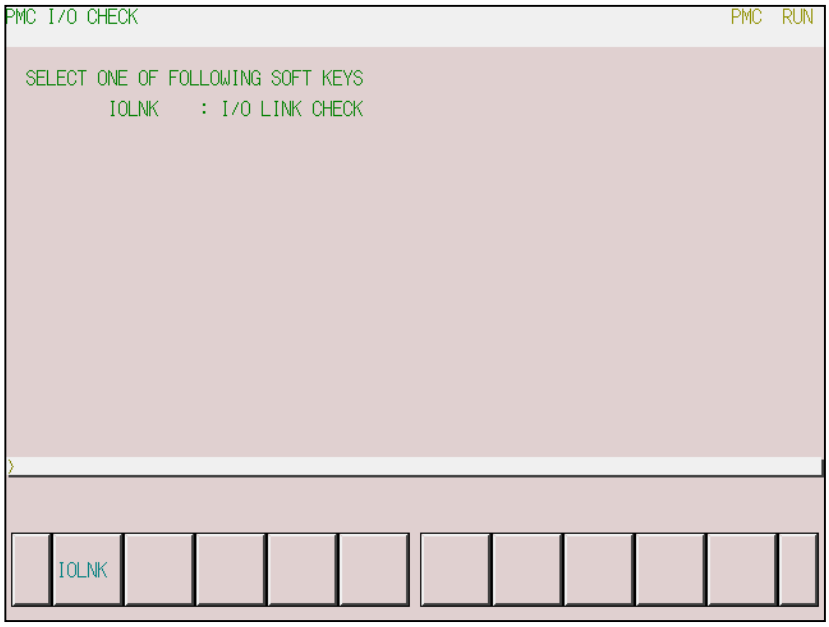


(4) EXCHANGE OF SAMPLING SIGNAL

Pushing [MV.UP] soft key exchanges the signal indicated by the signal cursor for the signal one line above. Pushing [MV.DOWN] soft key exchanges the signal indicated by the signal cursor for the signal one line below. The result of the operation is cancelled by the execution of trace or putting the power off. When you would like to preserve the order of displayed signals against the executing or powering off, please change the order on “SAMPLING ADDRESS” screen.

6.3.5 Displaying and Setting the Configuration Status of I/O Devices (I/OCHK)

Call this screen by pressing of soft key [I/OCHK] in PMCDGN.

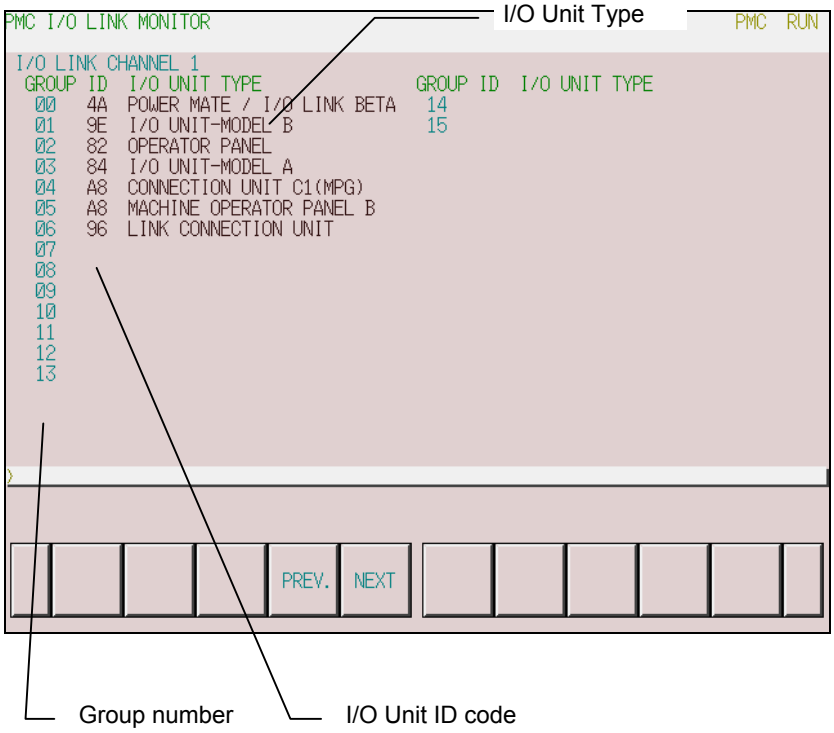


There are following sub screens under the I/O Check screen. Soft keys are displayed when each function can be used.
Please refer to the manual about requirement of the functions and detail of the sub screens.

Soft key : Name of sub screen
[IOLNK] : I/O Link Connecting Check screen
[IOLNK2] : I/O Link-II Parameter Setting screen
For details, please refer to “FANUC I/O Link-II CONNECTING MANUAL (B-62714EN).”
[PROFI] : PROFIBUS-DP Parameter Setting screen
For details, please refer to “FANUC PROFIBUS-DP Board OPERATOR’S MANUAL (B-62924EN).”
[D_NET] : DeviceNet Parameter Setting screen
For details, please refer to “FANUC DeviceNet Board OPERATOR’S MANUAL (B-63404EN).”
[FL-NET] : FL-net Parameter Setting screen
For details, please refer to “FANUC FL-net Board OPERATOR’S MANUAL (B-63434EN).”

I/O Link Connecting Check Screen (IOLNK)

The I/O Link connecting check screen displays the types and ID codes of the connected I/O devices for each group. If there is a problem of input or output signals for I/O devices, check connection of I/O Link by referring to this screen. Call this screen by pressing of soft key [IOLNK] in I/OCHK. Then check if browsed group Ids are consistent with the setting of I/O link configured under the hardware configuration node on the programmer FANUC LADDER-IIIC.



If the channel 2 is available, the [PREV.] and [NEXT] soft keys are displayed. This example screen shows that the channel 2 is available.

NOTE
I/O Link expansion option is necessary to use the channel 2 of I/O Link.

Table 6.3.5 Displayed type and true type of I/O Units

Displayed I/O Unit	ID	True I/O Unit
CONNECTION UNIT	80	Connection Unit
OPERATOR PANEL	82	Connection Unit for Operators Panel
I/O-B3	83	I/O B3
IO UNIT-MODEL B	84 86 87	I/O Unit-MODEL A
PLC SERIES 90-30	45	PLC SERIES 90-30
POWER MATE / IO LINK BETA	4A	Power Mate or I/O Link Beta
SERIES 0	50	Series 0
OPERATOR IF BOARD (MPG1)	53	Machine Operators Panel Interface
LINK CONNECTION UNIT	96	I/O Link Connecting Unit
I/O UNIT-MODEL B	9E	I/O Unit-MODEL B
R-J MATE	61	R-J Mate
CONNECTOR PANEL MODULE	A9	I/O module for connector panel
OPERATOR PANEL A1	AA	I/O module for operator's panel
OPERATOR I/F BOARD (MPG3)	6B	Operator Interface(with MPG)
LOADER I/O	AF	I/O Board for Loader
ROBOCUT DIF	B0	DIF Board for ROBOCUT
ROBOCUT MIF	B1	MIF Board for ROBOCUT
I/O CARD	B2	I/O board
ROBOSHOT I/O CARD A	B3	I/O for ROBOSHOT
LOADER I/O (MATRIX)	B4	I/O Board for Loader(Matrix)
PROCESS I/O FA	B5	Process I/O for R-J3
PROCESS IO	89	Process I/O for R-X
I/O LINK ADAPTER	8B	I/O Link adapter
ROBOT CONTROLLER	52	Controller for R-X
PLC SERIES 90	54	PLC SERIES 90
OPERATOR PANEL	95	I/O for Series 0
LASER OSCILLATOR	97	Laser Oscillator
FIXED I/O TYPE A	98	I/O for Robot Type A
FIXED I/O TYPE B	99	I/O for Robot Type B
AS-I CONVERTER	77	AS-i Converter
OPERATOER PANEL B	A8	I/O Module(for Operator Panel 48/32)
MACHINE OPERATOER PANEL A	A8	I/O Module(for Machine Operator Panel of 0 Type)
CONNECTION UNIT C1 (MPG)	A8	Connection Unit C1(with MPG)
MACHINE OPERATOER PANEL B	A8	I/O Module (for Machine Operator Panel)
I/O MODULE WITH LCD	A8	LCD display embedded I/O
UNKNOWN UNIT	-	Unsupported I/O Unit

I/O Link–II Parameter Setting Screen

In case of using the I/O Link–II function, set the following I/O Link–II parameter on this screen. Depending on the kind of I/O Link–II interface board, master/slave screen is displayed automatically.
Please refer to FANUC I/O Link–II Operator's Manual (B–62714EN) about details of I/O Link–II and each parameter.

(1) Set parameters.
Move the cursor to the parameter by using the cursor key.
Type the data and press the soft key[INPUT] or MDI key<INPUT>.
The set parameter is saved to the I/O Link–II board when the data is input.

(2) Change channel.
Change the channel by the soft key [PRV.CH],[NXT.CH]. These keys are not displayed when the single channel is used.

(3) Delete parameter.
Move the cursor to the parameter by using the cursor key.
Press the soft key[DELETE].

(4) Delete all parameters.
Press the soft key[DELALL].
Press the soft key[EXEC] to delete all parameters.
Press the soft key[CANCEL] to cancel the deletion.

(5) Change page.
This screen is composed of two pages when the 9 inch CRT is used.
Change the page by using (PAGE) key of MDI.

(6) Re–start I/O Link–II
Press the soft key [START] to re–start I/O Link–II after editing the parameter.
When the re–start is completed normally, "LINK STARTED" is displayed.
If the re–start fails, "START ERROR" is displayed. In this case, check the parameter that is set.

NOTE
The PMC address of this screen is displayed in PMC-SB7 address.
It is not displayed in PMC-SD7 address.

6.4 DISPLAYING AND SETTING THE PMC PARAMETERS (PMCPRM)

6.4.1 Overview

Pressing the [PMCPRM] soft key causes the PMCPRM screen to appear.
The PMC parameters in the permanent memory areas (%MK, %SK and % R) can be displayed and set using the CNC display.

6.4.2 Input PMC Parameters from MDI Panel

Usually, no data can be entered for PMC parameters because they are protected. The following two methods can be used to make it possible to enter data for them.

- If the sequence program is running (RUN state) (usually, this method should be used when the machine is operating.)
  i) Place the NC in MDI mode or bring it to an emergency stop.
  ii) Set "PWE" on the NC setting screen to 1 (see the following table).
  iii) Alternatively, set the program protect signal (KEY4) to 1 (only if data register tables are involved).
  iv) The parameters are released from protection; so data can be entered for them (see the following table).

	PWE	KEY4
KEEP RELAY (%MK, %SK)	O	
DATA REGISTER TABLE (%R)	O	O

: Alternative

  v) After entering data for the parameters, return "PWE" or the KEY4 signal to the previous state.
- If the sequence program can be stopped (STOP state), for example, while it is being debugged
  i) Stop the sequence program.
  ii) The parameter protection is released; so data can be entered for them.

> ⚠ **WARNING**
> If a sequence program is stopped while the machine is operating, the machine may behave unexpectedly. Before stopping the sequence program, make sure that nobody is near the machine and that the tool cannot interfere with the workpiece or machine. Incorrect operation of the machine presents an extreme risk of death or serious injury to the user. Damage the tool, workpiece, and/or the machine is also likely.

An attempt to enter data for protected parameters causes the error message “WRITE PROTECT” to be displayed. Please refer to "Screen for displaying General settings (GENERAL)" in Subsection 6.4.4.

Multiple Data Input

1 This function is effective on the screen of KEEP RELAY and DATA REGISTER TABLE.

2 The cursor is moved to the final data position of inputted data.

(1) Input method

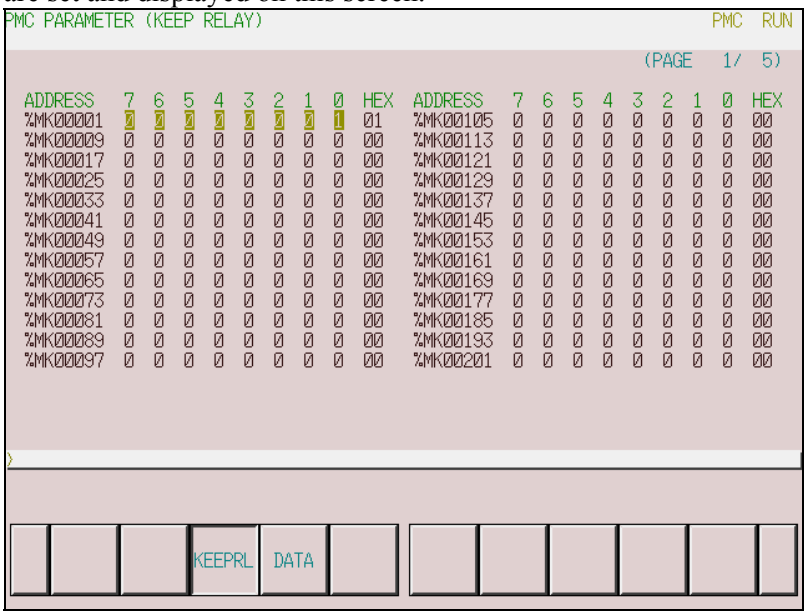
- “ ; (EOB)” is used for separating data.
  Press the INPUT key after typing “100; 200; 300”.
- “ ; =” is used for inputting the same value as preceding data.
  Press the INPUT key after typing “100; =; =; 200; =”, and it becomes “100, 100, 100, 200, 200”.
- “ ; ;” is used for skipping an input address.
  Press the INPUT key after typing “100; ; 100”.
  The second data is not inputted.

6.4.3 Setting and Display Screen

Keep Relay (KEEPRL [%MK, %SK])

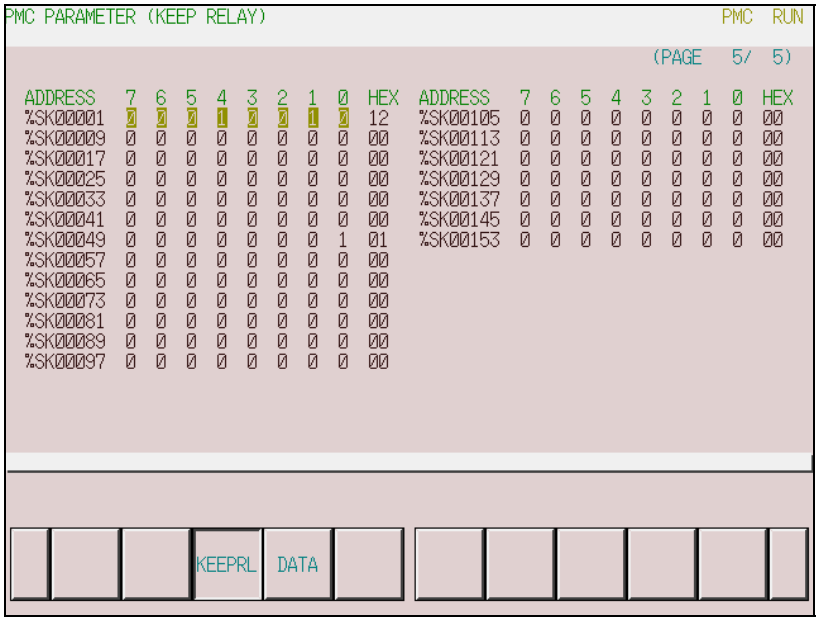
- KEEPRL [%MK]

The %MK Relays and the Data for controlling nonvolatile memory are set and displayed on this screen.



- KEEPRL [%SK]

For %SK, the system area used by the management software is divided.



NOTE
Be careful of using the %SK Relays, because they are used by PMC Management Software.
%SK is displayed by the setup of the GENERAL screen of PMC setting. Refer to “Screen for displaying General settings (GENERAL)” in Subsection 6.4.4.

%SK00002 0 : The programmer function is disabled.
#EN_PRG 1 : The programmer function is enabled.

⚠ WARNING
Set this bit to 0 before shipment from the factory. If the bit setting is left as 0, the operator may stop execution of the ladder diagram by mistake, and cause an accident.

%SK00003 0 : The sequence program is executed automatically when the power is turned on.
#RUN_NAT 1 : The sequence program is executed when the [RUN] soft key is pressed.

%SK00005 0 : Data cannot be entered at the FORCING screen.
#EN_RAM 1 : Data can be entered at the FORCING screen.

⚠ WARNING
Set this bit to 0 before shipment from the factory.

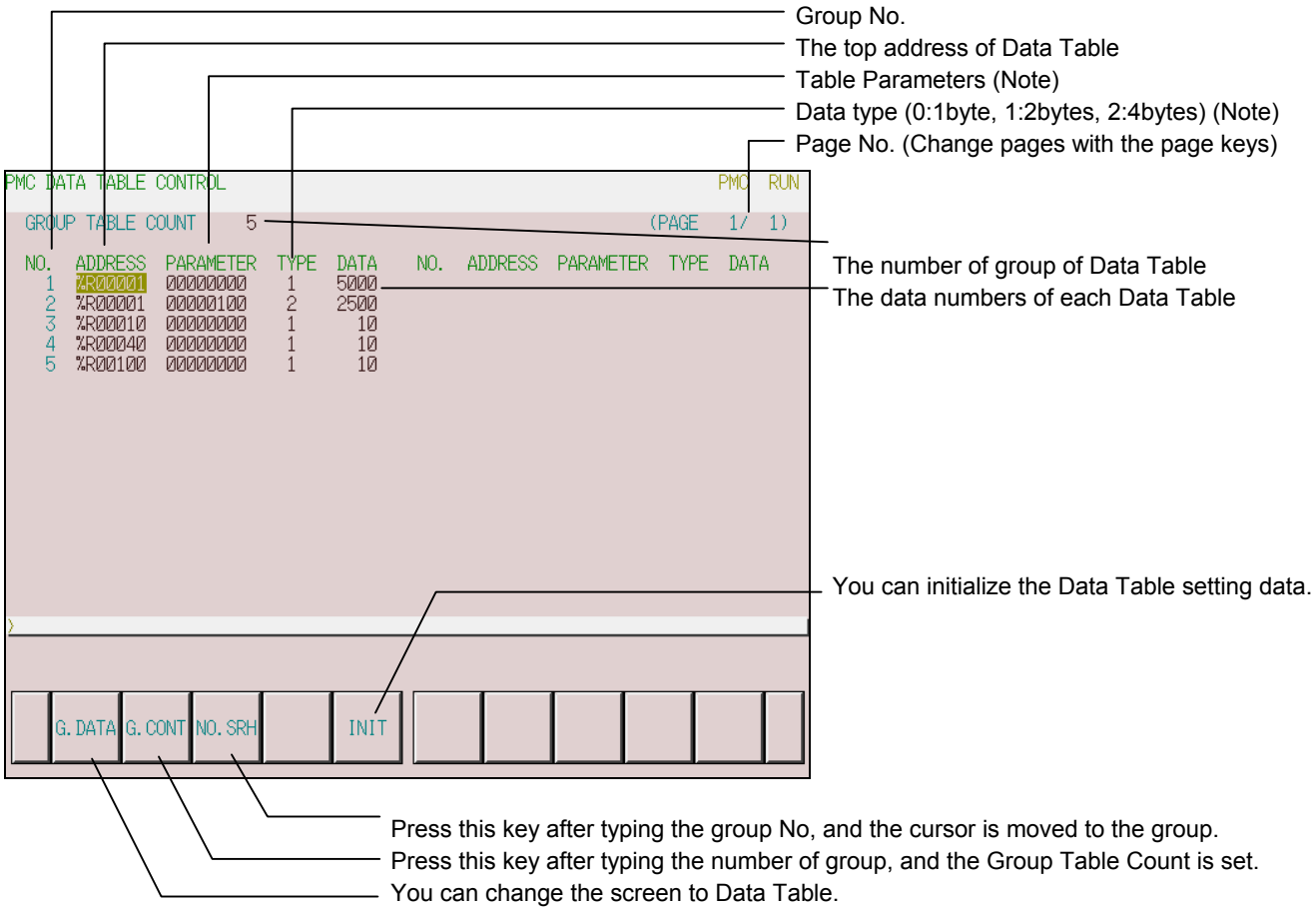
%SK00008 #NEN_DTC	0 : 1 :	The PMC parameter data register table control screen is displayed. The PMC parameter data register table control screen is not displayed.
%SK00014	0 : 1 :	The parity of the sequence program is checked. The parity of the sequence program is not checked.
%SK00015 #EN_IN	0 : 1 :	Prevents the edit (input) operation of the sequence program. Allows the edit (input) operation of the sequence program.
%SK00017 #FROM_AT	0 : 1 :	The program is not automatically written to F-ROM. After a sequence program has been updated, the program is automatically written to F-ROM.
%SK00019 #DSP_RUN	0 : 1 :	Prevents run/stop and edit (input) operation of the sequence program. Allows run/stop and edit (input) operation of the sequence program.
%SK00023 #HIDEPRM	0 : 1 :	The PMC parameter screen is displayed. The PMC parameter screen is not displayed.
%SK00024 #PRTCPRM	0 : 1 :	The PMC parameter can be changed. The PMC parameter cannot be changed.
%SK00054 #TRC_AT	0 : 1:	In the trace function, tracing starts when the [EXEC] soft key is pressed. In the trace function, tracing starts automatically when the power is turned on.

> ⚠ **CAUTION**
> Set to 0 the bit that is not used by PMC control software.

Data Register Table (DATA [%R])

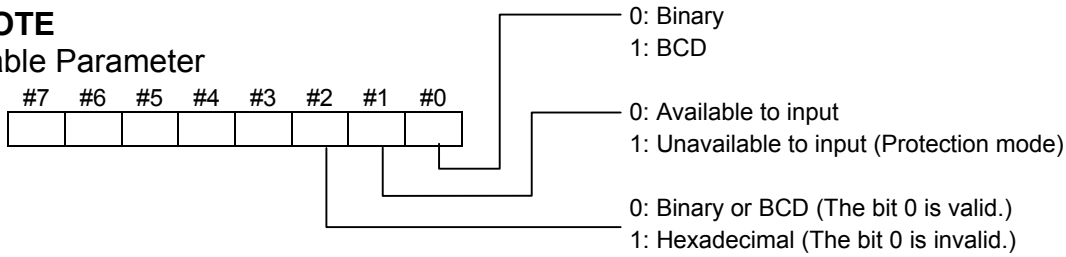
DATA REGISTER TABLE consists of 2 screens, that is, Data Register Table Controlling Data screen and Data Register Table screen.

(1) Data Register Table Controlling Data Screen
Data Register Table Controlling Data Screen for controlling Data Register Table is displayed by pressing the soft key [DATA].



NOTE

Table Parameter

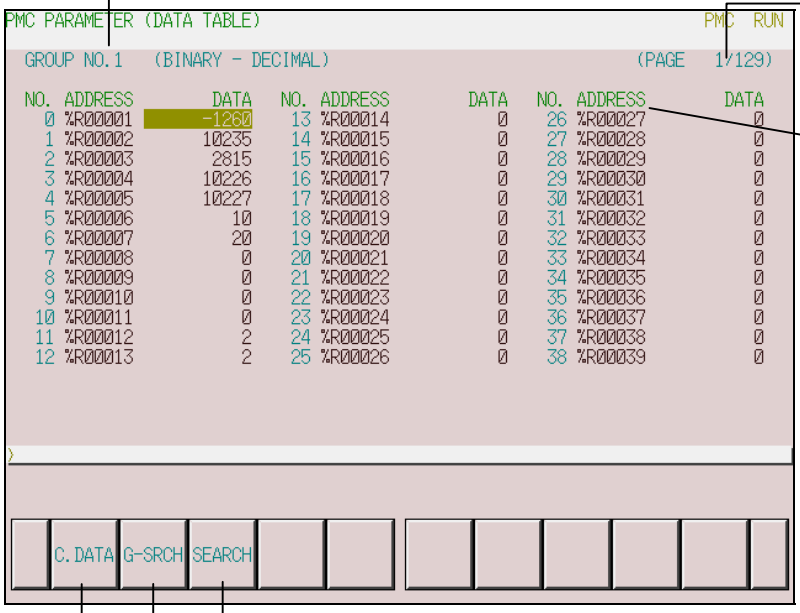


Data type

TYPE	DATA FORM	LENGTH
0	SINT, USINT, BYTE	1 byte
1	INT, UINT, WORD	2 bytes
2	DINT, UDINT, DWORD	4 bytes

(2) Data Register Table Screen

If the Data Register Table Controlling Data is set, Data Register Table Screen is displayed by pressing the soft key [G.DATA].



Group No.

Page No.
(Change pages with the page keys)

The address used by sequence program

Press this key after typing the address (ex. “%R8” %R can be omitted), and the cursor is moved to the address in the current group.

If you search the Data Table in the other group, press this key after typing the group No., and the cursor is moved to the top of the address in the specified group.

You can change the screen to Data Table Controlling Data.

6.4.4 Setting Screen (SETING)

Pushing the [SETING] soft key on the PMC basic module screen displays the following setting menu screen.

Contents of the menu.

1.GENERAL Screen for displaying general setting data.

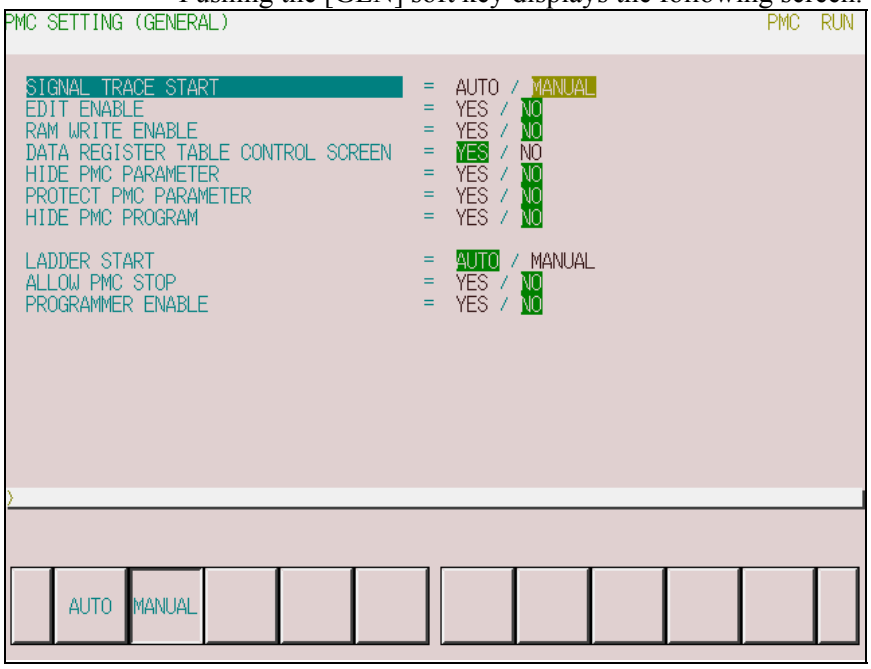
2.EDIT/DEBUG Screen for displaying setting data related to editing and debugging.

3.ONLINE Screen for displaying the communication settings for the online-function. (It is displayed by setting "PROGRAMMER ENABLE" to "YES" on the GENERAL screen.)

Pushing the soft key, each SETTING screen is displayed. A part of the setup value of this screen is stored in keep relay. It is possible to prevent changes to the settings of this setting screen, using a sequence program for writing to the keep relay.

Screen for displaying General settings (GENERAL)

Pushing the [GEN] soft key displays the following screen.



- SIGNAL TRACE START (%SK54)
  - MANUAL (0): It executes by pushing the [START] soft key on the SIGNAL TRACE screen.
  - AUTO (1): Starts the tracing automatically after the power turns on.

- EDIT ENABLE (%SK15)
  - NO (0): Prevents editing of the sequence program.
  - YES (1): Allows editing of the sequence program.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

- RAM WRITE ENABLE (%SK5)
  - NO (0): Prevents forcing function.
  - YES (1): Allows forcing function.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

• DATA REGISTER TABLE CONTROL SCREEN (%SK8)
  YES (0): Displays PMC parameter data register table control screen.
  NO (1): Does not displays PMC parameter data register table control screen.

• HIDE PMC PARAMETER (%SK23)
  NO (0): Allows PMC parameter display.
  YES (1): Prevents PMC parameter display.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

• PROTECT PMC PARAMETER (%SK24)
  NO (0): The PMC parameter can be changed.
  YES (1): The PMC parameter cannot be changed.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

• HIDE PMC PROGARAM (%SK1)
  NO (0): Allows the edit (output) operation of the sequence program.
  YES (1): Prevents the edit (output) operation of the sequence program.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

• LADDER START (%SK3)
  AUTO (0): Executes the sequence program automatically after the power turns on.
  MANUAL (1): Executes the sequence program by [RUN] soft key.

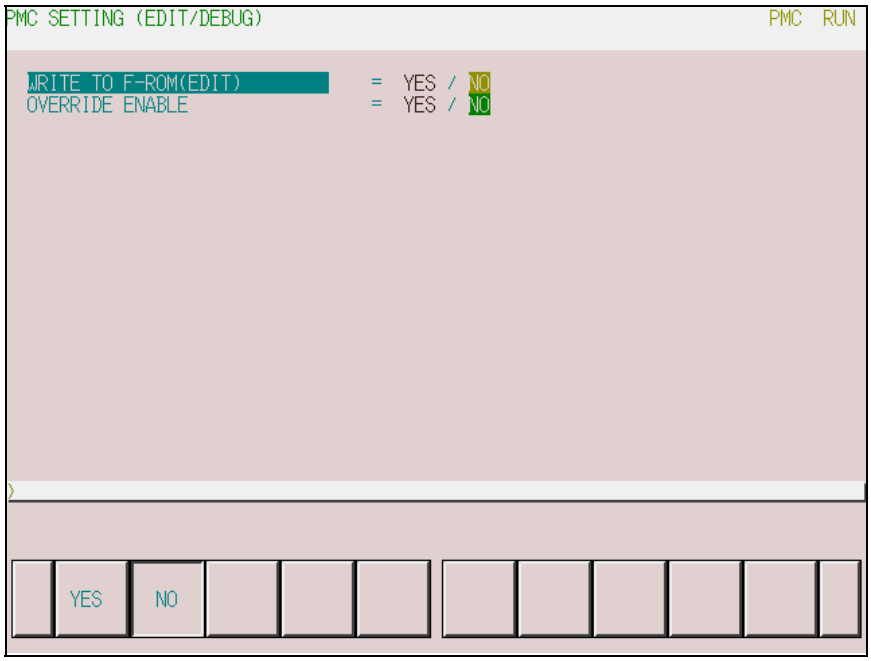
- ALLOW PMC STOP (%SK19)
  NO (0): Prevents run/stop operation of the sequence program.
  YES (1): Allows run/stop operation of the sequence program.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

- PROGRAMMER ENABLE (%SK2)
  NO (0): Disables embedded programmer function.
  YES (1): Enables embedded programmer function.

NOTE
This setting effects some PMC functions.
Please refer to "Programmer protection function" in Subsection 6.4.4.

Screen for Displaying the Setting Data Related to Editing and Debugging (EDTDBG)



(%SK17)
(%SK49)

- WRITE TO FROM (EDIT) (SK17)
  NO (0): The sequence program is not automatically written to F-ROM after reading in I/O screen.
  YES (1): The sequence program is automatically written to F-ROM after reading in I/O screen.

- OVERRIDE ENABLE (SK49)
  NO (0): The override mode becomes disabled.
  YES (1): The override mode becomes enabled.
  This screen is displayed in the case of "PROGRAMMERENABLE=YES" or "RAM WRITE ENABLE=YES".

> ⚠ **CAUTION**
> This parameter is effective when turning on the power supply after setting. Turn on the power supply if setting this parameter.
> Set this parameter "NO" when the machine is shipped.

Programmer Protection Function

> ⚠ **CAUTION**
> This section contains important information for developers of application system controlled by PMC. Improperly implemented application system may increase possibility of defects in its safety. Careful examinations and considerations on using and implementing with the functions explained especially in this section are strongly required.

PMC system provides various embedded programmer functions such as edit, diagnosis and debugging which help the programming and debugging of sequence program. To use these functions which may even disable safety mechanism realized by sequence program, it is required that the operator of these functions should be an expert who fully understands the sequence program and the operation of PMC. It is also strongly recommended to the developer of machine that these functions should be protected from careless use by ordinary operators after the machine is shipped into the field. Furthermore, if these functions partly need to be used in the field for any purpose such as the maintenance or adjustment, the developer of the machine should implement any means to enable these functions after forcing the machine in safe mode or should
let the operator know and strictly follow proper procedure to ensure the safety.
The setting parameters described in this section are provided for the developer of machine to be able to properly program the sequence or control the parameters for necessary conditions on which the operator is allowed to use PMC programmer functions safely by eliminating careless operation which may cause ”stopping the ladder”, ”changing sequence program” or ”changing important setting data”.
These parameters can be set on the setting screen or in some system keep relays (%SK1=1).

PROGRAMMER ENABLE (%SK2)

If you set ”PROGRAMMER ENABLE ” to ”YES”, it enables the following functions as a supervisor mode.

- Start/stop of ladder
- Forcing function
- Override function *1
- I/O screen
- Online setting screen
- Setting screen for keep relay %SK

NOTE
The override function also requires the setting of ”OVERRIDE ENABLE” in the setting parameters.

⚠ CAUTION
Set this setting to ”NO”(0) before shipment of the machine. If this setting is left as ”YES”(1), the operator may stop execution of the ladder program by mistake. If you want to protect this setting, please make a sequence that always writes 0 in this bit by your ladder.

HIDE PMC PROGRAM (%SK1)

If you set ”HIDE PMC PROGRAM” to ”YES”, it disables the following functions which have the sequence program display.

- Writing of sequence program on the I/O screen

NOTE
Even if this parameter is set to "YES", these settings are invalid if "PROGRAMMER ENABLE" is set to "YES".

EDIT ENABLE (%SK15)

If you set ”EDIT ENABLE” to ”YES”, it enables the following functions which can edit the program.

- Reading of sequence program on the I/O screen *1
- Setting screen for keep relay %SK

NOTE
These screens with stop of ladder program require below setting ”ALLOW PMC STOP”.

⚠ CAUTION
Set this setting to ”NO”(0) before shipment of the machine if you want to prohibit operator form editing the program. If you want to protect this setting, please make a sequence that always writes 0 in this bit by your ladder.

ALLOW PMC STOP (%SK19)

If you set ”ALLOW PMC STOP” to ”YES”, it enables the following functions which require stop/start of ladder program.

- Reading of sequence program on the I/O screen *1
- Start/stop of ladder

> ⚠ **CAUTION**
> Set this setting to ”NO”(0) before shipment of the machine if you want to prohibit operator form editing the program. If you want to protect this setting, please make a sequence that always writes 0 in this bit by your ladder.

RAM WRITE ENABLE (%SK5)

If you set ”RAM WRITE ENABLE” to ”YES”, it enables both the forcing function and the override function.

> **NOTE**
> The override function also requires the setting of ”OVERRIDE ENABLE” in the setting parameters (EDIT/DEBUG).

DATA REGISTER TABLE CONTROL SCREEN (%SK8)

If you set ”DATA REGISTER TABLE CONTROL SCREEN” to ”NO”, the data register table control screen does not be displayed.

HIDE PMC PARAMETER (%SK23)

If you set ”HIDE PMC PARAMETER” to ”YES”, the PMC parameter screen does not be displayed.

> **NOTE**
> Even if this parameter is set to "YES", these settings are invalid if "PROGRAMMER ENABLE" is set to "YES".

PROTECT PMC PARAMETER (%SK24)

If you set "PROTECT PMC PARAMETER" to "YES", the PMC parameter does not change by reading of the I/O screen.

> **NOTE**
> Even if this parameter is set to "YES", these settings are invalid if "PROGRAMMER ENABLE" is set to "YES".

EXAMPLE FOR SETTING PARAMETERS

(1) If you want to prohibit completely operator from accessing the sequence program;

PROGRAMMER ENABLE	(%SK2)	”NO”
HIDE PMC PROGRAM	(%SK1)	”YES”
EDIT ENABLE	(%SK15)	”NO”
ALLOW PMC STOP	(%SK19)	”NO”

(2) If you want to allow operator only monitoring the sequence program;

PROGRAMMER ENABLE	(%SK2)	”NO”
HIDE PMC PROGRAM	(%SK1)	”NO”
EDIT ENABLE	(%SK15)	”NO”
ALLOW PMC STOP	(%SK19)	”NO”

(3) If you want to allow operator monitoring and editing the sequence program;

PROGRAMMER ENABLE	(%SK2)	”NO”
HIDE PMC PROGRAM	(%SK1)	”NO”
EDIT ENABLE	(%SK15)	”YES”
ALLOW PMC STOP	(%SK19)	”NO”

(4) If you want to allow operator monitoring and editing the sequence program which requires stop of ladder;

PROGRAMMER ENABLE	(%SK2)	”NO”
HIDE PMC PROGRAM	(%SK1)	”NO”
EDIT ENABLE	(%SK15)	”YES”
ALLOW PMC STOP	(%SK19)	”YES”

> ⚠ **WARNING**
> If a sequence program is stopped while the machine is operating, the machine may behave unexpectedly. Before stopping the sequence program, make sure that nobody is near the machine and that the tool cannot interfere with the work–piece or machine. Incorrect operation of the machine presents an extreme risk of death or serious injury to the user. Damage the tool, work–piece, and/or the machine is also likely.

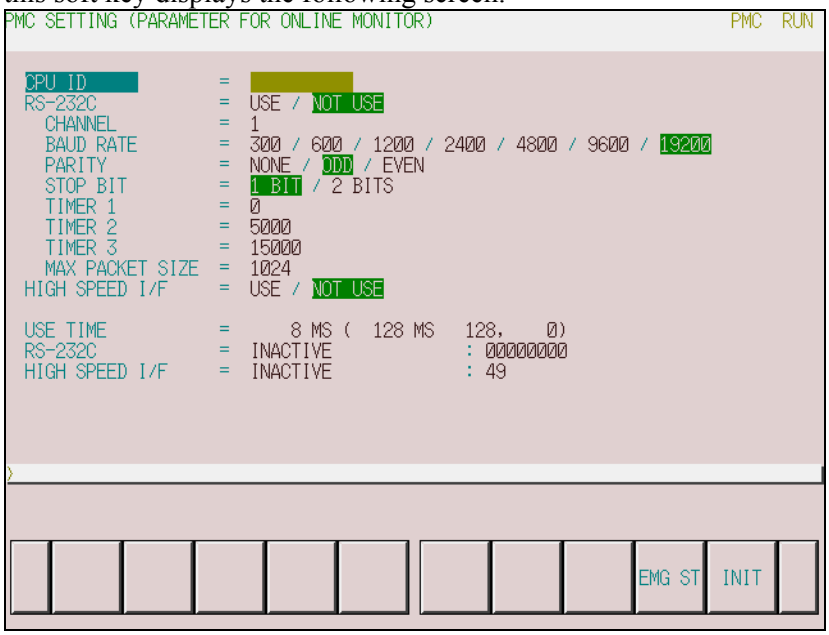
(5) The case that operator who familiar with the machine and the ladder sequence operate all the PMC programmer functions;

PROGRAMMER ENABLE	(%SK2)	”YES”
HIDE PMC PROGRAM	(%SK1)	”NO”

> ⚠ **WARNING**
> If a sequence program is stopped while the machine is operating, the machine may behave unexpectedly. Before stopping the sequence program, make sure that nobody is near the machine and that the tool cannot interfere with the work–piece or machine. Incorrect operation of the machine presents an extreme risk of death or serious injury to the user. Damage the tool, work–piece, and/or the machine is also likely.

Screen for Setting/Displaying Online Monitor Parameters (ONLINE)

If PROGRAMMER ENABLE is set to YES on the GENERAL screen, the [ONLINE] soft key appears on the setting menu screen. Pressing this soft key displays the following screen.



When you use the online function, the setting of communication condition is necessary in advance.

> ⚠ **CAUTION**
> When the online function is used with RS-232C, the selected channel is occupied by the PMC system. To use other input/output functions with RS-232C, specify other channel setting than the one used by online function.

Menu descriptions

- CPU ID
  The CPU ID value is displayed. The value can also be entered here, but its entry is usually not necessary.
- RS-232C (prompt)
  USE: An RS-232C port can be connected to FANUC LADDER-IIIC.
  NOT USE: RS-232C port is not used.
  Note) If no RS-232C is to be connected to FANUC LADDER-IIIC, select NOT USE.
- CHANNEL
  A channel number to be used is displayed. The number can also be entered.
- BAUD RATE
  300: A baud rate of 300 is specified.
  600: A baud rate of 600 is specified.
  900: A baud rate of 900 is specified.
  1200: A baud rate of 1200 is specified.
  2400: A baud rate of 2400 is specified.
  4800: A baud rate of 4800 is specified.
  9600: A baud rate of 9600 is specified.
  19200: A baud rate of 19200 is specified.
- PARITY
  NONE: No parity is specified.
  ODD: Odd parity is specified.
  EVEN: Even parity is specified.
- STOP BIT
  1 BIT: The number of stop bits is set to 1.
  2 BITS: The number of stop bits is set to 2.
- TIMER 1
  The value in communication parameter timer 1 is displayed. The value can also be entered, but its specification is usually not necessary.
- TIMER 2
  The value in communication parameter timer 2 is displayed. The value can also be entered, but its specification is usually not necessary.
- TIMER 3
  The value in communication parameter timer 3 is displayed. The value can also be entered, but its specification is usually not necessary.
- MAX PACKET SIZE
  The maximum packet size for the communication parameter is displayed. The size can also be entered, but its specification is usually not necessary.
- RS-232C (status display)
  The status of an RS-232C port is displayed.
  INACTIVE: No RS-232C port is in use.
  STOPPING: An RS-232C port is closed.
  STARTING: An RS-232C port is open.
  STAND-BY: An RS-232C port is waiting to be connected to FANUC LADDER-IIIC.
  CONNECTED: An RS-232C port has been connected to FANUC LADDER-IIIC.

If the high speed interface is included in the system configuration, the HIGH SPEED I/F prompt menu appears above the RS-232C status display menu, and the HIGH SPEED I/F status display menu appears below the RS-232C status display menu.

- HIGH SPEED I/F (prompt)
  USE: A HIGH SPEED I/F port can be connected to FANUC LADDER-IIIC.
  NOT USE: No HIGH SPEED I/F port will be connected to FANUC LADDER-IIIC.
- HIGH SPEED I/F (status display)
  The status of a HIGH SPEED I/F port is displayed.
  INACTIVE: No HIGH SPEED I/F port is in use.
  STOPPING: A HIGH SPEED I/F port is closed.
  STARTING: A HIGH SPEED I/F port is open.
  STAND-BY: A HIGH SPEED I/F port is waiting to be connected to FANUC LADDER-IIIC.
  CONNECTED: A HIGH SPEED I/F port has been connected to FANUC LADDER-IIIC.

Soft key descriptions
[EMG ST]: Pressing this key causes communication to be terminated. It is used if it is impossible to terminate a connection due to abnormal communication.
[INIT]: Pressing this key initializes the parameter settings.

Setting of online connection

To communicate with FANUC LADDER-IIIC, you need to put the PMC system in waiting situation of the connection. There are two ways for setting this, setting at PMC screen and setting in NC parameter. Also, there are three connection types, for example Ethernet or RS-232C.

(a) How to set at PMC Screen
To display the soft key [ONLINE] in the PMC setting menu screen, set “PROGRAMMER ENABLE” to “YES” in the setting screen. When pushing the soft key [SETING]->[ONLINE], the online setting screen is displayed.

1. Case of connection by RS-232C (FANUC LADDER-IIIC)
   (1) Check that “NOT USE” is selected at the “RS-232C” item.
   (2) Set the parameter of “CHANNEL” and “BAUD RATE”.
   (3) Move the cursor to the “RS-232C” item with Up or Down Cursor key.
   (4) Select “USE” with Left or Right Cursor key.

2. Case of connection by Ethernet (FANUC LADDER-IIIC)
   (1) Move the cursor to the “HIGH SPEED I/F” item with Up or Down Cursor key.
   (2) Select “USE” with Left or Right Cursor key.

NOTE
1 When both "RS-232C = USE" and "HIGH SPEED I/F = USE" are selected, the PMC system will communicate with the application which is connected at first. If PMC system is already connecting with an application, it can not connect with other applications.
2 When you use the online function by Ethernet, the setting of Ethernet parameters at CNC is necessary in advance.

(b) Setting of online connection by NC parameter

You can enable and disable the online connection for Ethernet and RS-232C by NC parameter No.24 without setting on the PMC online monitor setting screen. This NC parameter is made effective immediately after setting the parameter.

If the value of this parameter is changed, the items "RS232-C"and "HIGH SPEED I/F" in the online monitor screen are automatically changed too. Please refer to following table.

Contents of NC parameter No.24.

NC parameter No.24	Meanings	Each item of the online monitor screen after setting	
		RS-232C	HIGH SPEED I/F
0	The settings on the online monitor setting screen are effective.	This does not affect "RS-232C" and "HIGH SPEED I/F".	
1	Enables "Channel 1 of RS-232C" and disables "HIGH SPEED I/F".	USE (Channel 1)	NOT USE
2	Enables "Channel 2 of RS-232C" and disables "HIGH SPEED I/F".	USE (Channel 2)	NOT USE
10	Disables "RS-232C" and enables "HIGH SPEED I/F".	NOT USE	USE
11	Enables "Channel 1 of RS-232C" and "HIGH SPEED I/F".	USE (Channel 1)	USE
12	Enables "Channel 2 of RS-232C" and "HIGH SPEED I/F".	USE (Channel 2)	USE
3 to 10 13 to 254	Reserve (Don't use this setting.)	(Reserved)	(Reserved)
255	Terminates communication forcibly. It is the same effect as soft key [EMG ST].	NOT USE	NOT USE

NOTE
The Smart Download to PMC and the Verify Equality with the Logic of the FANUC LADDER-IIIC are only supported by HIGH SPEED I/F. RS232C does not support these functions.

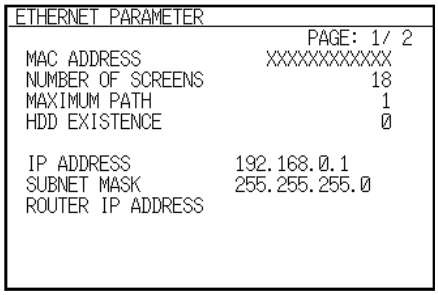
How to set the parameter
(1) Display the No.24 of NC parameter.
(2) To connect by Ethernet, input "10", "11" or "12".
(3) To connect by RS-232C, input "1", "2", "11" or "12".

NOTE
Even if the setting of the online monitor screen of PMC is changed, the value of No.24 in NC parameter is not changed

Setting of Ethernet parameters

When you try to connect FANUC LADDER-IIIC with PMC by Ethernet, it is necessary to set some Ethernet parameters. The setting of Ethernet parameters can be set in the following Ethernet parameter screen of CNC. Please refer to "FANUC Ethernet Board/DATA SERVER Board OPERATOR'S MANUAL" (B-63354EN) about the detail of the setting screen and setting parameters. The setting item necessary for Ethernet connection for PMC online function is as follows.

- IP ADDRESS (Set the IP address of CNC. 192.168.0.1 etc.)
- SUBNET MASK(Set the mask address of the IP address. 255.255.255.0 etc.)
- ROUTER IP ADDRESS (If you use the router, set the Router IP Address.)
- PORT NUMBER (TCP) (8193 etc.)



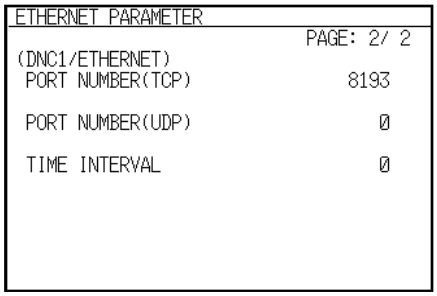
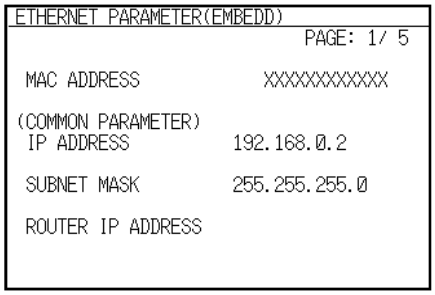


Fig. (a)
Ethernet Board



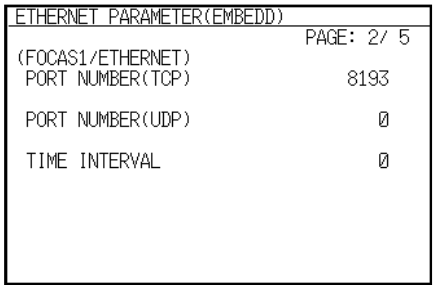
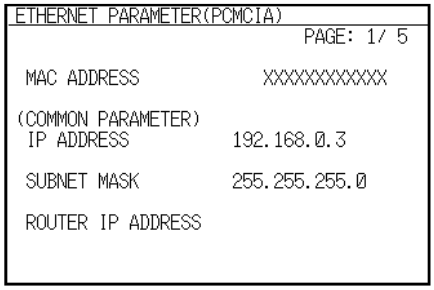


Fig. (b)
Embedded Ethernet



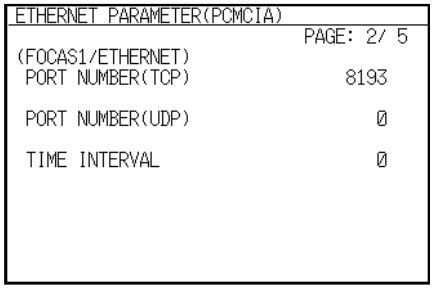
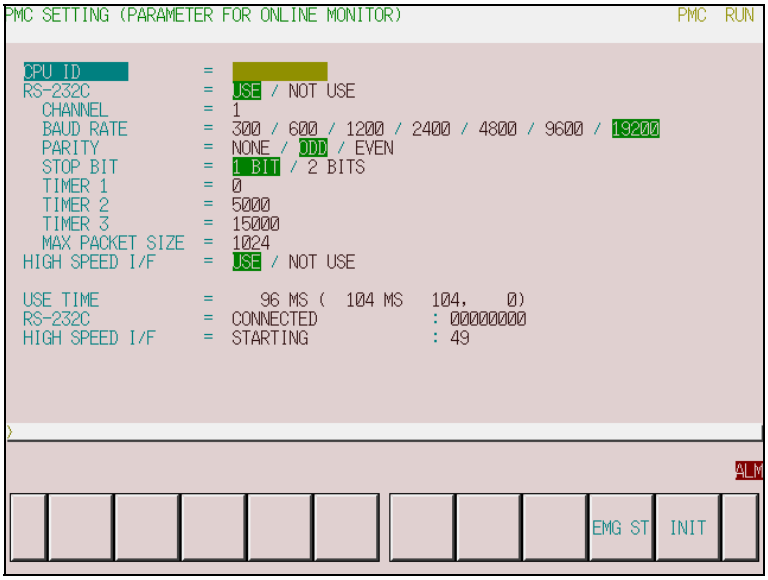


Fig. (c)
Ethernet Card (PCMCIA)

Communication Status

The communication status of RS-232C and HIGH SPEED I/F are displayed at the online monitor screen during the online communication.



USE TIME	:	The maximum time in the communication processing is displayed.
RS-232C	:	The communication condition of RS-232C is displayed.
HIGH SPEED I/F	:	The communication condition of HIGH SPEED I/F is displayed.
ETHER_BOARD	:	Displayed during the communication with Ethernet board. The IP address of the communication partner is displayed.
EMB_ ETHERNET	:	Displayed during the communication with embedded Ethernet. The IP address of the communication partner is displayed.

The display messages and the meanings are shown in the table of below.

Displayed messages	Meanings
INACTIVE	The communication is inactive.
STOPPING	The communication is being stopped. (Wait for the termination of communication)
STARTING	The communication is being started. (Wait for the termination of communication over another communication path)
STAND-BY	The communication is active and in standby mode.
CONNECTED	The communication is active and being connected.
NO OPTION	The port can be not opened because there is not option of RS-232C.
BAD PARAMETER	Invalid open parameters are specified.
TIMEOUT ERROR	A time-out has occurred and communication is aborted.
TIMEOUT(K) ERROR	A time-out has occurred and communication is aborted.
BCC ERROR	A Block Check Code (packet parity) error has occurred.
PARITY ERROR	A parity error has occurred.
OVER-RUN ERROR	A reception overrun has occurred and the communication can not recover.
SEQUENCE ERROR	Packets are out of sequence. (Incorrect procedure)
DATA ERROR	Incorrect packets have been received through retry process.
QUEUE OVERFLOW	The transmit/receive queue has overflowed.
DISCONNECTED	Communication has been terminated successfully.
NO CONNECTION	The cable is disconnected.

6.5 SEQUENCE PROGRAM EXECUTION

6.5.1 Starting and Stopping a Sequence Program

(1) Starting a sequence program (RUN)
When a program is stopped, clicking the [RUN] soft key causes the program to start and the status line display to change to "PMC RUN."
The sequence program starts from the beginning. The soft key changes to [STOP].

(2) Stopping a sequence program (STOP)
When a program is executed, clicking the [STOP] soft key causes the program to stop and the status line display to change to "PMC STOP."
The soft key changes to [RUN].

> ⚠ **WARNING**
> If the sequence program is stopped while the machine is operating, the machine may behave in an unexpected way.
> Before stopping the sequence program, ensure that there are no people near the machine and that the tool cannot collide with the workpiece or machine.
> Otherwise, there is an extreme risk of death or serious injury, as well as the likelihood of the tool, workpiece, and machine being damaged.

(3) Automatic operation of a sequence program
When AUTOMATIC LADDER START is set to AUTO (the keep relay %SK3 = 0) on the setting screen, a sequence program can be executed automatically when the power is turned on.

6.5.2 Forced Termination of Sequence Program

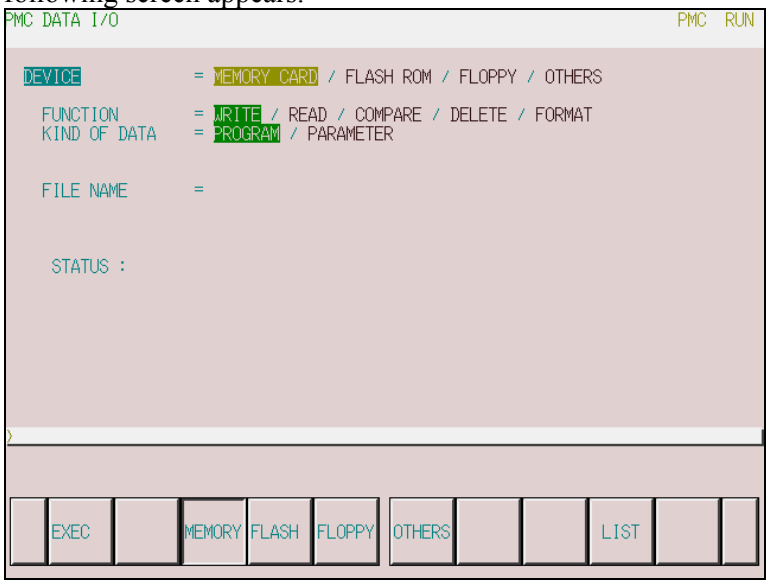
To forcibly stop starting the sequence program at power-on, turn on the power by pressing the [Z] key while pressing the [CAN] key. This method is effective for locating the error when a system error occurs after power is turned on and when the error may be caused by the sequence program.
Never perform this operation in a normal state.

> ⚠ **WARNING**
> In normal operation, do not use the Z + CAN keys to stop the sequence program forcibly.

6.6 WRITING, READING, AND VERIFYING THE SEQUENCE PROGRAM AND PMC PARAMETER DATA (I/O)

6.6.1 I/O Screen

When the [I/O] soft key on the PMC main menu is pressed, the following screen appears.



On this screen, sequence programs and PMC parameters can be written to the specified device, read from the device, and compared. The question selection cursor, which moves vertically from one question to another, is displayed, as is the option selection cursor, which moves horizontally from one option to another. The displayed soft keys differ depending on the position of the question selection cursor.

The following types of devices can be used for input/output. The desired device type can be selected by positioning the question selection cursor to “DEVIDE” and either moving the option selection cursor to that type or selecting the soft key corresponding the type.

MEMORY CARD	Data can be output to and input from a memory card.
FLASH ROM	Data can be output to and input from frash ROM.
FLOPPY	Data can be output to and input from handy files or floppy cassettes.
OTHERS	Data can be output to and input from other input/output devices.

When you read a file from an I/O device, one of following messages appear and whether to operate the important thing is confirmed.

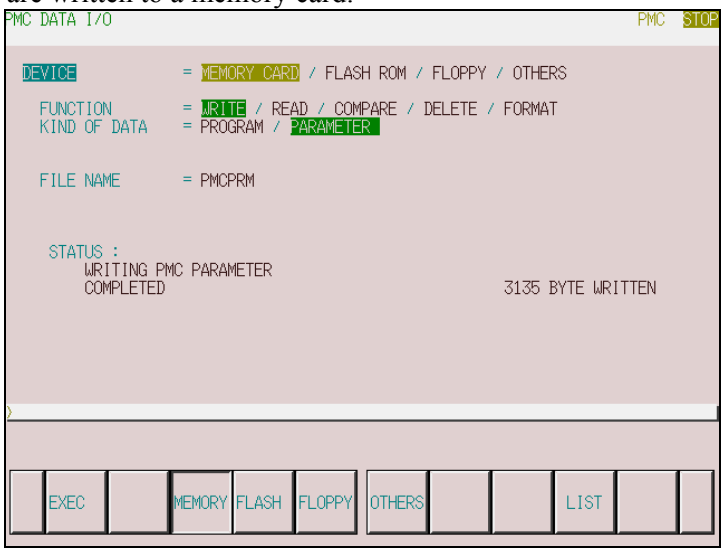
```
<WARNING> READING SEQUENCE PROGRAM OR PMC PARAMETER
REQUIRES SPECIAL CARE.
* READING IMPROPER DATA MAY CAUSE UNEXPECTED
MOVEMENT OF MACHINE.
* PROGRAM WILL BE STOPPED BY READING SEQUENCE
PROGRAM.
ARE YOU SURE YOU WANT TO READ THIS FILE?
```

When you proceed to read PMC parameters, new PMC parameters will be stored even if the ladder program is running. If the ladder and the parameter are protected, the error message of "THIS FUCTION IS PROTECTED" is displayed. Please refer to “Programmer Protection Function” in Subsection 6.4.4.

⚠ WARNING

1 If a Ladder program is input while a Ladder program is being executed, the executions of the Ladder program stop automatically by the setting of GENERAL screen. You have to pay special attention to stop Ladder program. Stopping Ladder program in a wrong timing, or with machine in improper status, may cause unexpected reaction of machine. You have to make it sure that machine is in proper status, and nobody is near the machine when you stop Ladder program.
2 At stopping Ladder program, it may take rather long time to completely stop it in some cases according to the activity of Ladder program. If Ladder takes too long time to stop, or never stop, correct Ladder program.
3 If the PMC parameters are input while a Ladder program is being executed, You have to special attention to input it. Because changed PMC parameters, may cause unexpected effect to Ladder. You have to make it sure that PMC parameters are not effect to Ladder when you input PMC parameters.
4 Set keep relay %SK2 to 0 when the machine tool is shipped.

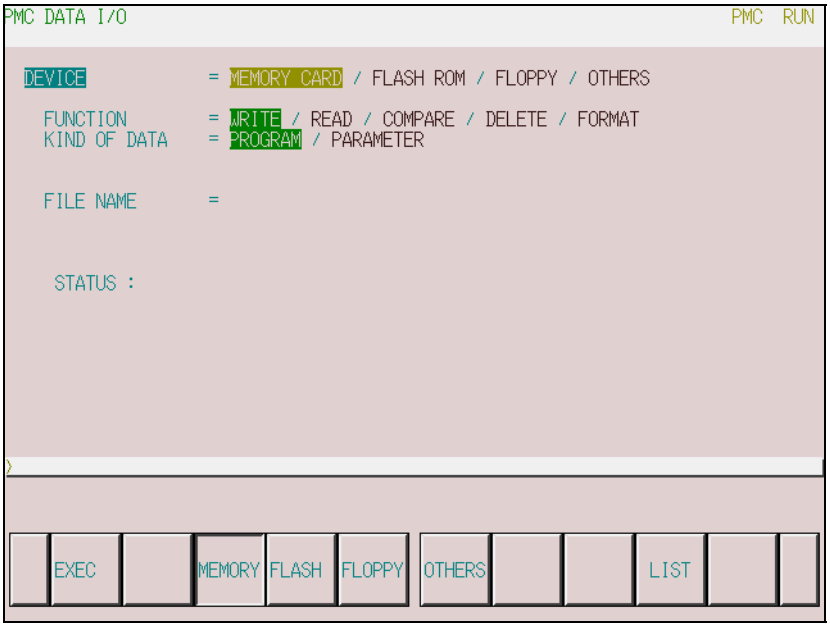
In STATUS in the lower part of the screen, a detailed explanation of execution and the execution status are displayed. During write, read, and comparison, the size of the data already transferred is indicated as the execution (intermediate) result.
The following gives a display example shown when PMC parameters are written to a memory card:



NOTE
1 For an explanation of error messages on I/O screen, see Subsection 6.6.9, “I/O screen Error Messages”.
2 The conditions of outputting PMC parameters
1) When sequence program is stopped
You can input/output them.
2) When sequence program is executed
You must satisfy the following conditions.

Output (WRITE)	Set NC to “EDIT” mode.
Input (READ)	Set NC to “Emergency Stop” status, and, set “PWE” of NC parameters to 1.

6.6.2 Outputting to and Inputting from Memory Cards



When "MEMORY CARD" is selected for DEVICE, output to and input from memory cards are enabled.

- FUNCTION

  A data I/O command appears. Select FUNCTION with the item select cursor, then select an item by moving the content select soft key horizontally or pressing an appropriate soft key.

  Soft keys displayed when the question selection cursor is positioned to "FUNCTION"

  Explanation of options

WRITE:	Outputs data from the PMC to a memory card.
READ:	Inputs data from a memory card to the PMC.
COMPARE:	Compares the sequence programs on the PMC with those on a memory card.
DELETE:	Deletes files from a memory card. (Files on a flash card cannot be deleted.)
FORMAT:	Formats a memory card.

> ⚠ **CAUTION**
> When "FORMAT" is selected and executed, all data in the memory card is lost. Be careful when executing this function.

When you read a file from a memory card, one of following messages appear and whether to operate the important thing is confirmed.

```
<WARNING> READING SEQUENCE PROGRAM OR PMC PARAMETER
REQUIRES SPECIAL CARE.
* READING IMPROPER DATA MAY CAUSE UNEXPECTED
MOVEMENT OF MACHINE.
* PROGRAM WILL BE STOPPED BY READING SEQUENCE
PROGRAM.
ARE YOU SURE YOU WANT TO READ THIS FILE?
```

When you proceed to read PMC parameters, new PMC parameters will be stored even if the ladder program is running. If the ladder and the parameter are protected, the error message of "THIS FUCTION IS PROTECTED" is displayed. Please refer to “Programmer Protection Function” in Subsection 6.4.4.

⚠ WARNING

1 If a Ladder program is input while a Ladder program is being executed, the executions of the Ladder program stop automatically by the setting of GENERAL screen. You have to pay special attention to stop Ladder program. Stopping Ladder program in a wrong timing, or with machine in improper status, may cause unexpected reaction of machine. You have to make it sure that machine is in proper status, and nobody is near the machine when you stop Ladder program.
2 At stopping Ladder program, it may take rather long time to completely stop it in some cases according to the activity of Ladder program. If Ladder takes too long time to stop, or never stop, correct Ladder program.
3 If the PMC parameters are input while a Ladder program is being executed, You have to special attention to input it. Because changed PMC parameters, may cause unexpected effect to Ladder. You have to make it sure that PMC parameters are not effect to Ladder when you input PMC parameters.
4 Set keep relay %SK2 to 0 when the machine tool is shipped.

- KIND OF DATA

  KIND OF DATA is displayed only when “WRITE” is selected for “FUNCTION.”

  Set the type of data to be output by moving the cursor horizontally to that type or by pressing the corresponding soft key.

  Soft keys displayed when the question selection cursor is positioned to “KIND OF DATA”

Explanation of options

PROGRAM: Outputs sequence programs only.
PARAMETER: Outputs PMC parameters.

- FILE NO.

  FILE NO. is displayed only when "READ," "COMPARE," or "DELETE" is selected for "FUNCTION."
  Enter the file number in the edit box.

- FILE NAME

  FILE NAME is displayed when "WRITE," "READ," "COMPARE," or "DELETE" is selected for "FUNCTION."
  Enter the file name in the edit box.
  When "READ," "COMPARE," or "DELETE" is selected for "FUNCTION," the file name corresponding to the file number entered in "FILE NO." is displayed automatically.
  The file name must be in MS-DOS format: a file name of up to 8 characters followed by an extension of up to 3 characters.
  When "WRITE" is selected for "FUNCTION" and the file name is not entered, the following names are automatically assumed.

DATA KIND	File name
LADDER	PMCSD7.LAD
PARAM	PMCSD7.PRM

> ⚠ **CAUTION**
> When both "FILE NO." and "FILE NAME" are displayed at the same time, and a value is entered for "FILE NO." and another file name is entered in "FILE NAME," the value entered in "FILE NO." is erased and the file name entered in "FILE NAME" becomes effective.

Explanation of soft keys

[EXEC]: Executes the function selected for "FUNCTION." During execution, the soft key disappears and the [CANCEL] soft key appears to the right of the key.
[CANCEL]: Cancels the execution of the function. When the function terminates normally, the soft key disappears.
[LIST]: Replaces the current display with the memory card list screen. See Subsection 6.6.3 "Memory Card List Screen" for details.

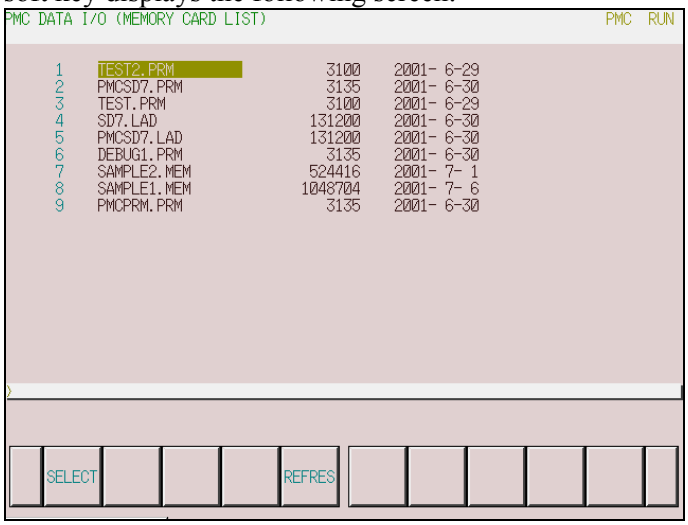
The system supports the following memory cards:

O: Supported
X: Not supported

	SRAM Card	Flash Memory Card		ATA Card
		Supported Card	Unsupported Card	
Read of a file	O	O	O	O
Format of a card	O	O	X	O
Write of a file	O	O	X	O
Delete of a file	O	X	X	O
List of a file	O	O	O	O

6.6.3 Memory Card List Screen

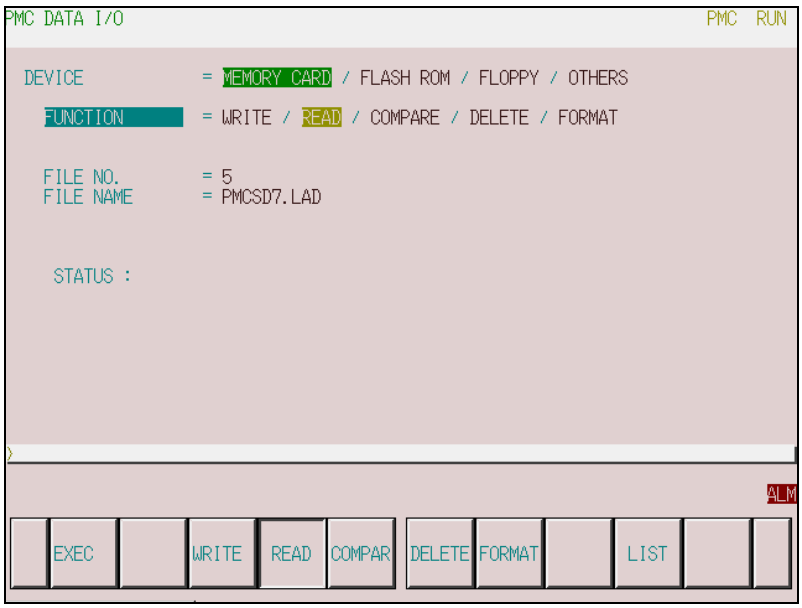
When MEMORY CARD is selected in DEVICE, pressing the [LIST] soft key displays the following screen:



If a memory card holding files is in the slot, the contents of the memory card are displayed as shown in the above.

NOTE
Up to 128 files can be displayed on this screen. When 129 or more files are saved in the memory card, the 129th and subsequent files are ignored.

When a file is selected on this screen, the screen display can be returned to the previous screen. To select a file, place the cursor at the name of the file, then press either the [SELECT] soft key or the INPUT key. After the key entry, the screen display switches to the previous screen automatically. In this case, the cursor is positioned at READ on the FUNCTION menu, and the number and name of the file selected on the list screen are indicated in the FILE NO. and FILE NAME fields, respectively. A display example is shown below.



To return the screen display to the previous screen without selecting a file, press the return key.

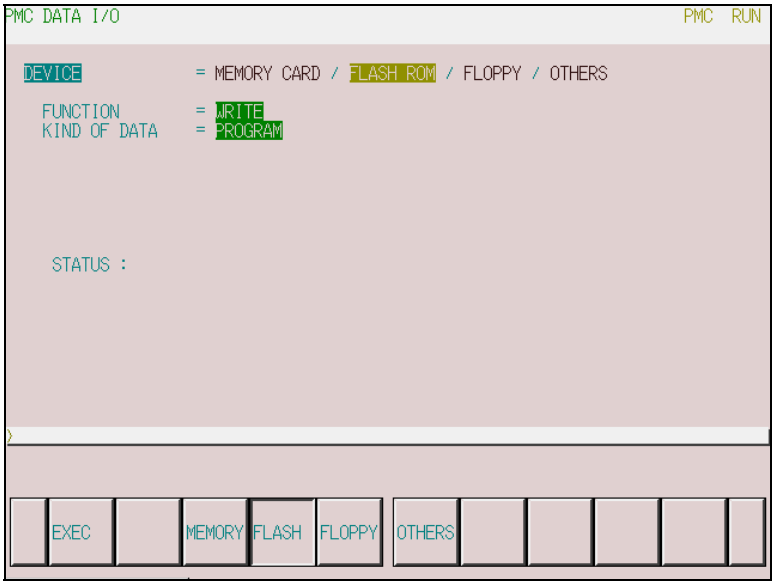
When the memory card is replaced with another card while the list screen is being displayed, the displayed information is not updated automatically. In this case, press the [REFRES] soft key. The contents of the new memory card are then displayed.

Explanation of soft keys

[SELECT]: Selects a file, and returns the screen display to the previous screen.

[REFRES]: Redisplays the contents of the memory card.

6.6.4 Outputting to and Inputting from Flash ROM



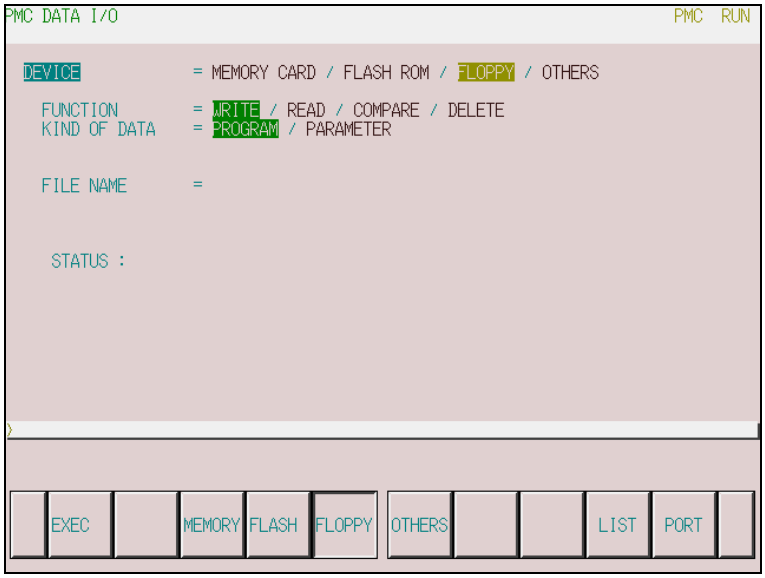
When "FLASH ROM" is selected for DEVICE, output to and input from flash ROM are enabled.

- FUNCTION
  The available data input/output commands are displayed. Select the desired command by moving the cursor horizontally to that command or select it with the corresponding soft key.
  Explanation of options
    WRITE: Outputs sequence programs from the PMC to flash ROM.

- KIND OF DATA
  Explanation
    PROGRAM: Outputs sequence programs only.

  Explanation of soft keys
    [EXEC]: Executes the function selected for "FUNCTION."

6.6.5 Outputting to and Inputting from Floppy



When "FLOPPY" is selected for "DEVICE", output to and input from a floppy cassette or a handy file that is connected via RS-232C are enabled.

- FUNCTION

  The available data input/output commands are displayed. Select the desired command by moving the cursor horizontally to that command or select it with the corresponding soft key.

  Soft keys displayed when the question selection cursor is positioned to "FUNCTION"

  Explanation of options

WRITE:	Outputs data from the PMC to a floppy disk or a handy file.
READ:	Inputs data from a floppy disk or a handy file to the PMC.
COMPARE:	Compares the sequence program on the PMC with those on a floppy disk or a handy file.
DELETE:	Deletes a file from a floppy disk or a handy file.

When you read a file from a floppy cassette or a handy file, one of following messages appear and whether to operate the important thing is confirmed.

<WARNING> READING SEQUENCE PROGRAM OR PMC PARAMETER REQUIRES SPECIAL CARE.
*** READING IMPROPER DATA MAY CAUSE UNEXPECTED MOVEMENT OF MACHINE.**
*** PROGRAM WILL BE STOPPED BY READING SEQUENCE PROGRAM.**
ARE YOU SURE YOU WANT TO READ THIS FILE?

When you proceed to read PMC parameters, new PMC parameters will be stored even if the ladder program is running. If the ladder and the parameter are protected, the error message of "THIS FUCTION IS PROTECTED" is displayed. Please refer to “Programmer Protection Function” in Subsection 6.4.4.

⚠ WARNING

1 If a Ladder program is input while a Ladder program is being executed, the executions of the Ladder program stop automatically by the setting of GENERAL screen. You have to pay special attention to stop Ladder program. Stopping Ladder program in a wrong timing, or with machine in improper status, may cause unexpected reaction of machine. You have to make it sure that machine is in proper status, and nobody is near the machine when you stop Ladder program.
2 At stopping Ladder program, it may take rather long time to completely stop it in some cases according to the activity of Ladder program. If Ladder takes too long time to stop, or never stop, correct Ladder program.
3 If the PMC parameters are input while a Ladder program is being executed, You have to special attention to input it. Because changed PMC parameters, may cause unexpected effect to Ladder. You have to make it sure that PMC parameters are not effect to Ladder when you input PMC parameters.
4 Set keep relay %SK2 to 0 when the machine tool is shipped.

- KIND OF DATA

“KIND OF DATA” is displayed only when “WRITE” is selected for “FUNCTION”.

Set the type of data to be output by moving the cursor horizontally to that command or select it with the corresponding soft key.

Soft keys displayed when the question selection cursor is positioned to “KIND OF DATA”

Explanation of options

PROGRAM: Outputs sequence program.
PARAMETER: Outputs PMC parameters.

- FILE NO.

“FILE NO.” is displayed only when “READ”, “COMPARE”, or “DELETE” is selected for “FUNCTION”.

Enter the file number in the edit box.

- FILE NANE

"FILE NAME" is displayed when "WRITE", "READ", "COMPARE", or "DELETE" is selected for "FUNCTION".

Enter the file name in the edit box.

When "READ", "COMPARE", or "DELETE" is selected for "FUNCTION", the file name corresponding to the file number entered in "FILE NO." is displayed automatically.

When you output to or input from a floppy disk formatted in DOS format, the file name must be in MS-DOS format: a file name of up to eight characters followed by an extension of up to three characters.

When you output to or input from floppy disk formatted in FANUC format, a file name of up to 17 characters will be input.

When "WRITE" is selected for "FUNCTION" and the file name is not entered, the following names are automatically assumed.

DATA KIND	File name
LADDER	PMCSD7.LAD
PARAM	PMCSD7.PRM

⚠ CAUTION

1 When both "FILE NO." and "FILE NAME" are displayed at the same time, and a value is entered for "FILE NO." and another file name is entered in "FILE NAME", the value entered in "FILE NO." is erased and the file name entered in "FILE NAME" becomes effective.

2 Specifying the same name as that of an existing file results in an error.

Explanation of soft keys

[EXEC]: Executes the function selected for "FUNCTION". During execution, the soft key disappears and the [CANCEL] soft key appears to the right of the key.

[CANCEL]: Cancels the execution of the function. When the function terminates normally, the soft key disappears.

[LIST]: Replaces the current display with the Floppy list screen. See Subsection 6.6.6, "Floppy List Screen" for details.

[PORT]: Replaces the current display with the screen for setting communication parameters. See Subsection 6.6.8, "Port Setting Screen" for details.

6.6.6 Floppy List Screen

When “FLOPPY” is selected for “DEVICE”, pressing the [LIST] soft key display the following screen.

The contents of the floppy cassettes or the handy files are displayed. When a file is selected on this screen, the screen display can be returned to the previous screen. To select the file, place the cursor at the name of the file, then press either the [SELECT] soft key or the INPUT key. After the key entry, the screen display switches to the previous screen automatically. In this case, the cursor is positioned at “READ” on the “FUNCTION” menu, and the number and name of the file selected on the list screen are indicated in the “FILE NO.” and “FILE NAME” fields, respectively.

To return the screen display to the previous screen without selecting a file, press the return key.

When the floppy cassette or the handy file is replaced with another one while the list screen is being displayed, the displayed information is not updated automatically. In this case, press the [REFRES] soft key. The contents are then displayed.

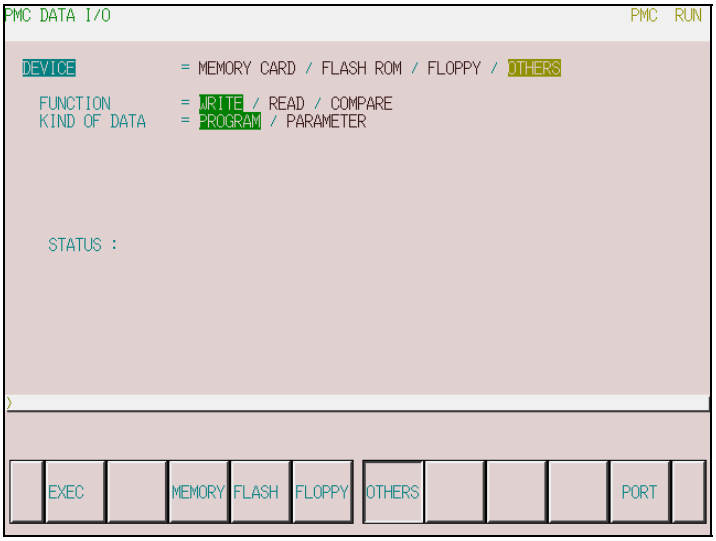
NOTE
Up to 128 files can be displayed on this screen. When 129 or more files are saved, the 129th and subsequent files are ignored.

Explanation of soft keys

[SELECT] Selects a file, and returns the screen display to the previous screen.

[REFRES] Redisplays the contents of a floppy cassette or a handy file.

6.6.7 Outputting to and Inputting from Other Input/Output Devices



When "OTHERS" is selected for "DEVICE", output to and input from other input/output devices are enabled.

- FUNCTION

The available data input/output commands are displayed. Select the desired command by moving the cursor horizontally to that command or select it with the corresponding soft key.
Soft keys displayed when the question selection cursor is positioned to "FUNCTION"

Explanation of options

WRITE	Outputs data from the PMC to other input/output device.
READ	Inputs data from other input/output device to the PMC.
COMPAR	Compares the sequence program on the PMC with those on other input/output device.

When you read a file from a I/O device, one of following messages appear and whether to operate the important thing is confirmed.

```
<WARNING> READING SEQUENCE PROGRAM OR PMC PARAMETER
REQUIRES SPECIAL CARE.
* READING IMPROPER DATA MAY CAUSE UNEXPECTED
MOVEMENT OF MACHINE.
* PROGRAM WILL BE STOPPED BY READING SEQUENCE
PROGRAM.
ARE YOU SURE YOU WANT TO READ THIS FILE?
```

When you proceed to read PMC parameters, new PMC parameters will be stored even if the ladder program is running. If the ladder and the parameter are protected, the error message of "THIS FUCTION IS PROTECTED" is displayed. Please refer to "Programmer Protection Function" in Subsection 6.4.4.

> ⚠ **WARNING**
> 1 If a Ladder program is input while a Ladder program is being executed, the executions of the Ladder program stop automatically by the setting of GENERAL screen. You have to pay special attention to stop Ladder program. Stopping Ladder program in a wrong timing, or with machine in improper status, may cause unexpected reaction of machine. You have to make it sure that machine is in proper status, and nobody is near the machine when you stop Ladder program.
> 2 At stopping Ladder program, it may take rather long time to completely stop it in some cases according to the activity of Ladder program. If Ladder takes too long time to stop, or never stop, correct Ladder program.
> 3 If the PMC parameters are input while a Ladder program is being executed, You have to special attention to input it. Because changed PMC parameters, may cause unexpected effect to Ladder. You have to make it sure that PMC parameters are not effect to Ladder when you input PMC parameters.
> 4 Set keep relay %SK2 to 0 when the machine tool is shipped.

- KIND OF DATA

"KIND OF DATA" is displayed only when "WRITE" is selected for "FUNCTION".
Set the type of data to be output by moving the cursor horizontally to that command or select it with the corresponding soft key.

Soft keys displayed when the question selection cursor is positioned to "KIND OF DATA"

Explanation of options

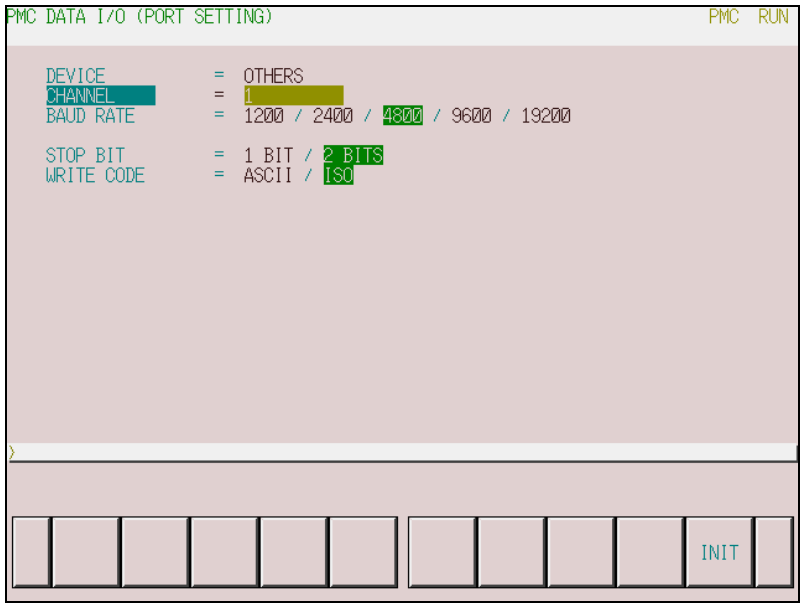
PROGRAM:	Outputs sequence program.
PARAMETER:	Outputs PMC parameters.

Explanation of soft keys

[EXEC]:	Executes the function selected for "FUNCTION". During execution, the soft key disappears and the [CANCEL] soft key appears to the right of the key.
[CANCEL]:	Cancels the execution of the function. When the function terminates normally, the soft key disappears.
[PORT]:	Replaces the current display with the screen for setting communication parameters. See Subsection 5.6.8, "Port Setting Screen" for details.

6.6.8 Port Setting Screen

When “FLOPPY” or “OTHERS” is selected for “DEVICE”, the [PORT] soft key is displayed. When the key is pressed, the following screen appears. The following gives a display example shown when “OTHERS” is selected for “DEVICE”.



This screen allows the setting of the communication data required for communication using the RS-232C. Communication data can be set for each of the two types of devices independently of the other.
Selected device type is displayed to “DEVICE” menu on screen.

Explanation of each question

- CHANNEL

  Check that an RS-232C cable is connected to the main board of the control unit. Directly enter the number corresponding to the connected connector.

  1 ..... JD36A
  2 ..... JD36B

- BAUDRATE

  1200: Sets the baud rate to “1200”.
  2400: Sets the baud rate to “2400”.
  4800: Sets the baud rate to “4800”.
  9600: Sets the baud rate to “9600”.
  19200: Sets the baud rate to “19200”.

- STOP BIT

  1 BIT: Sets the number of stop bits to “1”.
  2 BIT: Sets the number of stop bits to “2”.

- WRITE CODE
  "WRITE CODE" is displayed when "OTHERS" is selected for "DEVICE".
  ASCII: Sets the output code to "ASCII".
  ISO: Sets the output code to "ISO".

NOTE
Parity is always "NONE".

Explanation of soft key
[INIT] Sets all the parameters to their initial values.

Initial values

	DEVICE = FLOPPY	DEVICE = OTHERS
CHANNEL	1	1
BAUD RATE	4800	4800
STOP BIT	2 BITS	2 BITS
WRITE CODE	(None)	ISO

6.6.9 I/O Screen Error Messages

The error messages that may appear on the I/O screen and their meanings and actions are listed below.

Error messages displayed during memory card I/O operation

Displayed error message	Meaning and action
MEMORY CARD IS NOT READY	No memory card is installed. Action: Check whether a memory card is installed.
MEMORY CARD IS FULL	There is no available space in the memory card. Action: Delete files to create available space.
MEMORY CARD IS WRITE PROTECTED	The memory card is write-protected. Action: Release the write protection of the memory card.
MEMORY CARD IS NOT FORMATTED	The memory card cannot be recognized. Action: Format the memory card.
TOO MANY FILES IN MEMORYCARD	There are too many files. Action: Delete unnecessary files to reduce the number of files.
FILE NOT FOUND	The specified file cannot be found. Action: On the list screen, check the file name or file number.
FILE IS READ-ONLY	Write to the specified file is not permitted. Action: Check the attributes of the file.
FILE NAME IS INVALID	The file name is illegal. Action: Specify the file name in MS-DOS form.
CAN NOT FORMAT MEMORYCARD	The memory card cannot be formatted. Action: The NC cannot format this memory card. Use another unit such as a personal computer to format the memory card.
UNSUPPORTED MEMORYCARD	This memory card is not supported. Action: Replace the memory card with another one.
CAN NOT DELETE FILE	An error occurred when a file was deleted from the memory card. Action: Check the attributes of the file.
MEMORYCARD BATTERY ALARM	The battery of the memory card has become weak. Action: Replace the battery of the memory card.
THIS FILE NAME IS ALREADY USED	The file name is already used. Action: Change the file name to another one.
MEMORYCARD ACCESS ERROR	The memory card cannot be accessed. Action: Replace the memory card with another one.
DIFFERENCE FOUND	File comparison detected a mismatch.
MEMORYCARD IS LOCKED BY OTHER FUNCTION	Another PMC user is using the memory card. Action: Wait until the PMC user completes processing, then retry.
MEMORY CARD HEADER ROM DATA ID IS ILLEGAL	An attempt was made to read a file, but its ROM data ID was illegal. Action: This file cannot be read. Check the type of the file.
FLASH ROM HEADER ROM DATA ID IS ILLEGAL	An attempt was made to read a file, but its ROM data ID was illegal. Action: This file cannot be read. Check the type of the file.
FILE NUMBER CAN NOT SELECTED	The file number cannot be selected. Action: If the file does not exist, the key entry is invalid. If this error occurs even when the cursor is placed at a file name, contact the FANUC service center.
INPUT FILE NUMBER IS NOT EXIST	The entered file number is not present. The entered number exceeds the total number of files. Action: Check the total number of files on the list screen.
FILE NUMBER IS RESTRICTED TO ”128”	A value up to 128 can be entered as the file number. Action: Enter a numeric value not exceeding 128.

Displayed error message	Meaning and action
INTERNAL ERROR (xxxxxxxxxx)	An error due to an internal factor occurred. Details on the error are displayed in parentheses. Action: Contact the FANUC service center, and report the displayed message correctly.

Error messages displayed during flash ROM I/O operation

Displayed error message	Meaning and action
NOT IN EMG STOP MODE	The system is not in the emergency stop state. Action: Place the system in the emergency stop state.
INVALID LADDER PROGRAM	The transfer program is illegal. Action: Check the program.
DIFFERENCE FOUND	A file comparison detected a mismatch.
FLASH ROM IS LOCKED BY OTHER FUNCTION.	Another PMC user is using the flash ROM. Action: Wait until the PMC user completes processing, then retry.
INTERNAL ERROR (xxxxxxxxxx)	An error due to an internal factor occurred. Details on the error are displayed in parentheses. Action: Contact the FANUC service center, and report the displayed message correctly.

Error messages displayed during floppy cassettes, handy files or other input/output devices I/O operation.

Displayed error message	Meaning and action
BAD PMC PARAMETER FORMAT	Specified file is not PMC parameter format. Action: Specify file of PMC parameter format, or check the contents of the file.
BAD HANDY FILE FORMAT	Specified file is not handy file format. Action: Specify file of handy file format, or check the contents of the file.
UNKNOWN FILE FORMAT	Can not recognize the format of specified file. Action: Specify file of recognizable format such as PMC parameter format, or check the contents of the file.
FILE NAME OR FILE NUMBER IS REQUIRED	Need file name or file number to identify file to read, compare, or delete. Action: Specify file name or file number for the operation.
COMMUNICATION TIMEOUT	Communication with the I/O device has been timeout. Action: Check the communication parameters such as baud rate, and retry to communicate.
I/O DEVICE IS NOT ATTACHED OR IN ERROR STATUS	Any I/O device is not connected, or some errors have occurred in it. Action: Check the power of I/O device is ON. Check the I/O device is connected. Check the cable that connects I/O device with PMC is correct one. If some error has occurred in I/O device, solve it.
RECEIVED BAD DATA: CHECK THE COMMUNICATION PARAMETERS	Invalid data has been received. Action: Check the PMC's communication parameters such as baud rate match the ones of I/O device.
RECEIVED DATA HAS OVERRUN	Too many data have received at once. Action: Check the communication parameters about flow control.
OTHERS FUNCTION IS USING THIS CHANNEL	Others function is using this channel. Action: Use the other channel, or stop the function.
BAD COMMUNICATION PARAMETER	Setting parameters of communication are not correct. Action: Check the communication parameters such as baud rate.

Displayed error message	Meaning and action
OTHER FUNCTION IS USING I/O	Others function such us FANUC LADDER-IIIC is using I/O function. Action: Wait until function that using I/O function do finish, or stop the function.
SEQUENCE PROGRAM IS IN USE BY ONLINE FUNCTION	Can not input/output of sequence program, because On-line function is using sequence program. Action: Wait until On-line function, do finish the using I/O function. In general, both of I/O function and On-line function should not be used at the same time.
THIS FUNCTION IS PROTECTED	Can not input/output PMC's data because the general setting screen of PMC protects them. Action: Cancel PMC's data protection on the general setting screen. Refer to “Programmer Protection Function” in Subsection 6.4.4.
WRONG MODE	Can not output the PMC parameters because mode of PMC and CNC is not correct. Action: Change the CNC to MDI mode or change the ladder to the stop state.
NOT EMG STOP	The system is not in the emergency stop state. Action: Place the system in the emergency stop state.
WRITE PROTECT	Can not input the PMC parameters because PWE of CNC parameter is 0. Action: Change “PWE” of CNC parameters to 1.

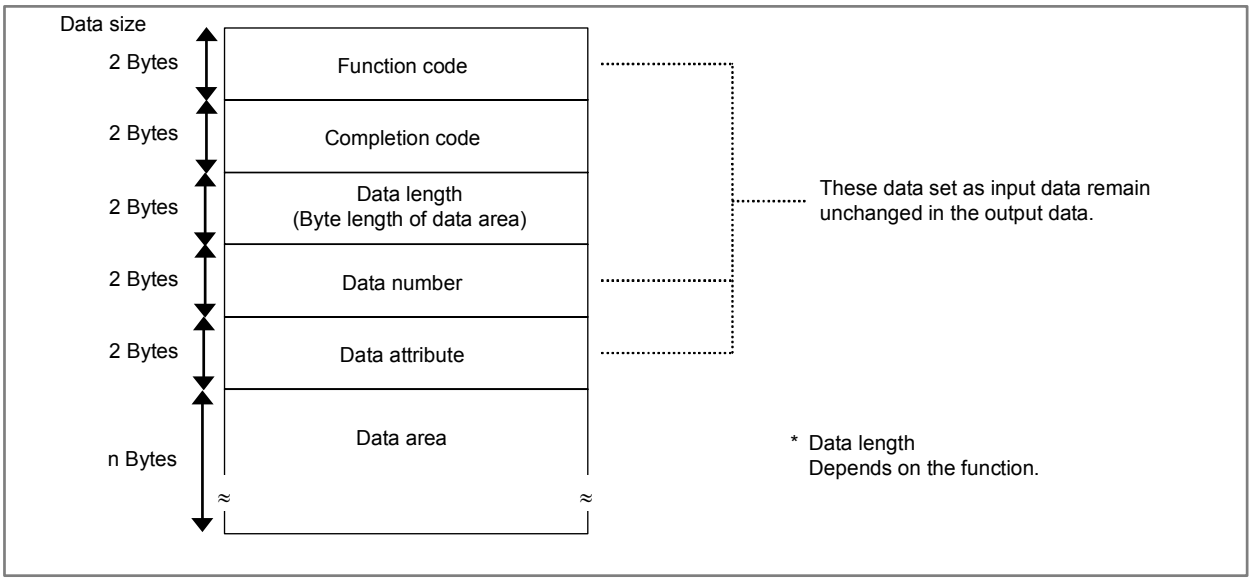
APPENDIX

A WINDOW FUNCTIONS

This chapter describes the functions that can be executed with the PMC_WINDOW functional instructions, as well as the formats and other details of the control data to be set for executing these functions.

A.1 FORMATS OF CONTROL DATA

Input and output control data has the following structure.



(1) In the explanation of the window functions below, minuses (-) in the data structure fields indicate that input data need not be set in these fields or that output data in these fields is not significant.
(2) All data is in binary unless otherwise specified.
(3) All data block lengths and data lengths are indicated in bytes.
(4) Output data is valid only when window processing terminates normally.
(5) Output data always includes one of the following completion codes. Note, however, that all of the completion codes listed are not always provided for each function.

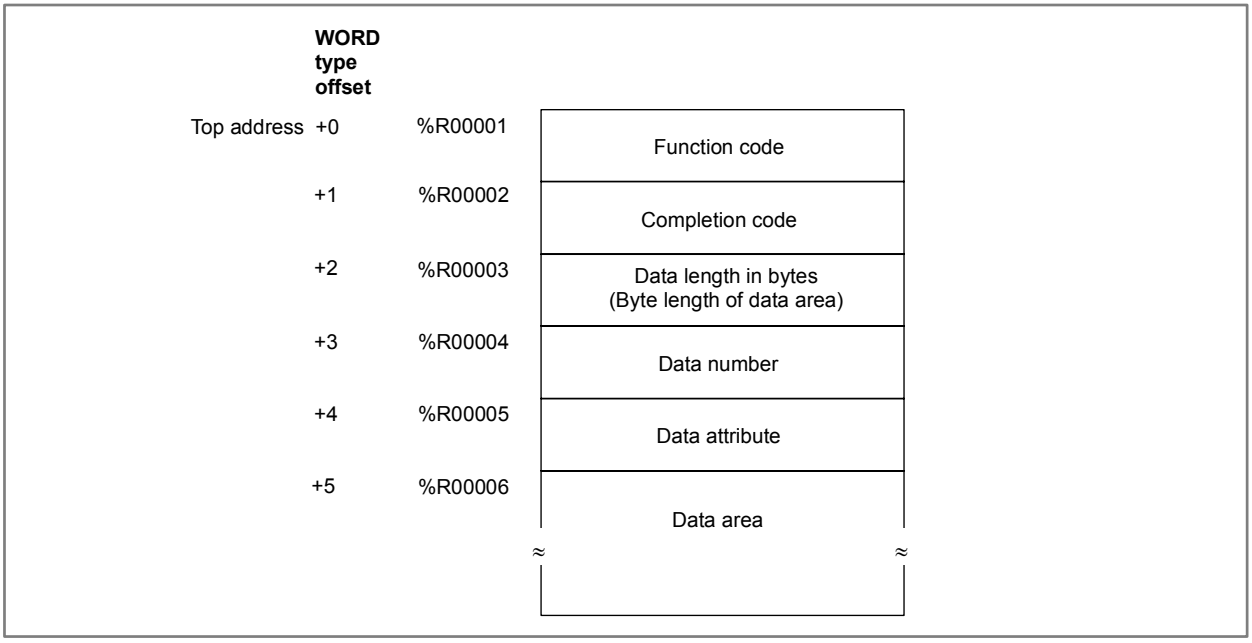
Completion code	Meaning
0	Normal termination
1	Error (invalid function code)
2	Error (invalid data block length)
3	Error (invalid data number)
4	Error (invalid data attribute)
5	Error (invalid data)
6	Error (necessary option missing)
7	Error (write-protected)

A.1.1 The Note about the Address Used for Control Data

There are WORD type and BOOL type in the address of PMC-SD7. When using these for the address for control data structure sets up the starting address as follows.

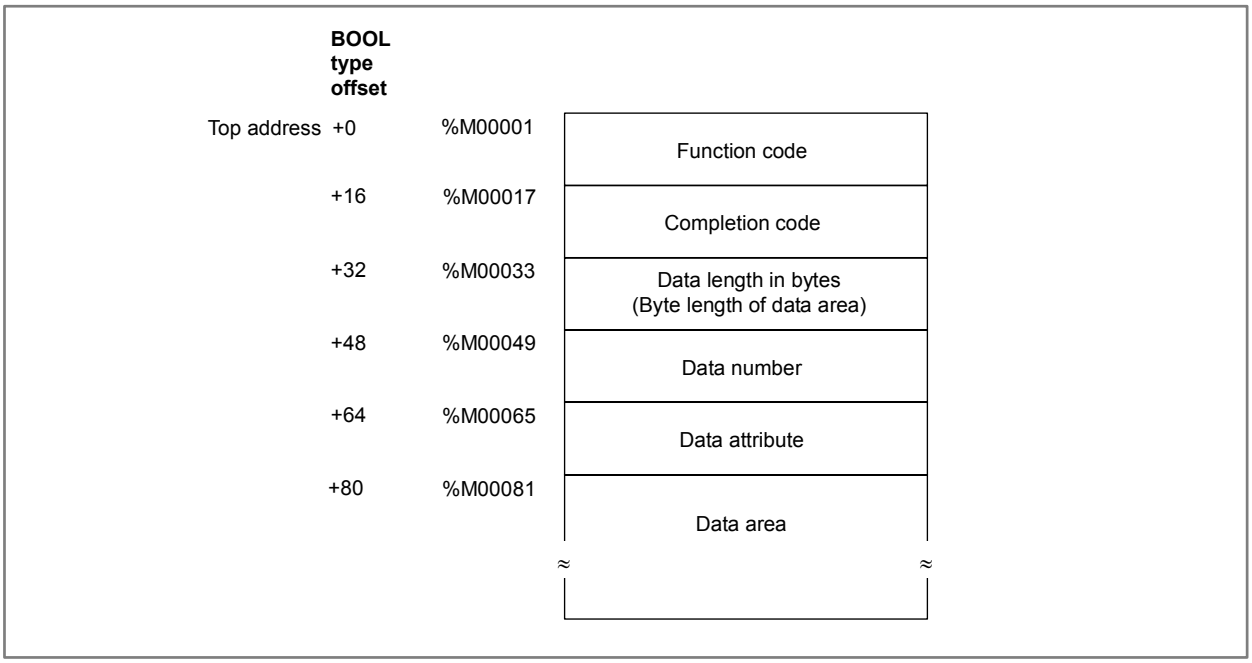
Example of WORD type address:

When the top address is %R00001, address assigned for each data in the control data structure is as follows.



Example of BOOL type address:

When the top address is %M00001, address assigned for each data in the control data structure is as follows.



NOTE

In order to use BOOL type address as BYTE type, the starting BOOL type address must be “modulo 8 plus 1”, in order words, 1, 9, 17, 25, 33, 41, etc.
Farther, starting address at word boundary, in other word “modulo 16 plus 1” such as 1, 17, 33, etc, is strongly recommended for the performance of execution.

A.2 LOW-SPEED RESPONSE AND HIGH-SPEED RESPONSE

There are two types of window function - one executed at high speed and the other executed at low speed.

TYPE	Number of scans to be executed until the window instruction is completed
LOW	TWO SCAN TIMES OR MORE (Depends on the CNC processing priority and operation status.)
HIGH	1 SCAN TIME

In case of a low-speed response, the data is read or written by the control between CNC and PMC. When using the low-speed response window function, set EN to 0 immediately after the data transfer end data (Q) is set to 1 (interlock) for the window instruction. For details, see "CAUTION" below. In a high-speed response, it is not necessity for take the interlock because the data is directly read.

⚠CAUTION

The window instruction of a low-speed response is controlled exclusively with the other window instructions of low-speed response.

Therefore, when the data is read or written continuously, it is necessary to clear EN of the functional instruction to 0 once when the completion information (Q) become 1.

It does not work about EN=1 of the other window instructions of low-speed response such as Q=1 and EN=1 of the window instruction of a low-speed response.

The window instruction of a high-speed response is not exclusively controlled like a low-speed response. Therefore, when the data is read or written continuously, you need not make EN=0.

A.2.1 Note on the Programming of a Low-speed Response Window Instruction

If a low-speed response window instruction is programmed to keep its EN condition to on for no apparent reason, it may result in the ladder program taking a long time to stop or not being able to stop at all. If the ladder program does not stop, all operations aimed at changing the ladder program will take longer to end or will never end.
To avoid such problems, when you code a ladder program using functional instructions, you need to design the ladder structure based on a thorough understanding of the control conditions of the individual instructions you use.

If the ladder program takes long to stop or does not stop for any of these reasons, the following operations will be affected.

1. Stopping the ladder program using a soft key on the screen
2. Reading a new ladder program from a memory card or other medium, by using the DATA I/O screen
3. Updating the ladder program with changes made using the FANUC LADDER-IIIC.

If any of the above phenomena occurs, the functional instruction causing the problem needs to be fixed. For information about how to fix the problem, see Section 4.11 or 5.10.

A.3 LIST OF WINDOW FUNCTIONS

Function group order

Group		Description	Function code	Response	R/W
CNC information (Section A.4)	1	Reading CNC system information	0	High-speed	R
	2	Reading a tool offset	13	High-speed	R
	3	Write a tool offset	14	Low-speed	W
	4	Reading a workpiece origin offset value	15	High-speed	R
	5	Writing a workpiece origin offset value	16	Low-speed	W
	6	Reading a parameter	17, 154	High-speed	R
	7	Writing a parameter	18	Low-speed	W
	8	Reading setting data	19, 155	High-speed	R
	9	Writing setting data	20	Low-speed	W
	10	Reading a custom macro variable	21	High-speed	R
	11	Writing a custom macro variable	22	Low-speed	W
	12	Reading the CNC alarm status	23	High-speed	R
	13	Reading the current program number	24	High-speed	R
	14	Reading the current sequence number	25	High-speed	R
	15	Reading modal data	32	High-speed	R
	16	Reading diagnosis data	33, 156	High-speed	R
	17	Reading the P-code macro variable	59	High-speed	R
	18	Writing the P-code macro variable	60	Low-speed	W
	19	Reading CNC status information	76	High-speed	R
	20	Reading the current program number (8-digit program numbers)	90	High-speed	R
	21	Entering data on the program check screen	150	Low-speed	W
	22	Reading clock data (date and time)	151	High-speed	R
	23	Specifying the Number of the Program for I/O Link	194	Low-speed	W
Axis information (Section A.5)	1	Reading the actual velocity of controlled axes	26	High-speed	R
	2	Reading the absolute position (absolute coordinates) of controlled axes	27	High-speed	R
	3	Reading the machine position (machine coordinates) of controlled axes	28	High-speed	R
	4	Reading a skip position (stop coordinates of skip operation (G31)) of controlled axes	29	High-speed	R
	5	Reading the servo delay for controlled axes	30	High-speed	R
	6	Reading the acceleration/deceleration delay on controlled axes	31	High-speed	R
	7	Reading the feed motor load current value (A/D conversion data)	34	High-speed	R
	8	Reading the actual spindle speed	50	High-speed	R
	9	Reading the relative position on a controlled axis	74	High-speed	R
	10	Reading the remaining travel	75	High-speed	R
	11	Reading actual spindle speeds	138	High-speed	R
	12	Entering torque limit data for the digital servo motor	152	Low-speed	W
	13	Reading load information of the spindle motor (serial interface)	153	High-speed	R
	14	Reading the servo data of the control axes	207	High-speed	R
	15	Reading the estimate disturbance torque data	211	High-speed	R

Group		Description	Function code	Response	R/W
Axis information (Section A.5)	16	Reading Fine Torque Sensing Data (Statistical Calculation Results)	226	High-speed	R
	17	Reading Fine Torque Sensing Data (Store Data)	232	High-speed	R
	18	Presetting the relative coordinate	249	Low-speed	W
Tool life management functions (Section A.6)	1	Reading the tool life management data (tool group number)	38	High-speed	R
	2	Reading tool life management data (number of tool groups)	39	High-speed	R
	3	Reading tool life management data (number of tools)	40	High-speed	R
	4	Reading tool life management data (tool life)	41	High-speed	R
	5	Reading tool life management data (tool life counter)	42	High-speed	R
	6	Reading tool life management data (tool length compensation number (1): Tool number)	43	High-speed	R
	7	Reading tool life management data (tool length compensation number (2): Tool order number)	44	High-speed	R
	8	Reading tool life management data (cutter compensation number (1): Tool number)	45	High-speed	R
	9	Reading tool life management data (cutter compensation number (2): Tool order number)	46	High-speed	R
	10	Reading tool life management data (tool information (1): Tool number)	47	High-speed	R
	11	Reading tool life management data (tool information (2): Tool order number)	48	High-speed	R
	12	Reading tool life management data (tool number)	49	High-speed	R
	13	Reading the tool life management data (tool life counter type)	160	High-speed	R
	14	Registering tool life management data (tool group)	163	Low-speed	W
	15	Writing tool life management data (tool life)	164	Low-speed	W
	16	Writing tool life management data (tool life counter)	165	Low-speed	W
	17	Writing tool life management data (tool life counter type)	166	Low-speed	W
	18	Writing tool life management data (tool length compensation number (1): Tool number)	167	Low-speed	W
	19	Writing tool life management data (tool length compensation number (2): Tool order number)	168	Low-speed	W
	20	Writing tool life management data (cutter compensation number (1): Tool number)	169	Low-speed	W
Tool life management functions (Section A.6)	21	Writing tool life management data (cutter compensation number (2): Tool order number)	170	Low-speed	W
	22	Writing tool life management data (tool information (1): Tool number)	171	Low-speed	W
	23	Writing the tool management data (tool information (2): Tool order number)	172	Low-speed	W
	24	Writing tool life management data (tool number)	173	Low-speed	W
	25	Reading the tool life management data (tool group No.) (8-digit tool number)	200	High-speed	R
	26	Reading tool life management data (tool information (1): Tool number) (8-digit tool number)	201	High-speed	R
	27	Registering tool life management data (tool group number) (8-digit tool number)	202	Low-speed	W

Group		Description	Function code	Response	R/W
Tool life management functions (Section A.6)	28	Reading tool life management data (tool length compensation number (1): Tool number) (8-digit tool number)	227	High-speed	R
	29	Reading tool life management data (cutter compensation number (1): Tool number) (8-digit tool number)	228	High-speed	R
	30	Writing tool life management data (tool length compensation number (1): Tool number) (8-digit tool number)	229	Low-speed	W
	31	Writing tool life management data (cutter compensation number (1): Tool number) (8-digit tool number)	230	Low-speed	W
	32	Writing the tool life management data (tool information (1): Tool number) (8-digit tool number)	231	Low-speed	W
	33	Deleting tool life management data (tool group)	324	Low-speed	W
	34	Deleting tool life management data (tool data)	325	Low-speed	W
	35	Clearing tool life management data (tool life counter and tool information)	326	Low-speed	W
	36	Writing tool life management data (arbitrary group number)	327	Low-speed	W
	37	Writing tool life management data (remaining tool life)	328	Low-speed	W
Tool management functions (Section A.7)	1	Exchange of tool management data number in the magazine management table	329	Low-speed	W
	2	Search of empty pot	330	Low-speed	R
	3	New-register of a tool management data	331	Low-speed	W
	4	Writing a tool management data	332	Low-speed	W
	5	Deletion of a tool management data	333	Low-speed	W
	6	Reading a tool management data	334	Low-speed	R
	7	Writing each tool data	335	Low-speed	W
	8	Search of tool management data	366	Low-speed	R
	9	Shift of tool management data	367	Low-speed	W
	10	Reading a decimal point of the customizing data	392	Low-speed	R
	11	Searching for a free pot (oversize tools supported)	397	Low-speed	R
	12	Reading a total life data	409	Low-speed	R

*1 Function codes that have R in the R/W column are window read functions specifiable with the PMC_WINDOW function command. Function codes that have W in the R/W column are window write functions specifiable with the PMC_WINDOW function command.

*2 Functions of "High-speed" in their Response field can read or write data immediately upon request. On the other hand, functions of "Low-speed" in their Response field need to request the CNC to read or write data and receiving response from CNC completes the request.

⚠CAUTION

To read or write data for the second tool post (HEAD2) in the TT series, add 1000 to the function code number.

To read or write data for the third path in three-path control CNC, add 2000 to the function code number.

Function code order

Function code	Description	Response	R/W
0	Reading CNC system information	High-speed	R
13	Reading a tool offset	High-speed	R
14	Writing a tool offset	Low-speed	W
15	Reading a workpiece origin offset value	High-speed	R
16	Writing a workpiece origin offset value	Low-speed	W
17	Reading a parameter	High-speed	R
18	Writing a parameter	Low-speed	W
19	Reading setting data	High-speed	R
20	Writing setting data	Low-speed	W
21	Reading a custom macro variable	High-speed	R
22	Writing a custom macro variable	Low-speed	W
23	Reading the CNC alarm status	High-speed	R
24	Reading the current program number	High-speed	R
25	Reading the current sequence number	High-speed	R
26	Reading the actual velocity of controlled axes	High-speed	R
27	Reading the absolute position (absolute coordinates) of controlled axes	High-speed	R
28	Reading the machine position (machine coordinates) of controlled axes	High-speed	R
29	Reading a skip position (stop coordinates of skip operation (G31)) of controlled axes	High-speed	R
30	Reading the servo delay for controlled axes	High-speed	R
31	Reading the acceleration/deceleration delay on controlled axes	High-speed	R
32	Reading modal data	High-speed	R
33	Reading diagnosis data	High-speed	R
34	Reading the feed motor load current value (A/D conversion data)	High-speed	R
38	Reading the tool life management data (tool group number)	High-speed	R
39	Reading tool life management data (number of tool groups)	High-speed	R
40	Reading tool life management data (number of tools)	High-speed	R
41	Reading tool life management data (tool life)	High-speed	R
42	Reading tool life management data (tool life counter)	High-speed	R
43	Reading tool life management data (tool length compensation number (1): Tool number)	High-speed	R
44	Reading tool life management data (tool length compensation number (2): Tool order number)	High-speed	R
45	Reading tool life management data (cutter compensation number (1): Tool number)	High-speed	R
46	Reading tool life management data (cutter compensation number (2): Tool order number)	High-speed	R
47	Reading tool life management data (tool information (1): Tool number)	High-speed	R
48	Reading tool life management data (tool information (2): Tool order number)	High-speed	R
49	Reading tool life management data (tool number)	High-speed	R
50	Reading the actual spindle speed	High-speed	R
59	Reading the P-code macro variable	High-speed	R
60	Writing the P-code macro variable	Low-speed	W
74	Reading the relative position on a controlled axis	High-speed	R
75	Reading the remaining travel	High-speed	R
76	Reading CNC status information	High-speed	R
90	Reading the current program number (8-digit program numbers)	High-speed	R
138	Reading actual spindle speeds	High-speed	R
150	Entering data on the program check screen	Low-speed	W

Function code	Description	Response	R/W
151	Reading clock data (date and time)	High-speed	R
152	Entering torque limit data for the digital servo motor	Low-speed	W
153	Reading load information of the spindle motor (serial interface)	High-speed	R
154	Reading a parameter	High-speed	R
155	Reading setting data	High-speed	R
160	Reading the tool life management data (tool life counter type)	High-speed	R
163	Registering the tool life management data (tool group)	Low-speed	W
164	Writing the tool life management data (tool life)	Low-speed	W
165	Writing the tool life management data (tool life counter)	Low-speed	W
166	Writing the tool life management data (tool life counter type)	Low-speed	W
167	Writing the tool life management data (tool length compensation number (1): Tool number)	Low-speed	W
168	Writing the tool life management data (tool length compensation number (2): Tool order number)	Low-speed	W
169	Writing the tool life management data (cutter compensation number (1): Tool number)	Low-speed	W
170	Writing the tool life management data (cutter compensation number (2): Tool order number)	Low-speed	W
171	Writing the tool life management data (tool information (1): Tool number)	Low-speed	W
172	Writing the tool management data (tool condition (2): Tool order number)	Low-speed	W
173	Writing the tool life management data (tool number)	Low-speed	W
194	Specifying the Number of the Program for I/O Link	Low-speed	W
200	Reading the tool life management data (tool group number) (8-digit tool number)	High-speed	R
201	Reading tool life management data (tool information (1): Tool number) (8-digit tool number)	High-speed	R
202	Registering tool life management data (tool group number) (8-digit tool number)	Low-speed	W
207	Reading the servo data of the control axes	High-speed	R
211	Reading the estimate disturbance torque data	High-speed	R
226	Reading Fine Torque Sensing Data (Statistical Calculation Results)	High-speed	R
227	Reading tool life management data (tool length compensation number (1): Tool number) (8-digit tool number)	High-speed	R
228	Reading tool life management data (cutter compensation number (1): Tool number) (8-digit tool number)	High-speed	R
229	Writing tool life management data (tool length compensation number (1): Tool number) (8-digit tool number)	Low-speed	W
230	Writing tool life management data (cutter compensation number (1): Tool number) (8-digit tool number)	Low-speed	W
231	Writing the tool life management data (tool information (1): Tool number) (8-digit tool number)	Low-speed	W
232	Reading Fine Torque Sensing Data (Store Data)	High-speed	R
249	Presetting the relative coordinate	Low-speed	W
324	Deleting the tool life management data (tool group)	Low-speed	W
325	Deleting the tool life management data (tool data)	Low-speed	W
326	Deleting the tool life management data (tool life counter and tool information)	Low-speed	W
327	Writing the tool life management data (arbitrary group number)	Low-speed	W
328	Writing the tool life management data (remaining tool life)	Low-speed	W
329	Exchange of tool management data number in the magazine management table	Low-speed	W
330	Search of empty pot	Low-speed	R
331	New-register of a tool management data	Low-speed	W
332	Writing a tool management data	Low-speed	W
333	Deletion of a tool management data	Low-speed	W

Function code	Description	Response	R/W
334	Reading a tool management data	Low-speed	R
335	Writing each tool data	Low-speed	W
366	Search of tool management data	Low-speed	R
367	Shift of tool management data	Low-speed	W
392	Reading a decimal point of the customizing data	Low-speed	R
397	Searching for a free pot (oversize tools supported)	Low-speed	R
409	Reading a total life data	Low-speed	R

*1 Function codes that have R in the R/W column are window read functions specifiable with the PMC_WINDOW function command. Function codes that have W in the R/W column are window write functions specifiable with the PMC_WINDOW function command.

*2 Functions of "High-speed" in their Response field can read or write data immediately upon request. On the other hand, functions of "Low-speed" in their Response field need to request the CNC to read or write data and receiving response from CNC completes the request.

⚠CAUTION

To read or write data for the second tool post (HEAD2) in the TT series, add 1000 to the function code number.

To read or write data for the third path in three-path control CNC, add 2000 to the function code number.

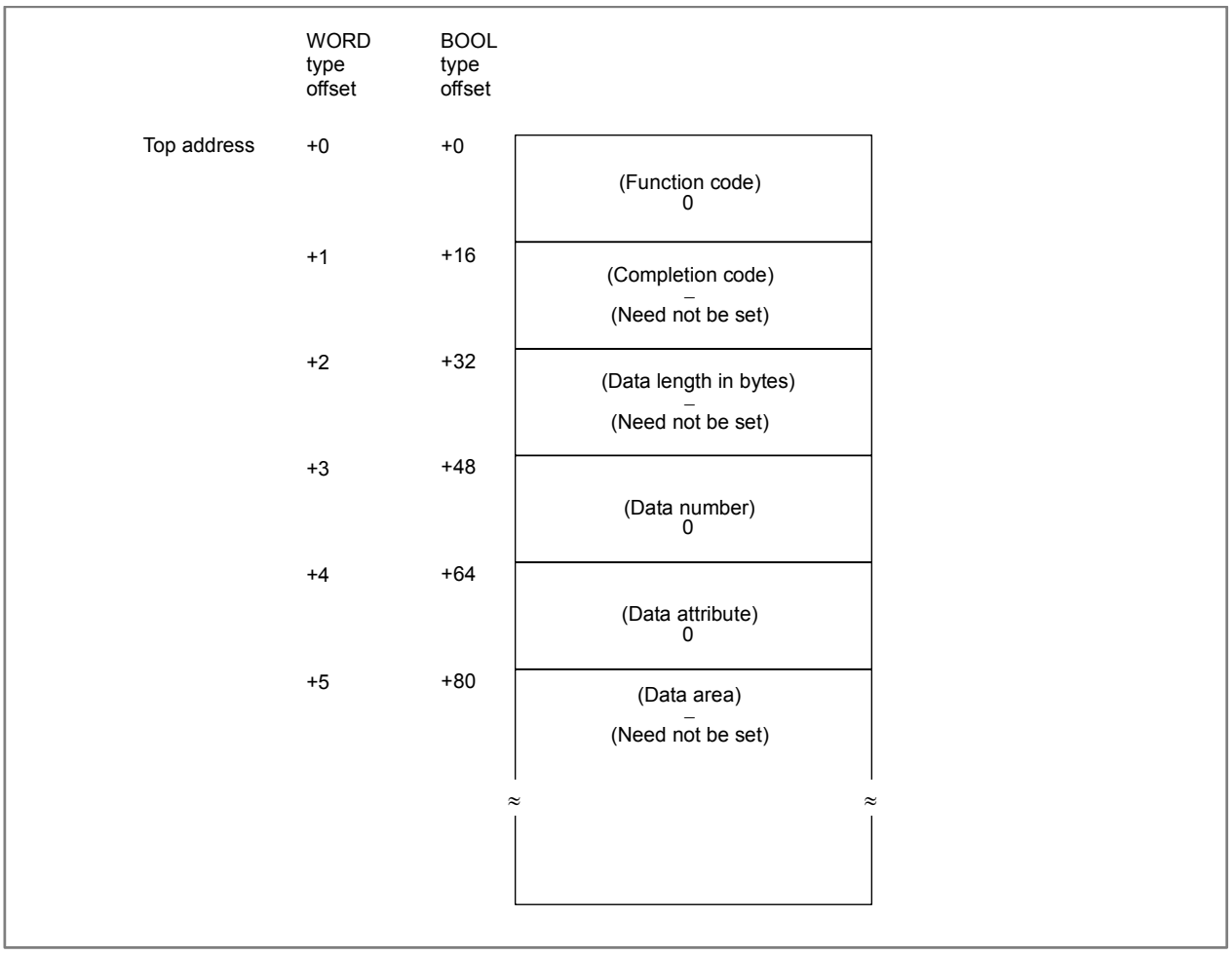
A.4 CNC INFORMATION

A.4.1 Reading CNC System Information (High-speed Response)

[Description]

The system information specific to the CNC can be read including the CNC type (e.g., series name like 16), the distinction between the machining center system (M) and the lathe system (T) for each CNC path, the ROM series and edition of the CNC system software, and the number of axes to be controlled for each CNC path.

[Input data [Input data structure]



[Completion codes]

0: CNC system information has been read normally.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 0	
	+1	+16	(Completion code) 0 (Always terminates normally.)	
	+2	+32	(Data length in bytes) 14	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	CNC series name (2 bytes)	ASCII characters (30)
	+6	+96	Machine type M/T (2 bytes)	ASCII characters (M, T)
	+7	+112	ROM series of CNC system software (4 bytes)	ASCII characters (B 0 0 0 1, . . .)
	+8	+144	ROM version of CNC system software (4 bytes)	ASCII characters (0 0 0 1, 0 0 0 2, . . .)
	+11	+176	Number of axes to be controlled for the specified CNC path (2 bytes)	ASCII characters (2, 3, 4, . . .)

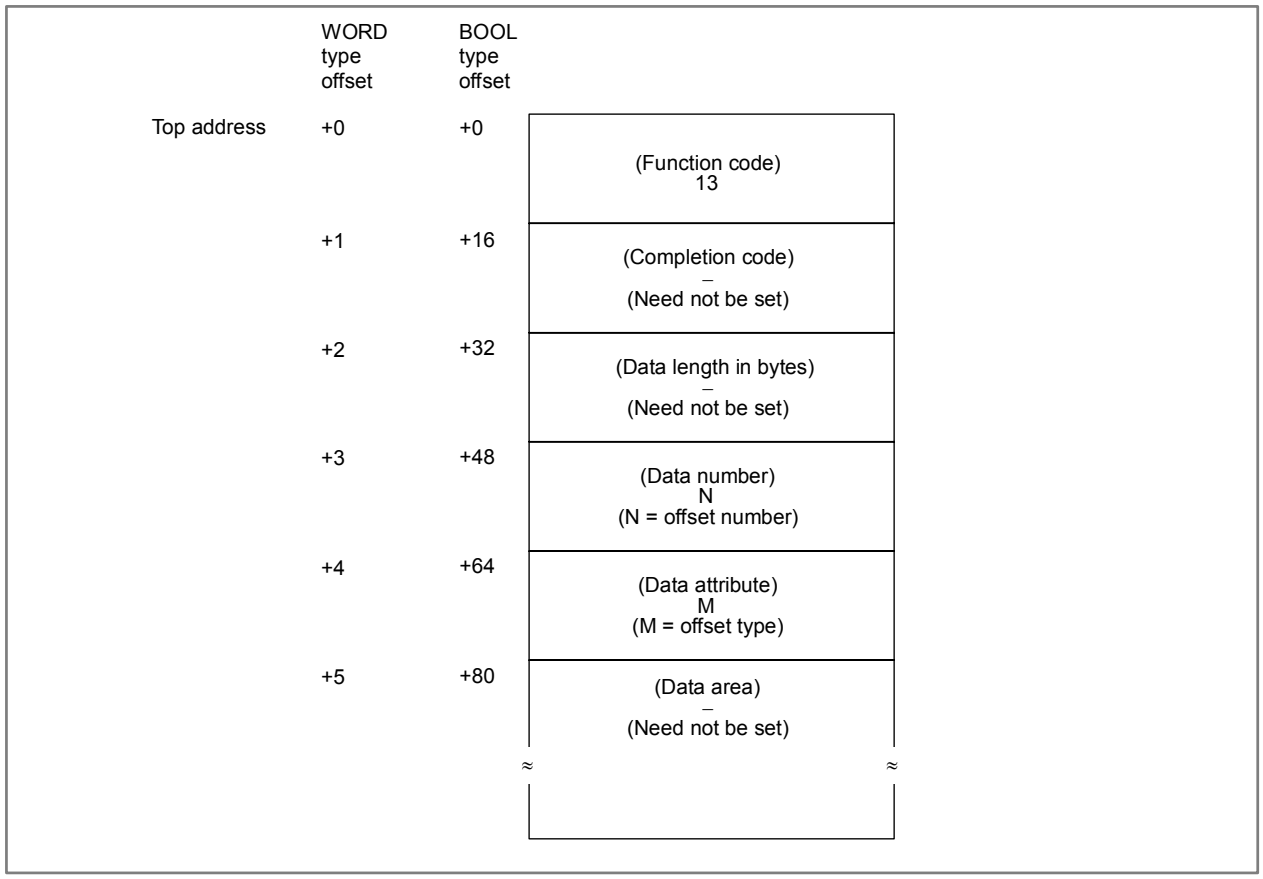
NOTE
Data is stored from the upper digit in each lower byte.

A.4.2 Reading a Tool Offset (High-speed Response)

[Description]

A tool offset value recorded in the CNC can be read.
Wear offset data, geometric offset data, cutter compensation data, and tool length offset data can be read as a tool offset.

[Input data structure]



(a) Offset types (for machining centers)

	Cutter	Tool length
Wear	0	2
Geometric	1	3

- If the type of tool offset need not be specified, enter 0.

(b) Offset types (for lathes)

	X axis	Z axis	Tool tip R	Virtual tool tip	Y axis
Wear	0	2	4	6	8
Geometric	1	3	5	7	9

[Completion codes]

0: The tool offset has been read normally.
3: The offset number specified for reading is invalid. (This completion code is returned when the specified offset number data is not from 1 to the maximum number of offsets.)
4: There are mistakes in the data attribute that specifies the type of the offset to be read.
6: For the offset number specified for reading, the additional tool offset number option is required, but it is missing.
In addition, for the type of the offset specified for reading, an additional tool function option is required, but it is missing.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 13	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (Normally set to 4) (L: Byte length of offset value)	
	+3	+48	(Data number) N (N = offset number)	
	+4	+64	(Data attribute) M (M = offset type)	Value
	+5	+80	Tool offset value	Signed binary (A negative value is represented in 2's complement.) Upper 3 bytes are always "0" for virtual tool tip.

[Output data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

A.4.3 Writing a Tool Offset (Low-speed Response)

[Description]

The tool offset value can be directly written into the CNC.
Wear offset data, geometric offset data, cutter compensation data, and tool length offset data can be written as a tool offset.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 14	
	+1	+16	(Completion code) – (Need not to be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = offset number)	
	+4	+64	(Data attribute) M (M = offset type)	Value
	+5	+80	Tool offset value	Signed binary (A negative value is represented in 2's complement.) Upper 3 bytes are always "0" for virtual tool tip.

(a) Offset types (for machining centers)

	Cutter	Tool length
Wear	0	2
Geometric	1	3

- If the type of tool offset need not be specified, enter 0.

(b) Offset types (for lathes)

	X axis	Z axis	Tool tip R	Virtual tool tip	Y axis
Wear	0	2	4	6	8
Geometric	1	3	5	7	9

[Input data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

[Completion codes]

0: The tool offset has been written normally.
2: The data byte length for the tool offset specified for writing is invalid.
3: The offset number specified for writing is invalid. (This completion code is returned when the specified offset number data is not from 1 to the maximum number of offsets.)
4: There are mistakes in the data attribute that specifies the type of the offset to be written.
6: For the offset number specified for writing, the additional tool offset number option is required, but it is missing.
Also, the tool function option is not added that is necessary for the type of the offset specified for writing.

[Output data structure]

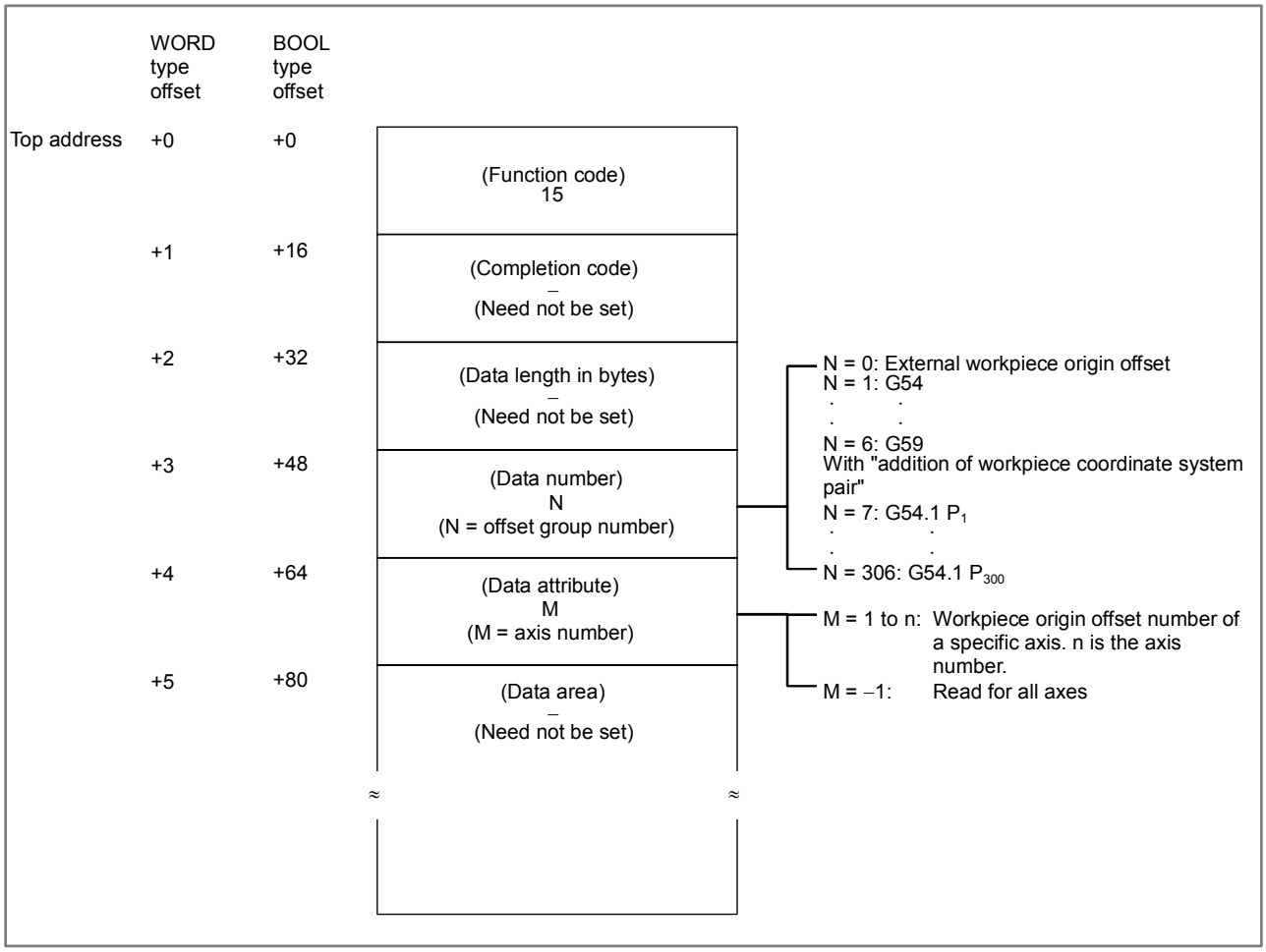
	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 14	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L: Input data)	
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) M (Input data)	Value
	+5	+80	Tool offset value: Input data	Signed binary (A negative value is represented in 2's complement.)

A.4.4 Reading a Workpiece Origin Offset Value (High-speed Response)

[Description]

The workpiece origin offset recorded in the CNC can be read.
A workpiece origin offset is provided for each controlled axis (the first axis to the 8th axis) in the CNC. Either the workpiece origin offset for a specific axis can be read, or the workpiece origin offsets for all axes can be read at one time. If the additional axis option is not provided, however, the workpiece origin offset for the additional axis cannot be read.

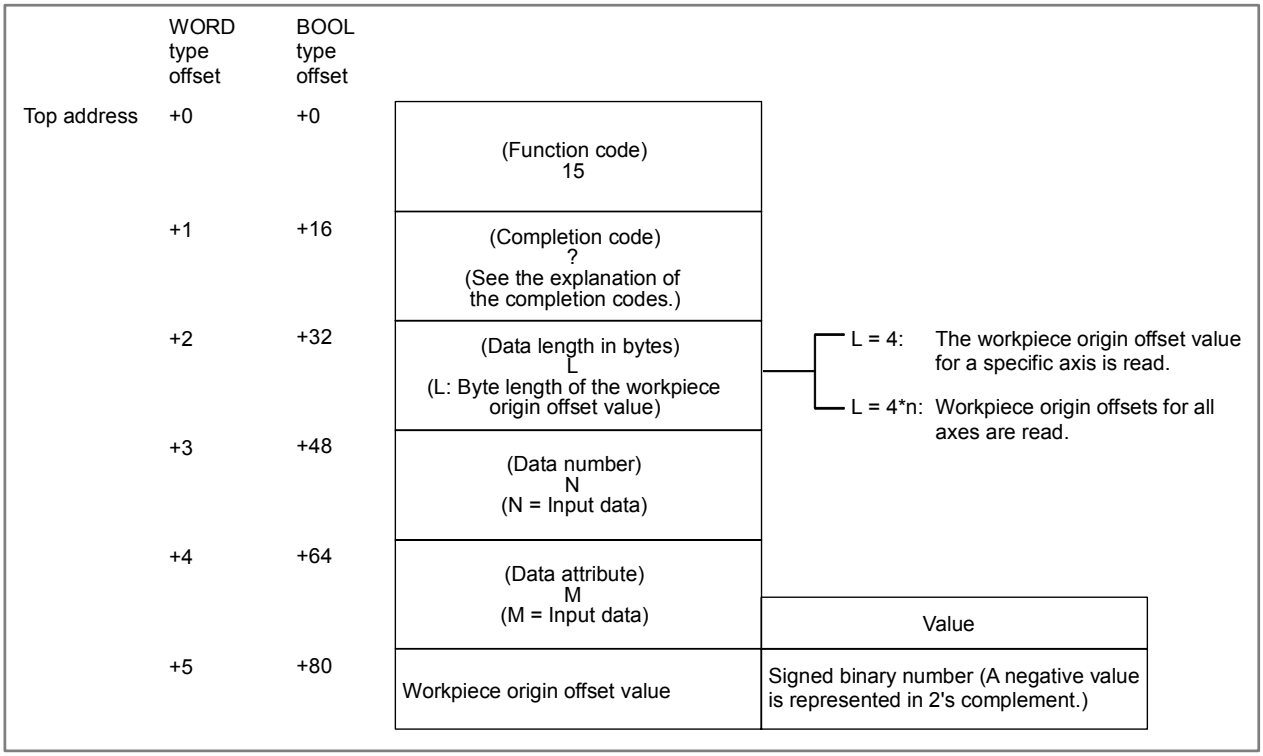
[Input data structure]



[Completion codes]

0: The workpiece origin offset has been read normally.
3: The specified offset number is invalid.
4: The specified axis number is invalid.
6: There is no workpiece coordinate shift option added.

[Output data structure]



[Output data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

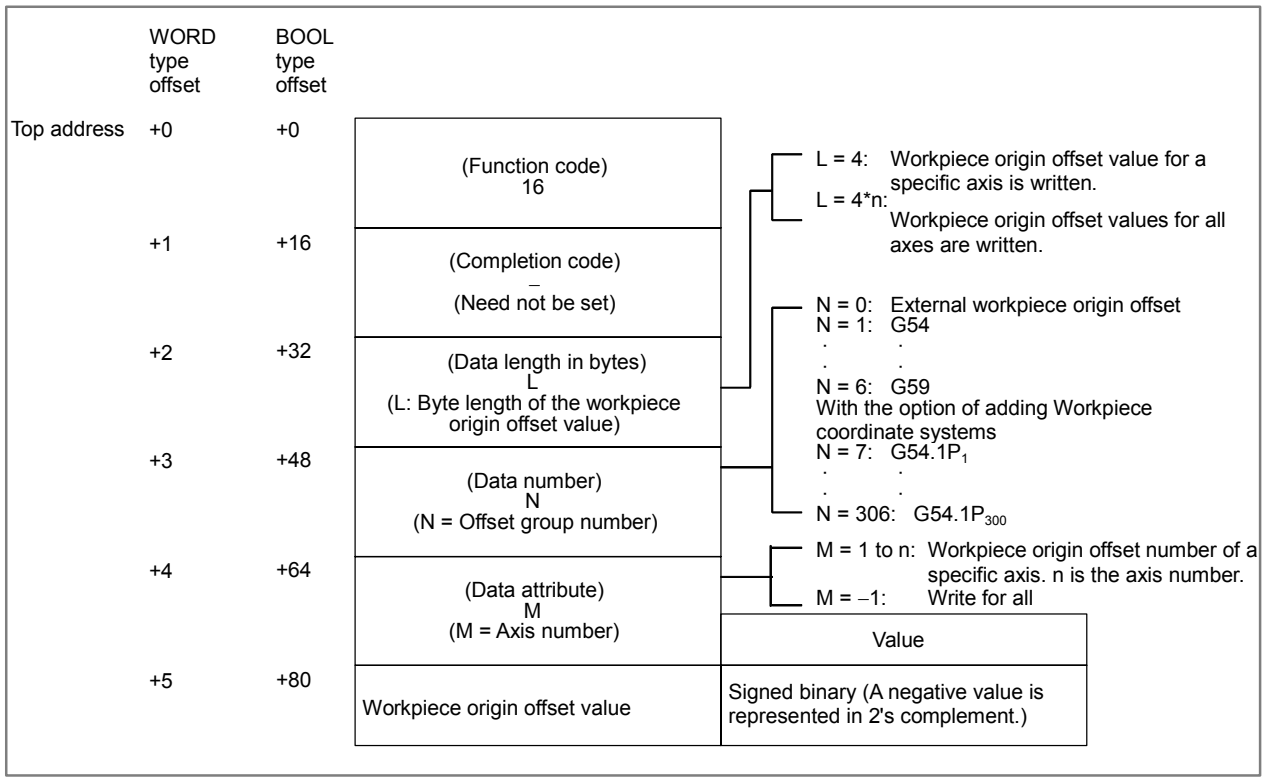
A.4.5 Writing a Workpiece Origin Offset Value (Low-speed Response)

[Description]

Data can be written directly as a workpiece origin offset value in the CNC.

A workpiece origin offset is provided for each controlled axis (the first axis to the 8th axis) in the CNC. Either the workpiece origin offset value for a specific axis can be written, or the workpiece origin offset values for all axes can be written at one time. If the additional axis option is not provided, however, the workpiece origin offset value for the additional axis cannot be written.

[Input data structure]



[Input data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

[Completion codes]

0: The workpiece origin offset has been written normally.
2: The specified data length is invalid.
3: The offset number is invalid.
4: The specified axis number is invalid.
6: There is no workpiece coordinate shift option added.

[Output data structure]

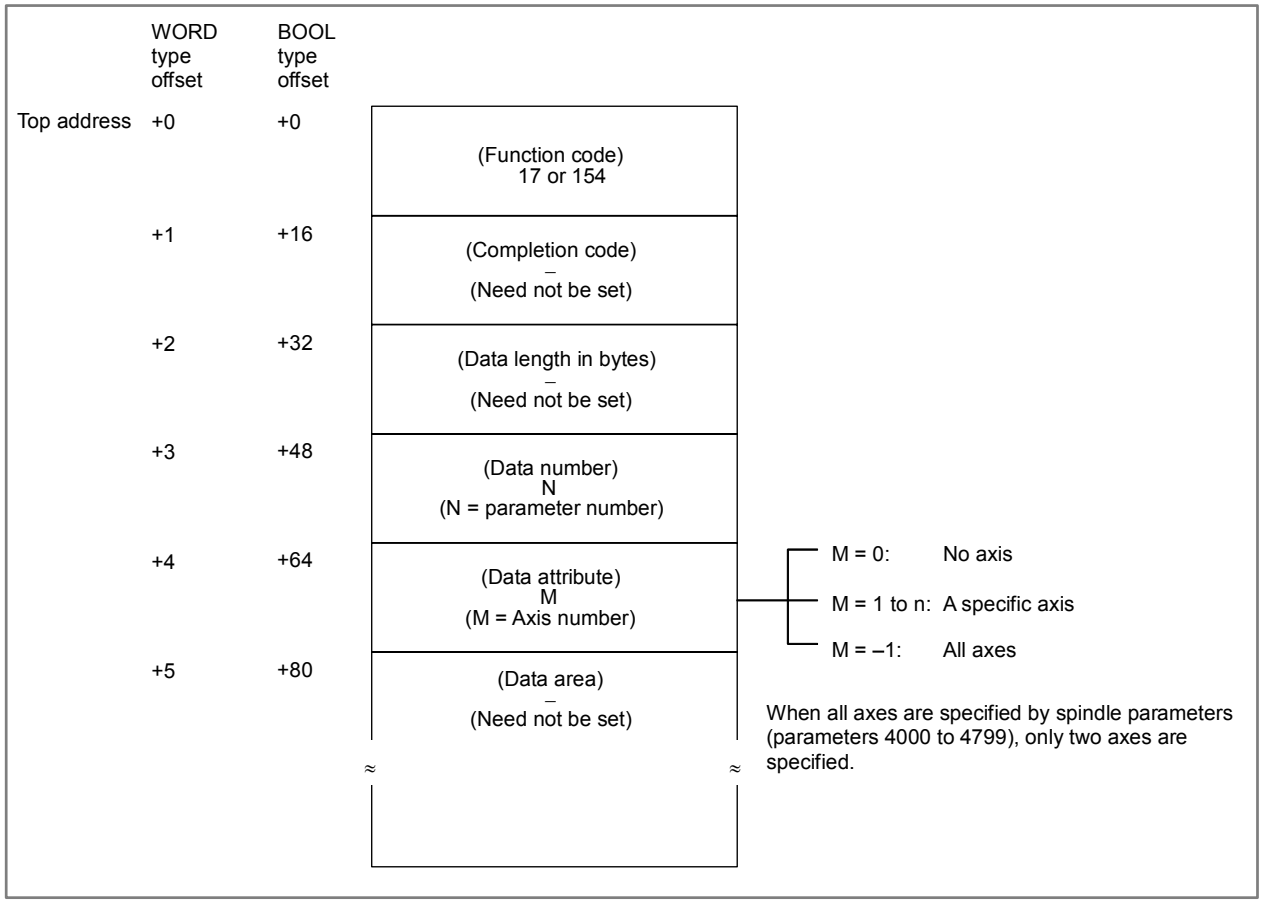
	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 16	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L: Input data)	
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) M (M = Input data)	Value
	+5	+80	Workpiece origin offset value: Input data	Signed binary number (A negative value is represented in 2's complement.)

A.4.6 Reading a Parameter (High-speed Response)

[Description]

Parameter data of the CNC is read by directly accessing the CNC.
There are four types of parameters in the CNC: Bit parameters having a definite meaning for each bit, byte parameters holding 1-byte data, word parameters holding 2-byte data, and double word parameters holding 4-byte data. Therefore, the length of the read data varies according to the parameter number specified.
Note that bit parameters cannot be read in bit units. The eight bits (one byte) for a parameter number must be read at a time.
For axis parameters, data for a specific axis can be read, or data for all axes can be read at a time.
Specify pitch error compensation data in data Nos. 10000 to 11023 (1024 points in total).
For details of parameter data, refer to the Operator's manual of the CNC.

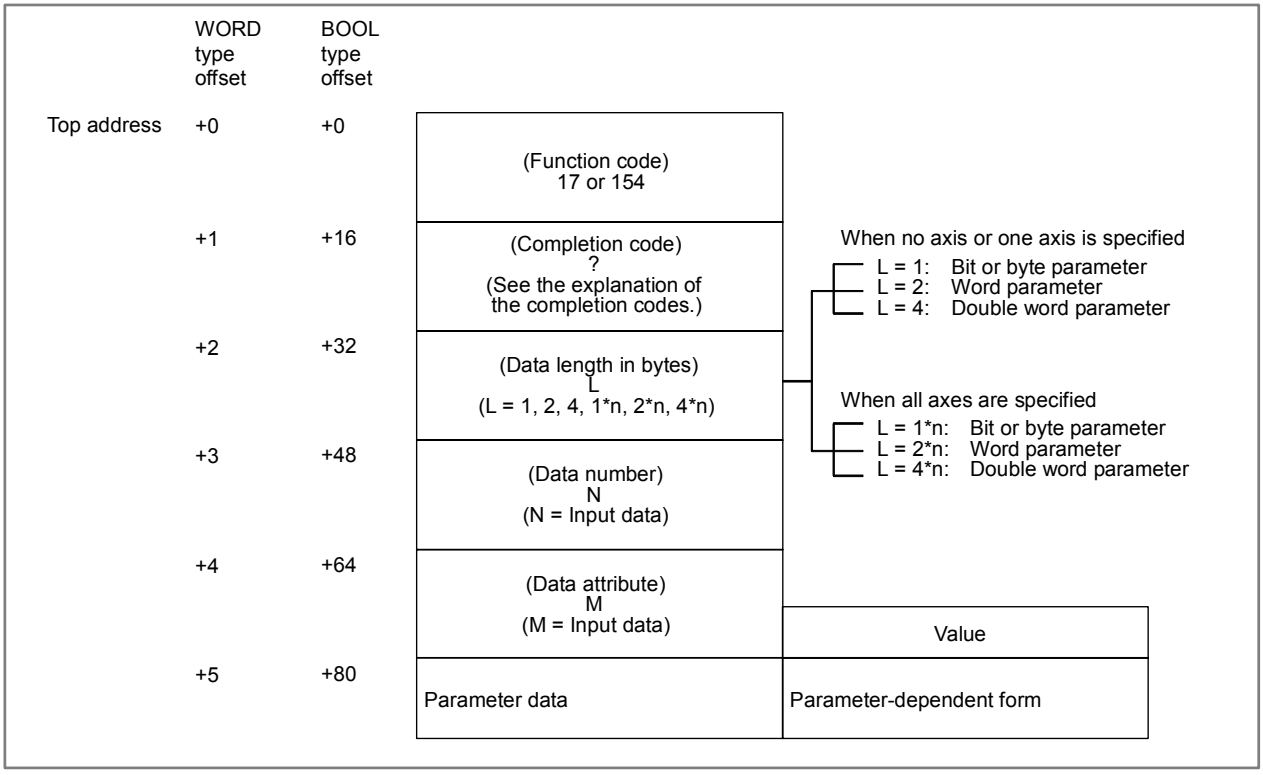
[Input data structure]



[Completion codes]

0: Parameter data has been read normally.
3: The parameter number specified for reading is invalid.
4: The specified data attribute is invalid because it is neither 0, -1, nor a value 1 to n (n is the number of axes).
6: Although a certain option, such as the pitch error compensation option, is required for the data of the parameter number specified for reading, it is not provided.

[Output data structure]



⚠CAUTION
Macro executor parameters 9000 to 9011 cannot be read.

A.4.7 Writing a Parameter (Low-speed Response)

[Description]

Data can be written in a parameter in the CNC.
There are four types of parameters in the CNC: Bit parameters having a definite meaning for each bit, byte parameters holding 1-byte data, word parameters holding 2-byte data, and double word parameters holding 4-byte data. Therefore, the length of the written data varies according to the parameter specified.
Note that bit parameters cannot be written in bit units. The eight bits (one byte) for the parameter number must be written at a time. This means that when a bit needs to be written, the whole data for the corresponding parameter number shall be read first, modify the target bit in the read data, then the data shall be rewritten.
For axis parameters, data for a specific axis can be read, or data for all axes can be read at a time.
For details of parameter data, refer to the Operator's manual of the CNC.
Some parameters cause a P/S alarm 000 when data is written. (The power must be turned off before continuing operation.)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 18	
	+1	+16	(Completion code) – (Need not be set)	When no axis or one axis is specified L = 1: Bit or byte parameter L = 2: Word parameter L = 4: Double word parameter
	+2	+32	(Data length in bytes) L (L = 1, 2, 4, 1*n, 2*n, 4*n)	When all axes are specified L = 1*n: Bit or byte parameter L = 2*n: Word parameter L = 4*n: Double word parameter
	+3	+48	(Data number) N (N = parameter number)	
	+4	+64	(Data attribute) M (M = Axis number)	M = 0: No axis M = 1 to n: A specific axis M = –1: All axes When all axes are specified by spindle parameters (parameters 4000 to 4799), only two axes are specified.
				Value
	+5	+80	Parameter data	Parameter-dependent form

[Completion codes]

0: Parameter data has been written normally.
2: The data byte length of the parameter specified for writing is invalid.
3: The parameter number specified for writing is invalid.
4: The specified data attribute is invalid because it is neither 0, -1, nor a value from 1 to n (n is the number of axes).
6: Although a certain option, such as the pitch error compensation option, is required for the data of the parameter number specified for writing, it is not provided.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 18	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = Input data)	
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) M (M = Input data)	Value
	+5	+80	Parameter data: Input data	Parameter-dependent form

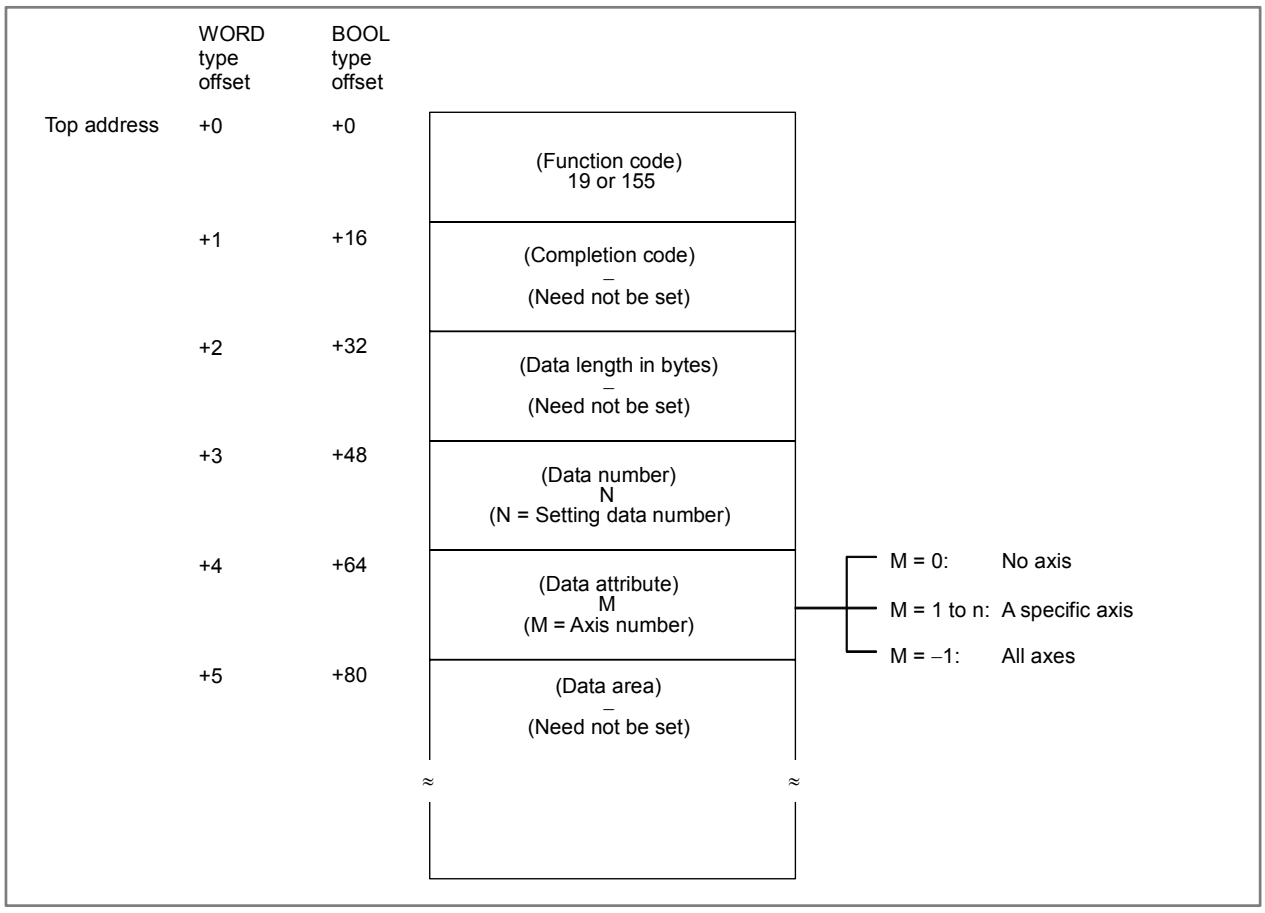
⚠CAUTION
Parameters may not become effective immediately depending on the parameter numbers.

A.4.8 Reading Setting Data (High-speed Response)

[Description]

The setting data in the CNC is read by directly accessing the CNC.
There are four types of setting data in the CNC: Bit setting data having a definite meaning for each bit, byte setting data stored in bytes, word setting data stored in 2-byte units, and double-word setting data stored in 4-byte units. Therefore, the length of the read data varies according to the setting data specified.
Note that bit setting data cannot be read in bit units. The eight bits (one byte) for the setting data number must be read at a time.
For axis parameters, data for a specific axis can be read, or data for all axes can be read at a time.
For details of setting data, refer to the Operator's manual of the CNC.

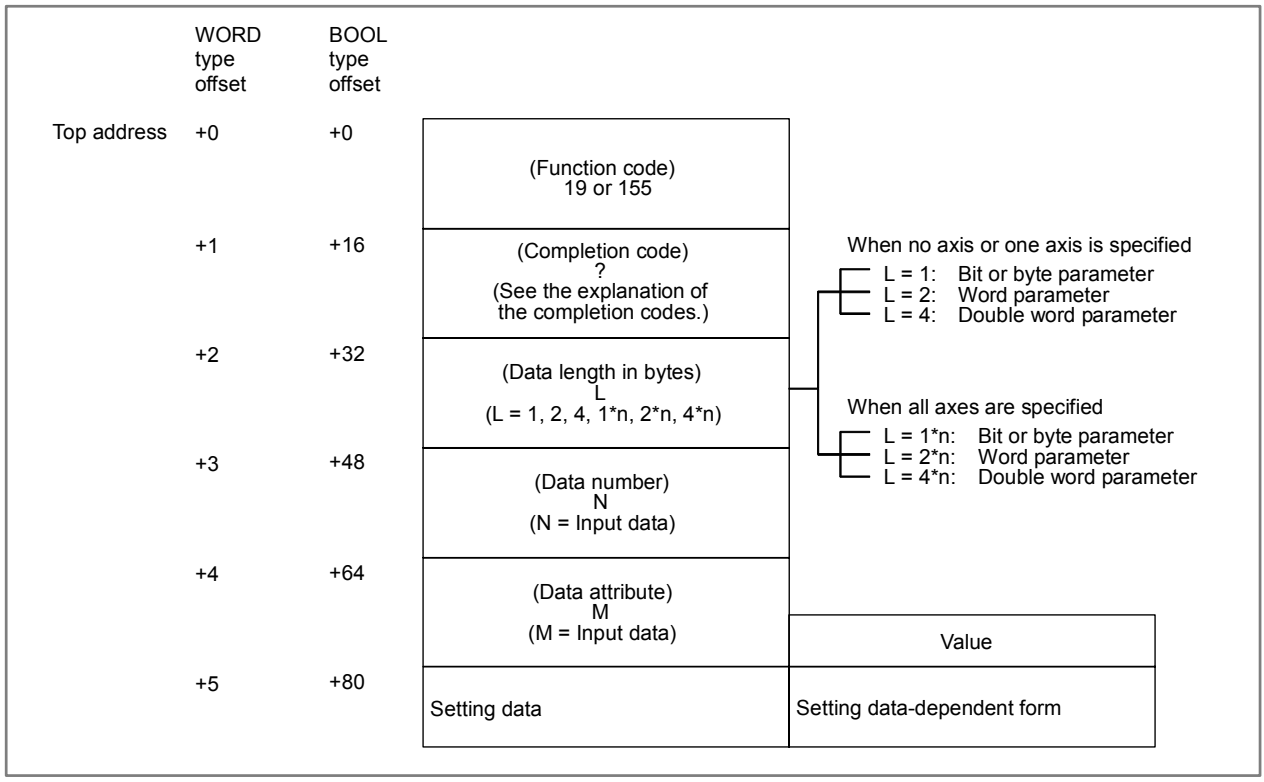
[Input data structure]



[Completion codes]

0: Setting data has been read normally.
3: The setting number specified for reading is invalid.
4: The specified data attribute is invalid because it is neither 0, -1, nor a value from 1 to n (n is the number of axes).

[Output data structure]

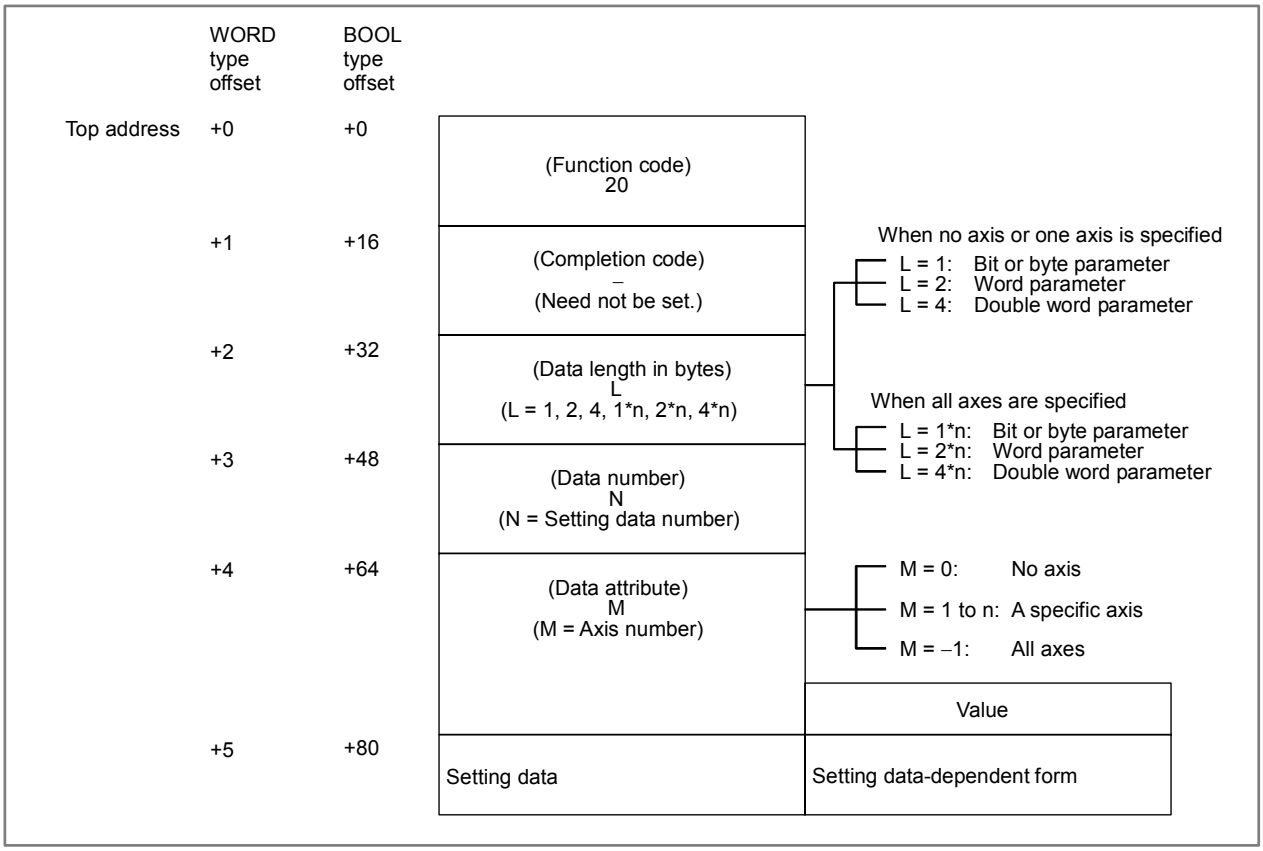


A.4.9 Writing Setting Data (Low-speed Response)

[Description]

Data can be written as setting data in the CNC.
For details of setting data, refer to the Operator's manual of the CNC.

[Input data structure]



[Completion codes]

0: Setting data has been written normally.
2: The byte length of the setting data specified for writing is invalid.
3: The setting data number specified for writing is invalid.
4: The specified data attribute is invalid because it is neither 0, -1, nor a value from 1 to n (n is the number of axes).
5: Data exceeding the allowable range was specified as setting data to be written. For example, when data outside the range from 0 to 3 is specified as the setting data to be written for I/O data, this completion code is returned.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 20	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (N = Input data)	
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) M (M = Input data)	Value
	+5	+80	Setting data: Input data	Setting data-dependent form

A.4.10 Reading a Custom Macro Variable (High-speed Response)

[Description]

A custom macro variable in the CNC can be read.
Custom macro variables may or may not be read depending on the variable type.

(1) Local variables
Local variables (#1 to #33) cannot be read.

(2) Common variables
Common variables (#100 to #149 and #500 to #531) can be read in floating-point representation. When the option to add common variables is provided, however, common variables range from #100 to #199 and #500 to #999.

For details of the custom macro variables, refer to the Operator's Manual for the CNC.

> ⚠**CAUTION**
> System variables cannot be read.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 21
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (N = Custom macro variable number)
	+4	+64	(Data attribute) M (M: Number of decimal places)
	+5	+80	(Data area) – (Need not be set) ≈ ≈

[Completion codes]

0: The custom macro variable has been read normally.
3: The number of a custom macro variable that cannot be read was specified as the data number.
5: The custom macro variable is not within the range from 0.0000001 to 99999999.
6: The custom macro option is not provided.

[Output data structure]

	WORD type offset	BOOL type offset		Value
Top address	+0	+0	(Function code) 21	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L: Byte length of custom macro variable data)	L = 6: Custom macro B The mantissa of a floating-point number is indicated in 4 bytes, and the exponent is indicated in 2 bytes.
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) M (M: Number of decimal places)	M = 0: The number of decimal places is not specified. M = 1 ≤ n ≤ 7: The number of decimal places is specified. n stands for the number of decimal places.
	+5	+80	Custom macro variable data (4 bytes) Mantissa (custom macro B)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Custom macro variable data (2 bytes) Exponent (custom macro B): The number of decimal digits	Signed binary 0 to 8 (no negative values)

A.4.11 Writing a Custom Macro Variable (Low-speed Response)

[Description]

Data can be written in a custom macro variable in the CNC.
For details of common variables, refer to the Operator's manual of the CNC.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 22
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) L (L: Byte length of custom macro variable data)
	+3	+48	(Data number) N (N = Custom macro variable number)
	+4	+64	(Data attribute) 0
	+5	+80	Custom macro variable data (4 bytes) Mantissa (custom macro B)
	+7	+112	Custom macro variable data (2 bytes) Exponent (custom macro B): The number of decimal digits

L = 6: Custom macro B
The mantissa of a floating-point number is indicated in 4 bytes, and the exponent is indicated in 2 bytes.

Value
Signed binary (A negative value is represented in 2's complement.)
Signed binary

[Completion codes]

0: The custom macro variable has been written normally.
2: The specified data length is invalid because it is not 6.
3: A custom macro variable number that cannot be written as the data number was specified.
6: The custom macro option or the additional common variable option has not been provided.

[Output data structure]

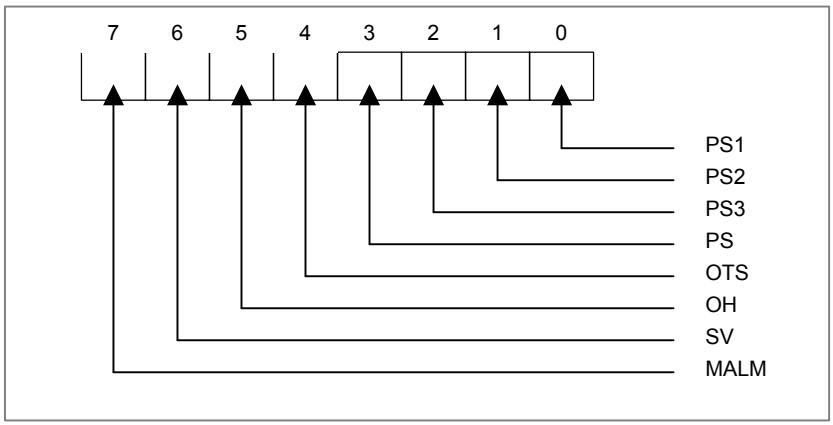
	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 22	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (N: Input data)	
	+3	+48	(Data number) N (N = Input data)	
	+4	+64	(Data attribute) – (Need not be set)	Value
	+5	+80	Custom macro variable data: Input data Mantissa (custom macro B)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Custom macro variable data: Input data Exponent (custom macro B): The number of decimal digits	Signed binary

A.4.12 Reading the CNC Alarm Status (High-speed Response)

[Description]

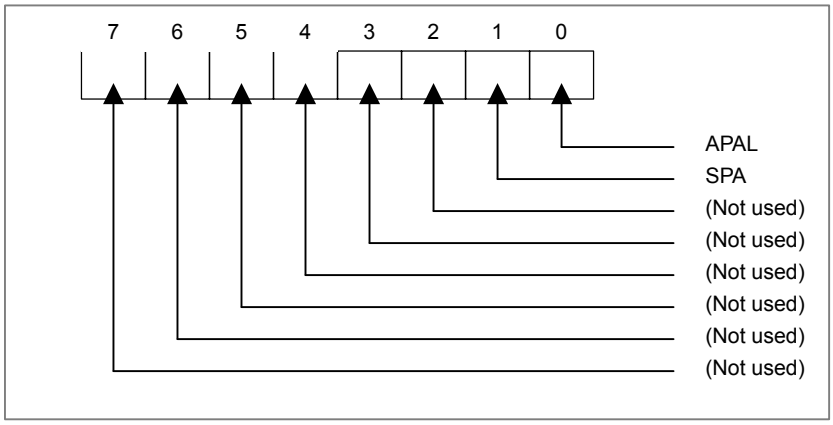
When the CNC is in the alarm status, the alarm status data can be read. The following alarm status data can be read:

(1) First byte of alarm status data



PS1 : P/S alarm 100 (PWE (parameter write enable) is set to 1.)
PS2 : P/S alarm 000 (Turn off the power before continuing operation. Some parameters activate this alarm status when they are written.)
PS3 : P/S alarm 101 (The part program recording area is disordered. This alarm is activated when the power to the CNC is turned off during part program editing or reading of a machining program. To release the alarm, then press the RESET key while holding down the PROG key.)
PS : A P/S alarm other than the above alarm is generated
OTS : Stroke limit alarm
OH : Overheat alarm
SV : Servo alarm
MALM : Memory alarm

(2) Second byte of alarm status data



APAL : APC alarm
SPA : Spindle alarm

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 23
	+1	+16	(Completion code) (Need not be set)
	+2	+32	(Data length in bytes) (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) (Need not be set)
	≈	≈	

[Completion codes]

0: This alarm status in the CNC has been read normally.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 23	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 2)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	CNC alarm status data	2 byte bit data. For the meanings of the bits, see [Description] in this section.

A.4.13 Reading the Current Program Number (High-speed Response)

[Description]

The program number of a machining program being executed or selected on the CNC can be read.
When a subprogram is executed on the CNC, the program number of the main program can also be read. Note that the program number that can be read is the first program number (first loop main program).
This function accepts only 4-digit program numbers. When the specification supports 8-digit program numbers, specify function code 90 to read 8-digit program numbers.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 24
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) – (Need not be set)

[Completion codes]

0: The program number of the currently executing program was read successfully.
6: The program number is an 8 - digit program number. (Use function code 90.)

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 24	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	Current program number: ON	Unsigned binary, 2 bytes long
	+6	+96	Program number of main program: OMN	

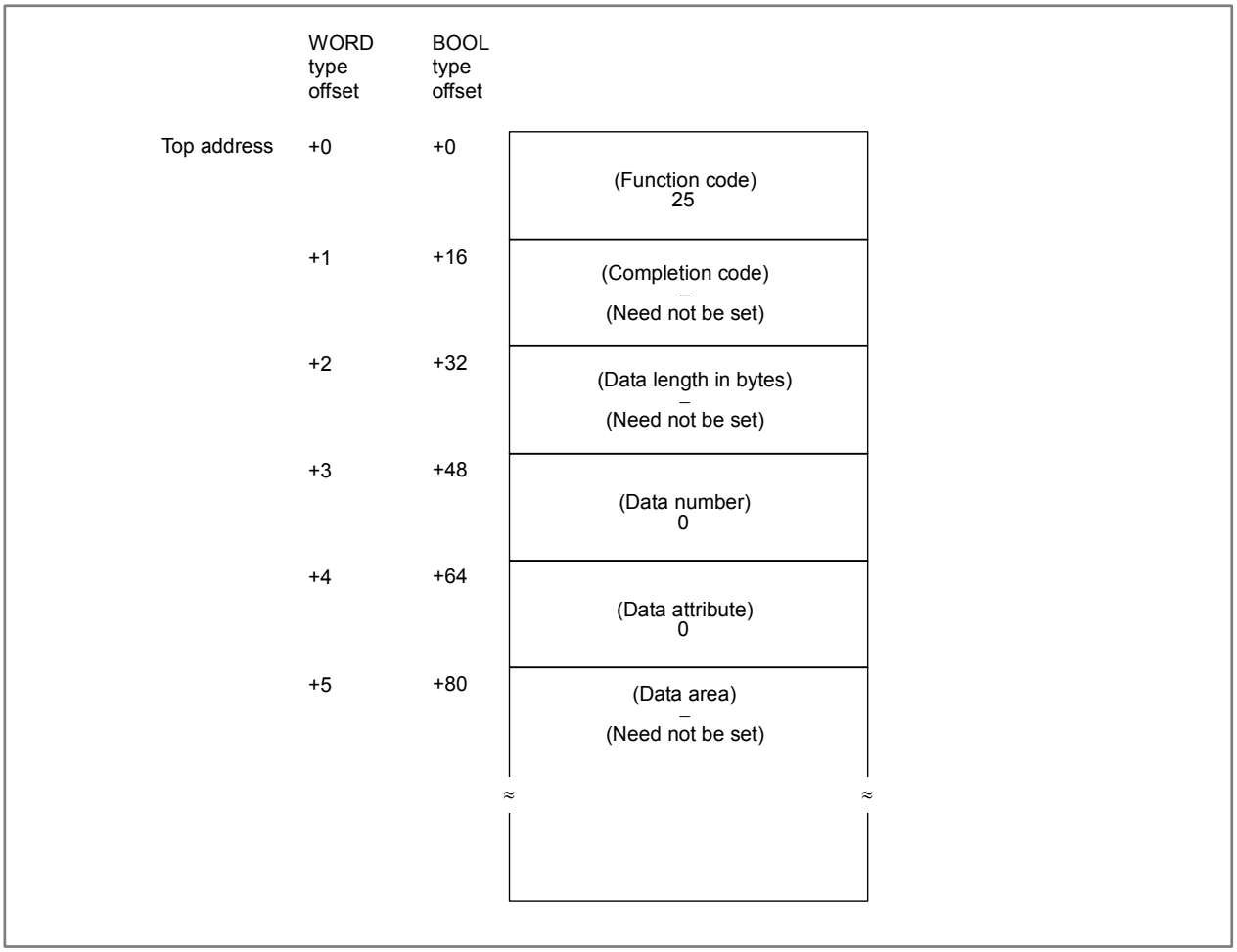
(a) Current program number (ON)
The program number of the program being executed is set.
(b) Program number of main program (OMN)
When the currently executing program is a subprogram, the program number of its main program (first loop main program) is set. When the currently executing program is not a subprogram, 0 is set.

A.4.14 Reading the Current Sequence Number (High-speed Response)

[Description]

The sequence number of a machining program being executed on the CNC can be read. If sequence numbers are not assigned to all blocks of the machining program, the sequence number of the most recently executed block is read.

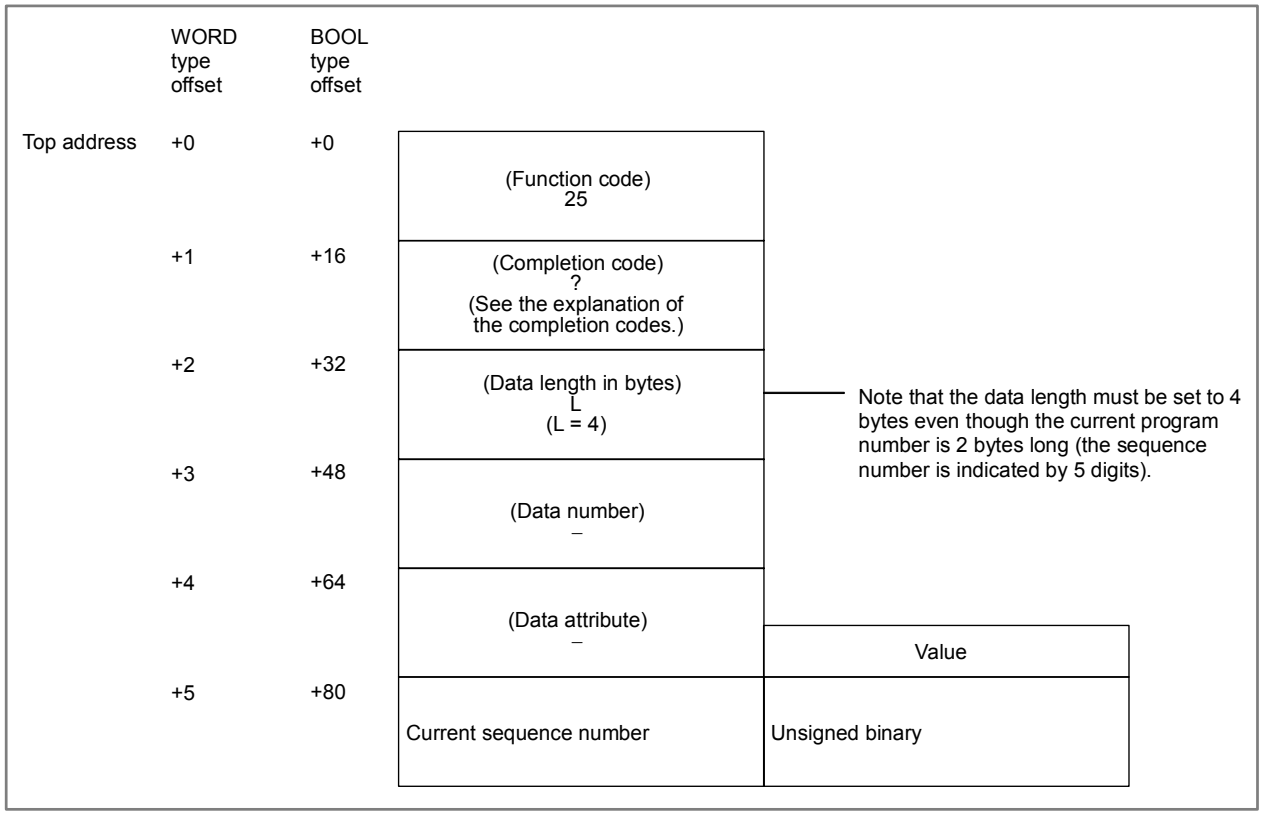
[Input data structure]



[Completion codes]

0: The current sequence number has been read normally.

[Output data structure]

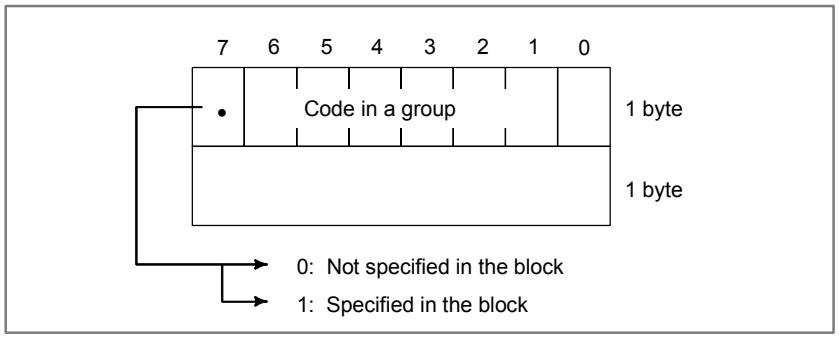


A.4.15 Reading Modal Data (High-speed Response)

[Description]

Modal information can be read from the CNC.

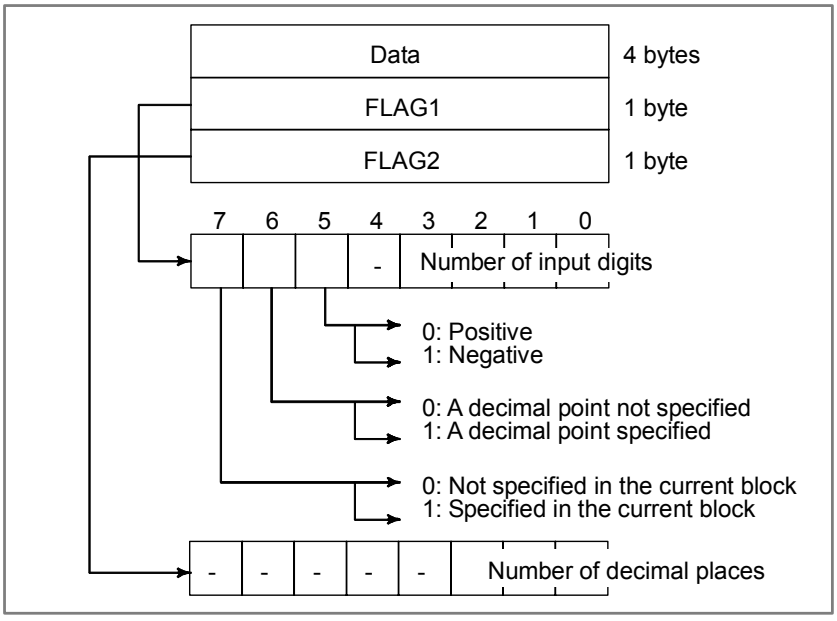
(1) Format and types of modal data for the G function
Data corresponding to the specified identification code is read and stored in the data area. Whether the data is specified in the block specified in the attribute of the data is determined by the value at the most significant bit.



Identification code	Data type: G code for machining center (M)	Data: Code in a group	Data type: G code for lathe (T, G) A series	B series	C series	Data: Code in a group
0	G00	0	G00	G00	G00	0
	G01	1	G01	G01	G01	1
	G02	2	G02	G02	G02	2
	G03	3	G03	G03	G03	3
	G33	4	G32	G33	G33	4
			G33			8
			G34	G34	G34	9
			G90	G77	G20	5
			G92	G78	G21	6
			G94	G79	G24	7
			G71 (G series only)	G71 (G series only)	G72 (G series only)	10
			G72 (G series only)	G72 (G series only)	G73 (G series only)	11
			G73 (G series only)	G73 (G series only)	G74 (G series only)	12
			G74 (G series only)	G74 (G series only)	G75 (G series only)	13
1	G17	0	G96	G96	G96	1
	G18	8	G97	G97	G97	0
	G19	4				
2	G90	0		G90	G90	0
	G91	1		G91	G91	1
3			G68	G68	G68	1
			G69	G69	G69	0
4	G94	0	G98	G94	G94	0
	G95	1	G99	G95	G95	1
5	G20	0	G20	G20	G70	0
	G21	1	G21	G21	G71	1
6	G40	0	G40	G40	G40	0
	G41	1	G41	G41	G41	1
	G42	2	G42	G42	G42	2
7	G43	1	G25	G25	G25	0
	G44	2	G26	G26	G26	1
	G49	0				
8	G73	10	G22	G22	G22	1
	G74	11	G23	G23	G23	0
	G76	12				
	G80	0				
	G81	1				
	G82	2				
	G83	3				
	G84	4				
	G85	5				
	G86	6				
	G87	7				
	G88	8				
	G89	9				
9	G98	0	G80	G80	G80	0
	G99	1	G83	G83	G83	1
			G84	G84	G84	2
			G85	G85	G85	3
			G87	G87	G87	5
			G88	G88	G88	6
			G89	G89	G89	7

Identification code	Data type: G code for machining center (M)	Data: Code in a group	Data type: G code for lathe (T, G) A series	Data type: G code for lathe (T, G) B series	Data type: G code for lathe (T, G) C series	Data: Code in a group
10	G50	0		G98	G98	0
	G51	1		G99	G99	1
11	G66	1	G66	G66	G66	1
	G67	0	G67	G67	G67	0
13	G54	0	G54	G54	G54	0
	G55	1	G55	G55	G55	1
	G56	2	G56	G56	G56	2
	G57	3	G57	G57	G57	3
	G58	4	G58	G58	G58	4
	G59	5	G59	G59	G59	5
14	G61	1				
	G62	2				
	G63	3				
	G64	0				
15	G68	1				
	G69	0				
16	G15	0				
	G16	1				
17	G40.1	1				
	G41.1	2				
	G42.1	0				
18	G25	0				
	G26	1				
19			G50.2	G50.2	G50.2	0
			G51.2	G51.2	G51.2	1
20	G13.1	0	G13.1	G13.1	G13.1	0
	G12.1	1	G12.1	G12.1	G12.1	1

(2) Format and types of modal data for other than the G function



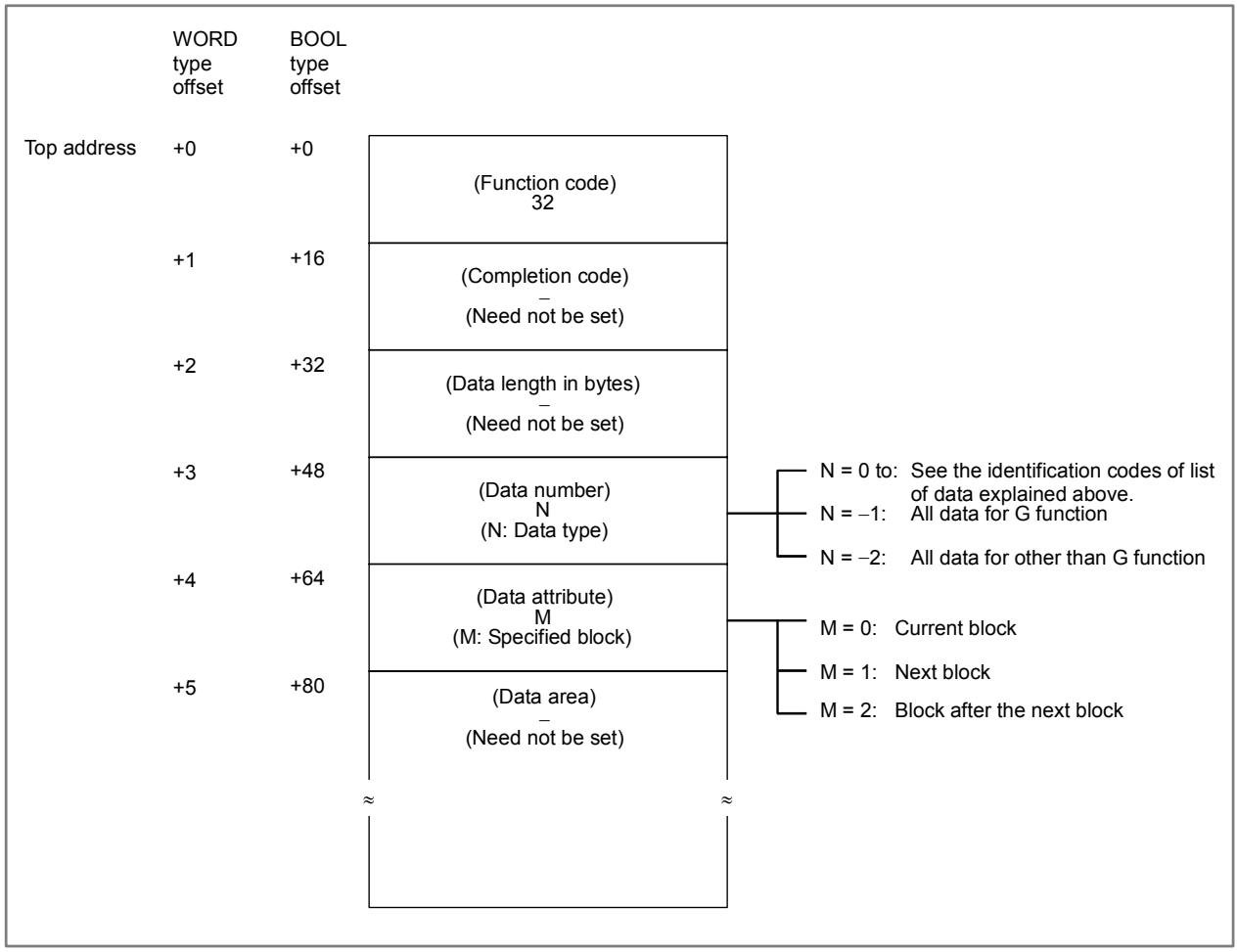
NOTE

1 The specification of whether a decimal point is specified or not, in FLAG1, and the specification of the number of decimal places, in FLAG2, are valid only for F code. Even if a decimal point is not specified, the number of decimal places may not be 0.

2 As the numbers of input digits, M, S, T, and B, in a command address, the allowable numbers of digits that are specified for the appropriate parameters are returned.
M: Allowable number of digits of M code No. 3030
S: Allowable number of digits of S code No. 3031
T: Allowable number of digits of T code No. 3032
B: Allowable number of digits of B code No. 3033

Data type		
Identification code	**Specified address**	
–2	**Enter identification codes 100 to 126 at one time.**	
100	B	(Second auxiliary function)
101	D	
102	E	(Reserved)
103	F	
104	H	
105	L	
106	M	
107	S	
108	T	
109	R	
110	P	
111	Q	
112	A	
113	C	
114	I	
115	J	
116	K	
117	N	
118	O	
119	U	
120	V	
121	W	
122	X	
123	Y	
124	Z	
125	M2	
126	M3	

[Input data structure]

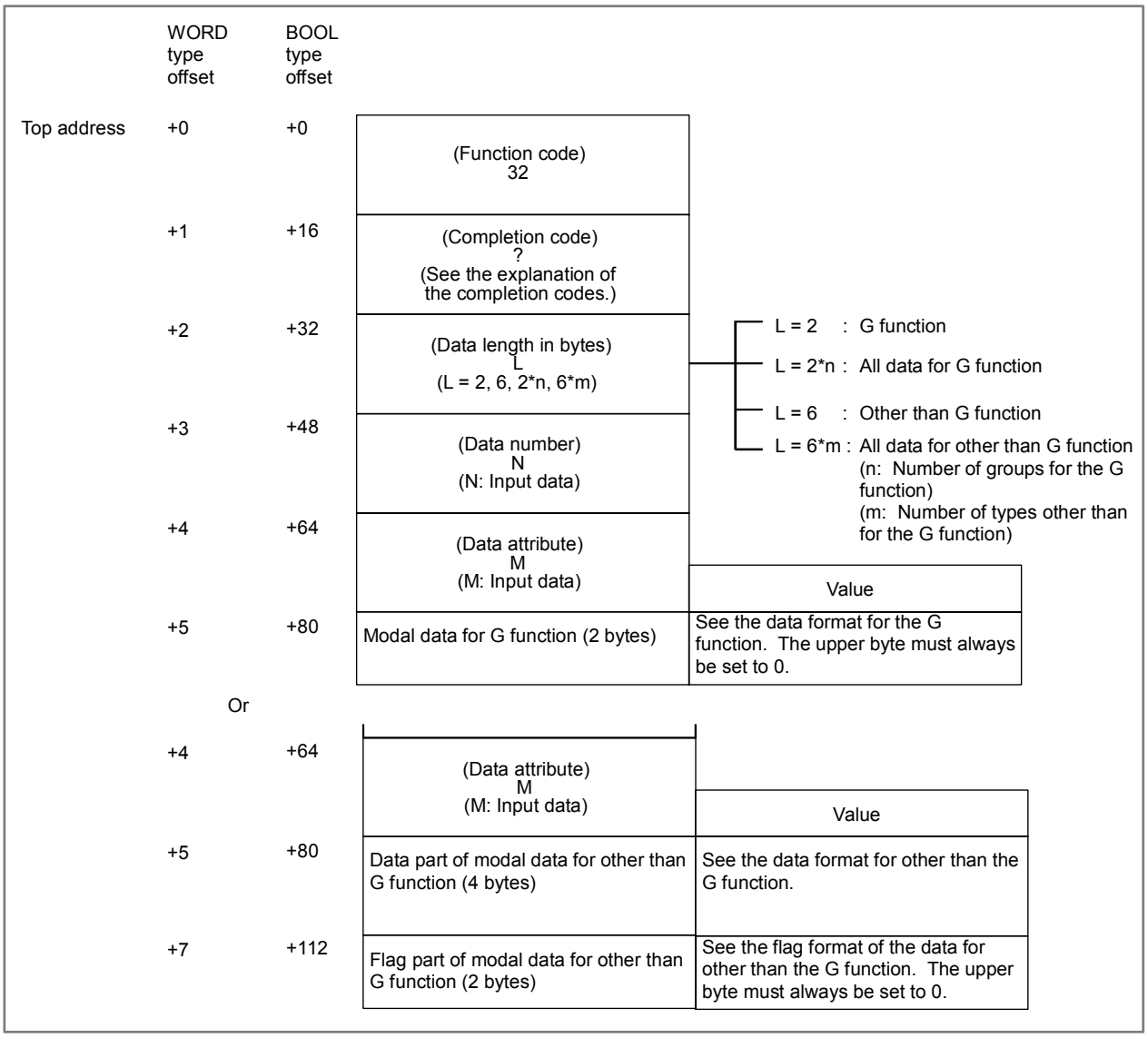


When all data items are specified to be read, the data items are all output simultaneously in the order specified in the above data table.

[Completion codes]

0: Modal information has been read normally.
3: Invalid data is specified as the data number.
4: Invalid data is specified as the data attribute.

[Output data structure]



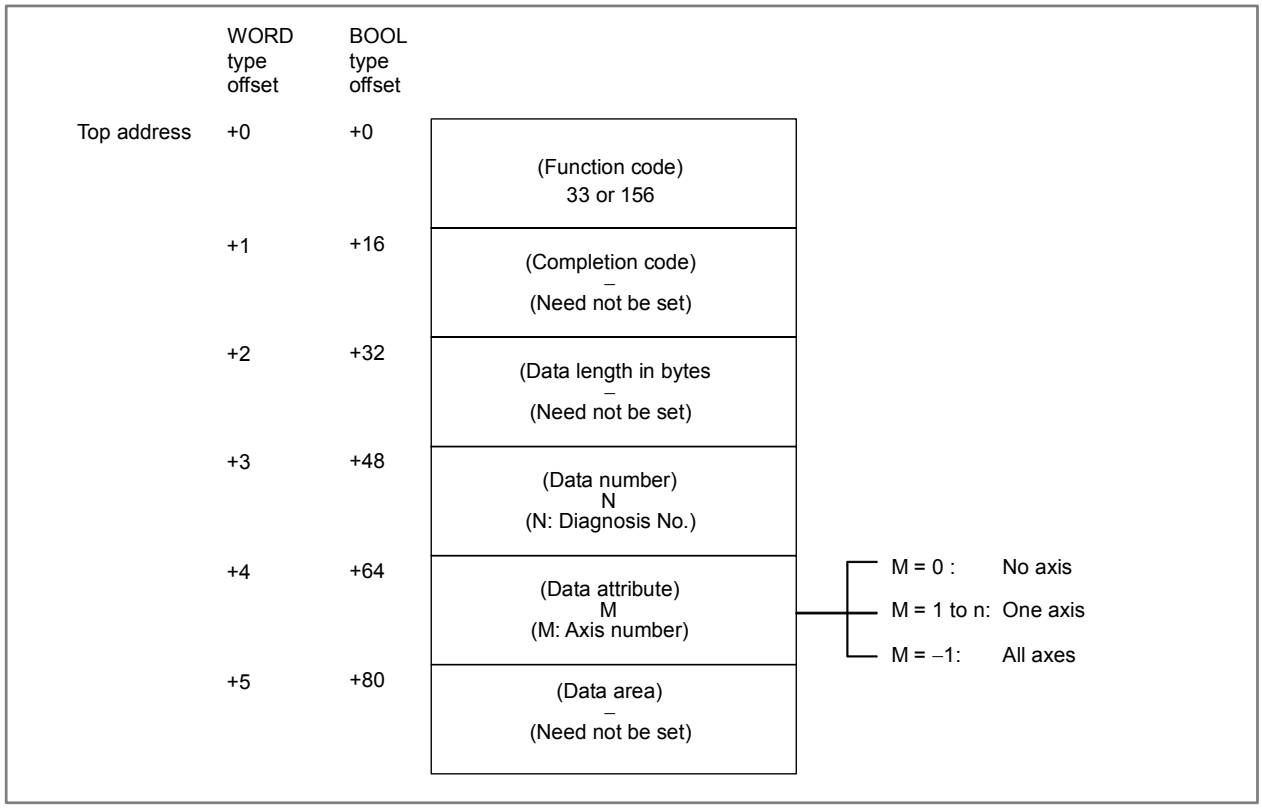
When all data items are specified to be read, the data items are all output simultaneously in the order specified in the above data table.

A.4.16 Reading Diagnosis Data (High-speed Response)

[Description]

The information displayed on the diagnosis data screen in the CNC can be read.

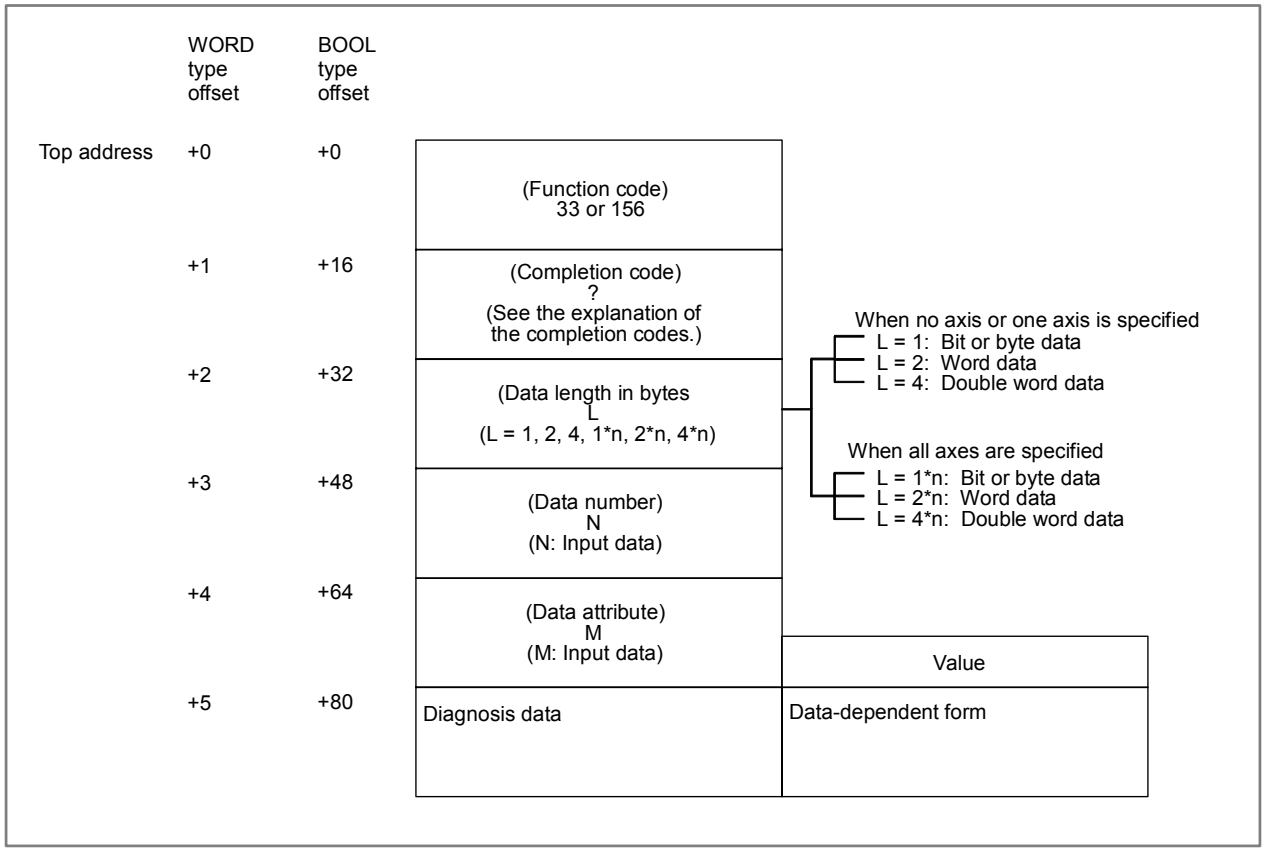
[Input data structure]



[Completion codes]

0: Diagnosis data has been read from the CNC normally.
3: The specified diagnosis data number is invalid.
4: The data specified as the data attribute is invalid because it is neither 0, -1, nor a value from 1 to n (n is the number of axes).
6: An option required for reading the specified diagnosis data, such as the remote buffer option, is not provided.

[Output data structure]



A.4.17 Reading Value of the P-code Macro Variable (High-speed Response)

[Description]

This function gets the value of variable for Macro-compiler (P-code macro variable) of specified number.
The extended P-code macro variable is not able to be read.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 59
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (P-code macro variable number)
	+5	+80	(Data attribute) 0
	+6	+96	(Data area) – (Need not be set)

⚠CAUTION
The 'Data number' occupies 4 bytes instead of 2 bytes of usual data structure.

[Completion codes]

0: Success to read the value of P-code macro variable.
3: The P-code macro variable specified by 'Data number' can not be read.
5: The value of the P-code macro variable is out of range (±0.0000001 -- ±99999999.).
6: No option, or no Macro ROM module.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 59	
	+1	+16	(Completion code) ? (See the explanation above)	
	+2	+32	(Data length in bytes) 6	
	+3	+48	(Data number) N (Same as input data)	
	+5	+80	(Data attribute) _ (Same as input data)	Value
	+6	+96	Value of P-code macro variable (4 bytes)	Signed binary (Minus number is represented by 2's complemental.)
	+8	+128	Figures after decimal point of the value of P-code macro variable (2 bytes)	Signed binary (Minus number is represented by 2's complemental.)

A.4.18 Writing Value of the P-code Macro Variable (Low-speed Response)

[Description]

This function stores the value into the variable for Macro-compiler (P-code macro variable) of specified number.
The extended P-code macro variable can not be written into.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 60	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 6	
	+3	+48	(Data number) N (P-code macro variable number)	
	+5	+80	(Data attribute) 0	Value
	+6	+96	Value of P-code macro variable (4 bytes)	Signed binary (Minus number is represented by 2's complemental.)
	+8	+128	Figures after decimal point of the value of P-code macro variable (2 bytes)	Signed binary (Minus number is represented by 2's complemental.)

⚠CAUTION
The 'data number' occupies 4 bytes instead of 2 bytes of usual data structure.

[Completion codes]

0: Success to store the value into P-code macro variable.
2: The data length has illegal data (is not 6).
3: The P-code macro variable specified by 'Data number' can not be written.
6: No option, or no Macro ROM module.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 60
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 6 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+5	+80	(Data attribute) _ (Same as input data)
	+6	+96	Value of P-code macro variable (4 bytes)
	+8	+128	Figures after decimal point of the value of P-code macro variable (2 bytes)

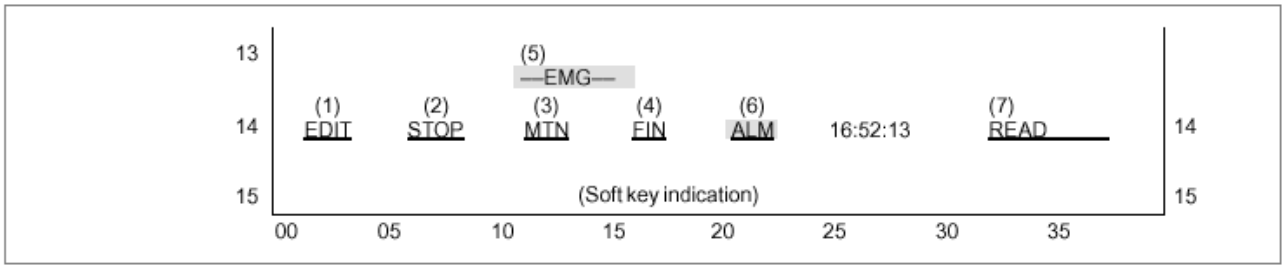
A.4.19 Reading CNC Status Information (High-speed Response)

[Description]

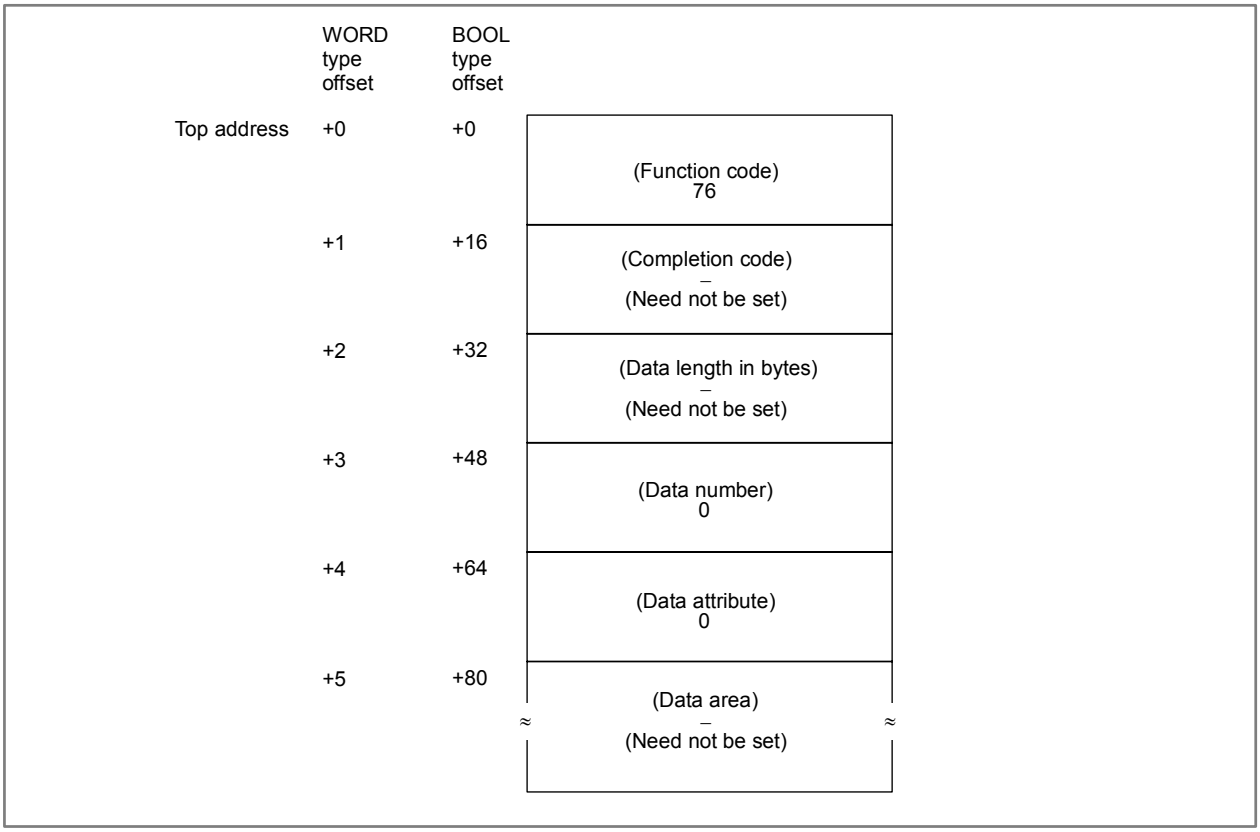
Status information (status indication on the screen) can be read from the CNC.
The types of status information that can be read are as follows.

(1) Indication of which mode is selected, automatic or manual
(2) Status of automatic operation
(3) Status of movement along the axis and dwelling
(4) Status of M, S, T, and B functions
(5) Statuses of emergency stop and the reset signal
(6) Alarm status
(7) Status of program edits

(Indication)



[Input data structure]



[Completion codes]

0: CNC status information has been read normally.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 76	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 14	
	+3	+48	(Data number) – (Input data)	
	+4	+64	(Data attribute) – (Input data)	Value
	+5	+80	Indication of which mode is currently selected, automatic or manual (2 bytes)	0 : MDI 1 : MEMory 2 : **** (Other states) 3 : EDIT 4 : HaNDle 5 : JOG 6 : Teach in JOG 7 : Teach in HND 8 : INC. feed 9 : REFerence 10 : ReMoTe
	+6	+96	Status of automatic operation (2 bytes)	0 : **** (Reset states) 1 : STOP 2 : HOLD 3 : STaRT
	+7	+112	Status of movement along the axis or dwelling (2 bytes)	0 : *** (Other states) 1 : MoTioN 2 : DWell
	+8	+128	Status of M, S, T, and B functions (2 bytes)	0 : *** (Other states) 1 : FIN
	+9	+144	Status of emergency stop (2 bytes)	0 : (Releases the emergency stop state) 1 : – –EMerGency – – 2 : – RESET – (The reset signal is on.)
	+10	+160	Alarm status (2 bytes)	0 : *** (Other states) 1 : ALarM 2 : BATtery low
	+11	+176	Status of program edit (2 bytes)	0 : ******* (Non editing) 1 : EDIT 2 : SeaRCH 3 : OUTPUT 4 : INPUT 5 : COMPARE 6 : LabelSKip 7 : OFST 8 : WSFT 9 : ReSTaRt

A.4.20 Reading the Current Program Number (8-digit Program Numbers) (High-speed Response)

[Description]

This function reads CNC program numbers extended to 8 digits from the usual 4 digits.
Basically, this function is the same as function code 24 excluding the different data length of function code 90.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 90
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) – (Need not be set)
	+9	+144	

[Completion codes]

0: The program number of the currently executing program has been read normally.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 90	
	+1	+16	(Completion code) ? (See the explanation of the completion codes, above.)	
	+2	+32	(Data length in bytes) 8	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	Number of the program currently being executed ON	Unsigned binary format, 4-byte length
	+7	+112	Program number of the main program ON	
	+9	+144		

(a) Number of the program currently being executed (ON)
The program number of the program currently being executed is set.

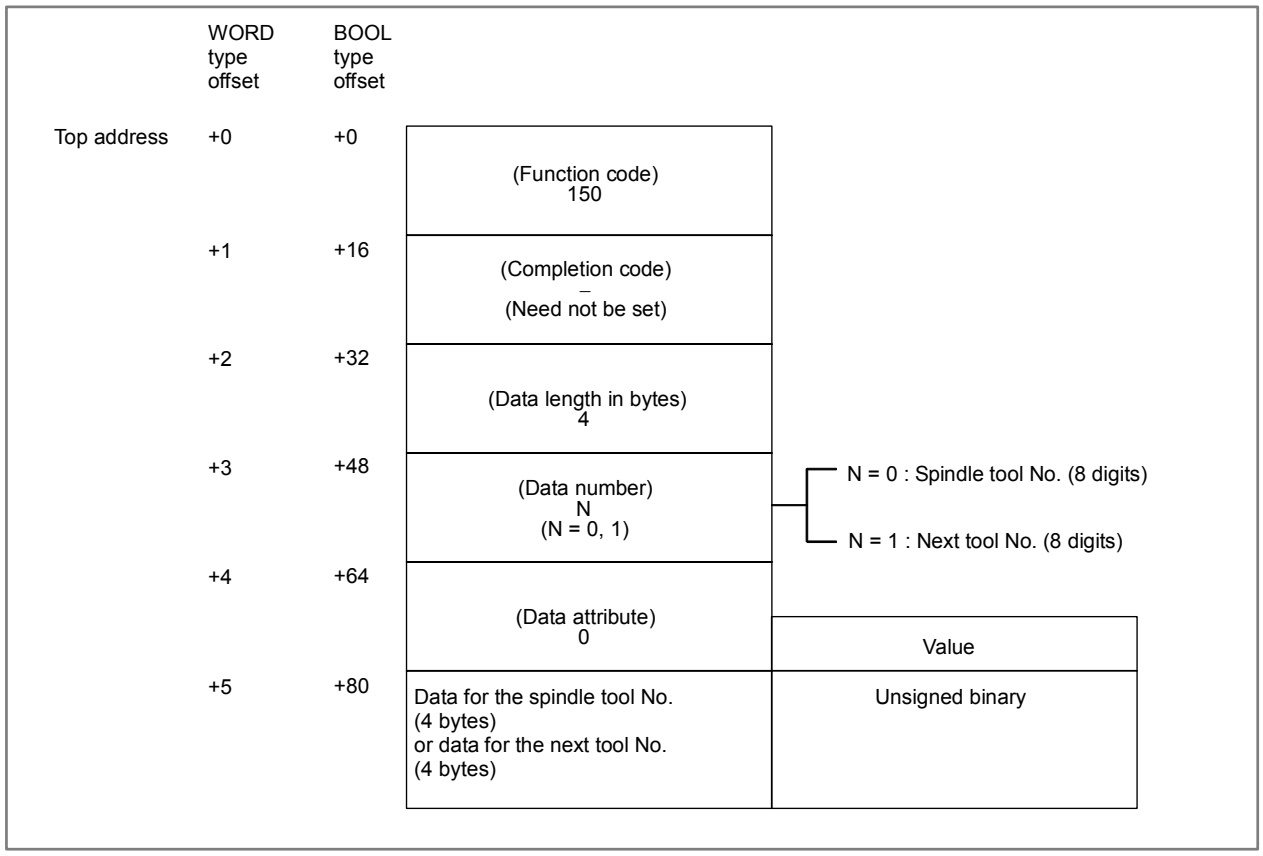
(b) Program number of the main program (OFF)
If the program currently being executed is a subprogram, the program number of its main program is set.
If the program currently being executed is not a subprogram, 0 is set.

A.4.21 Entering Data on the Program Check Screen (Low-speed Response)

[Description]

On the program check screen of the CNC, data can be entered for the spindle tool No. (HD.T) and the next tool No. (NX.T).
This function is effective only when bit 2 of parameter 3108 is 1.

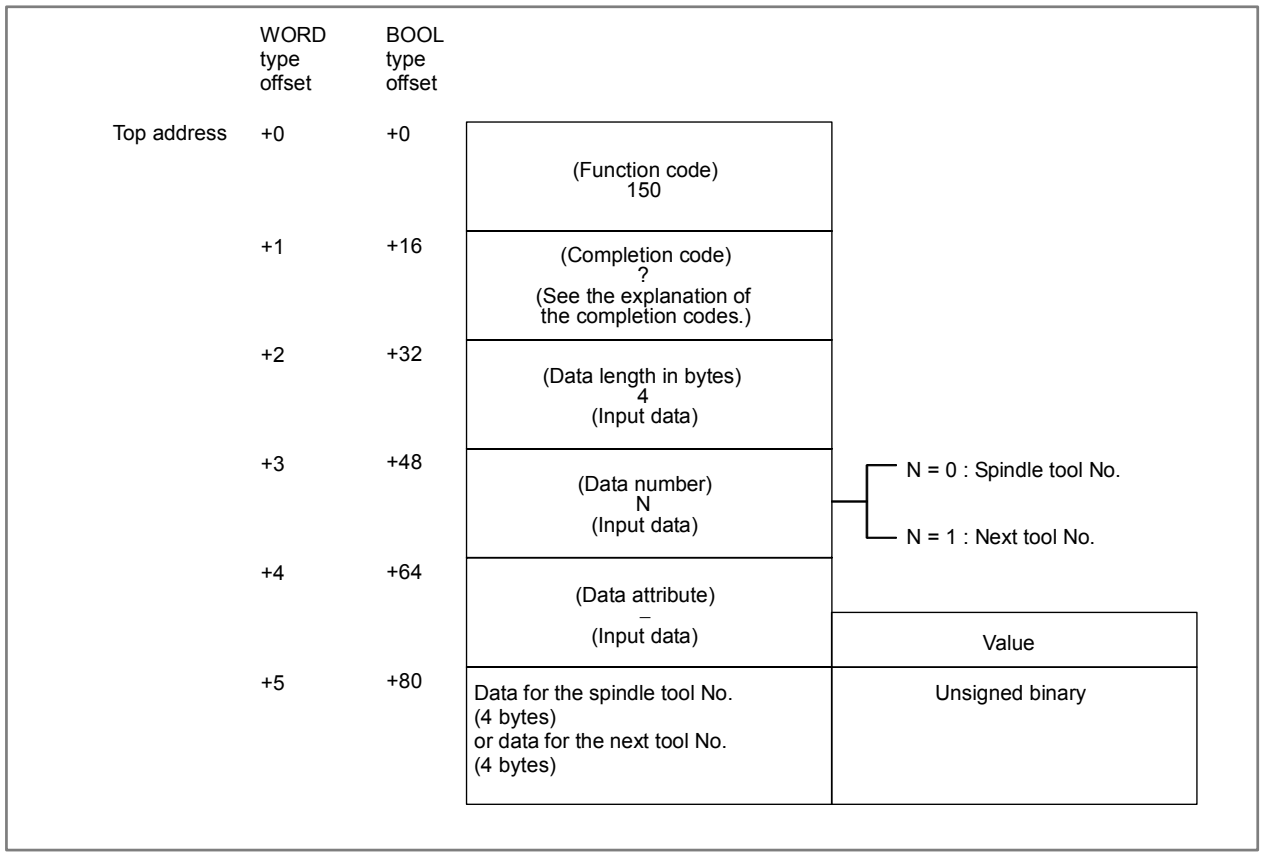
[Input data structure]



[Completion codes]

0: Data has been entered on the program check screen normally.
2: The data length in bytes is invalid.
3: The data No. is invalid.

[Output data structure]

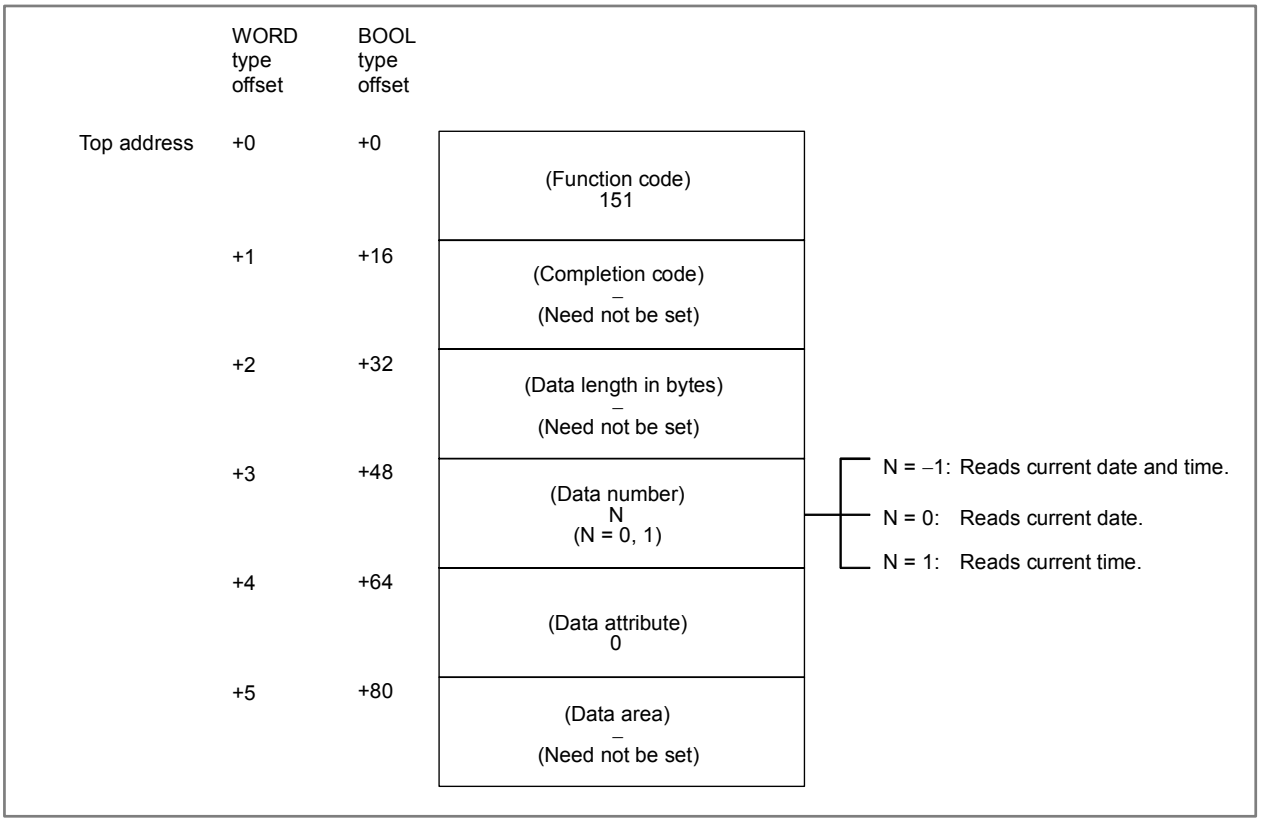


A.4.22 Reading Clock Data (Date and Time) (High-speed Response)

[Description]

The current date (year, month, day) and time (hours, minutes, seconds) can be read from the clock built into the CNC.

[Input data structure]



[Completion codes]

0: Data of the clock built into the CNC has been read normally.
3: A value other than 0, 1, and -1 was specified for the data No.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 151	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 6/12	
	+3	+48	(Data number) N (Input data)	
	+4	+64	(Data attribute) – (Input data)	Value
	+5	+80	Current date (year) or time (hours)	Unsigned binary
	+6	+96	Current date (month) or time (minutes)	
	+7	+112	Current date (day) or time (seconds)	

When both the current date and current time are specified to be read by entering [–1] for the data No.

			– (Input data)	Value
	+5	+80	Current date (year)	Unsigned binary
	+6	+96	Current date (month)	
	+7	+112	Current date (day)	
	+8	+128	Current time (hours)	
	+9	+144	Current time (minutes)	
	+10	+160	Current time (seconds)	

[Example] September 10th, 1990 (hours:minutes:seconds)

WORD type offset	BOOL type offset	Data area
		1990
+1	+16	9
+2	+32	10

[Example] 23:59:59

WORD type offset	BOOL type offset	Data area
		23
+1	+16	59
+2	+32	59

A.4.23 Specifying the Number of the Program for I/O Link(Low-speed Response)

[Explanation of data]

Specify the number of the program to be input/output using the data input/output function with I/O Link.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 194	
	+1	+16	(Completion code) — (Need not be set)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) 0	Value
	+5	+80	Program number	Signed binary format
	+6	+96		

[Completion codes]

0: The specification of the program number terminated normally.
5: Invalid data was specified for the program number, i.e., the data falls outside the range of 1 to 9999 or is not - 9999.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 194	
	+1	+16	(Completion code) — (See the explanation of completion codes, above.)	
	+2	+32	(Data length in bytes) 2 (Data at input time)	
	+3	+48	(Data number) 0 (Data at input time)	
	+4	+64	(Data attribute) 0 (Data at input time)	Value
	+5	+80	Program number (Data at input time)	Signed binary format
	+6	+96		

⚠CAUTION
For details of this function, see the section on data input/output functions using I/O Link in the "CNC Connection Manual (Functions)."

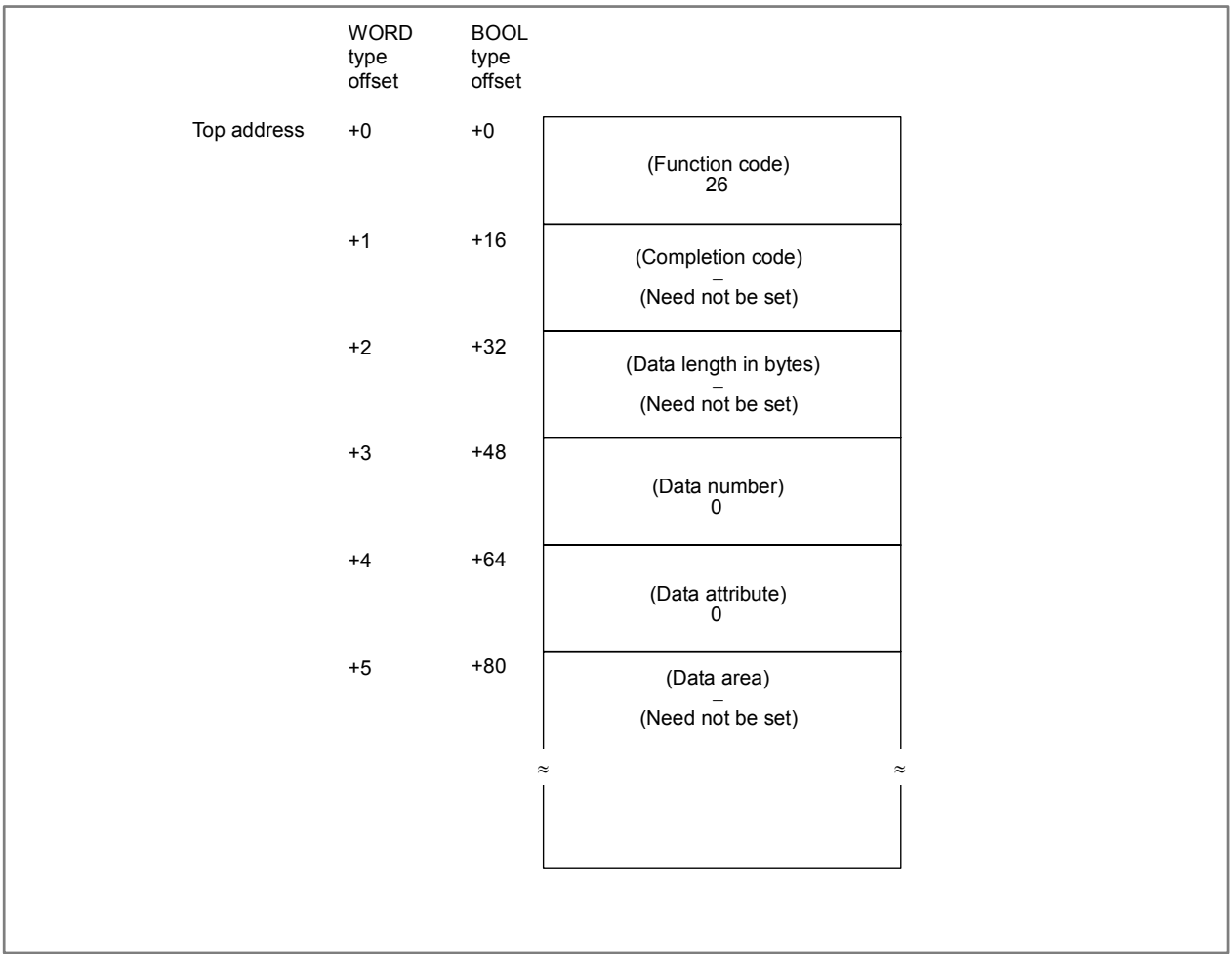
A.5 AXIS INFORMATION

A.5.1 Reading the Actual Velocity of Controlled Axes (High-speed Response)

[Description]

The actual velocity of a movement on CNC-controlled axes can be read. Note that the read speed is the composite velocity for the controlled axes. When movement involves only the basic three axes, the X, Y, and Z axes, the composite velocity equals the actual velocity. When movement, however, involves the fourth axis, such as a rotation axis or a parallel axis, as well as some of the basic three axes, the composite velocity for all the relevant axes does not equal the actual velocity.

[Input data structure]



[Completion codes]

0: The actual velocity for the controlled axes has been read normally.

[Output data structure]

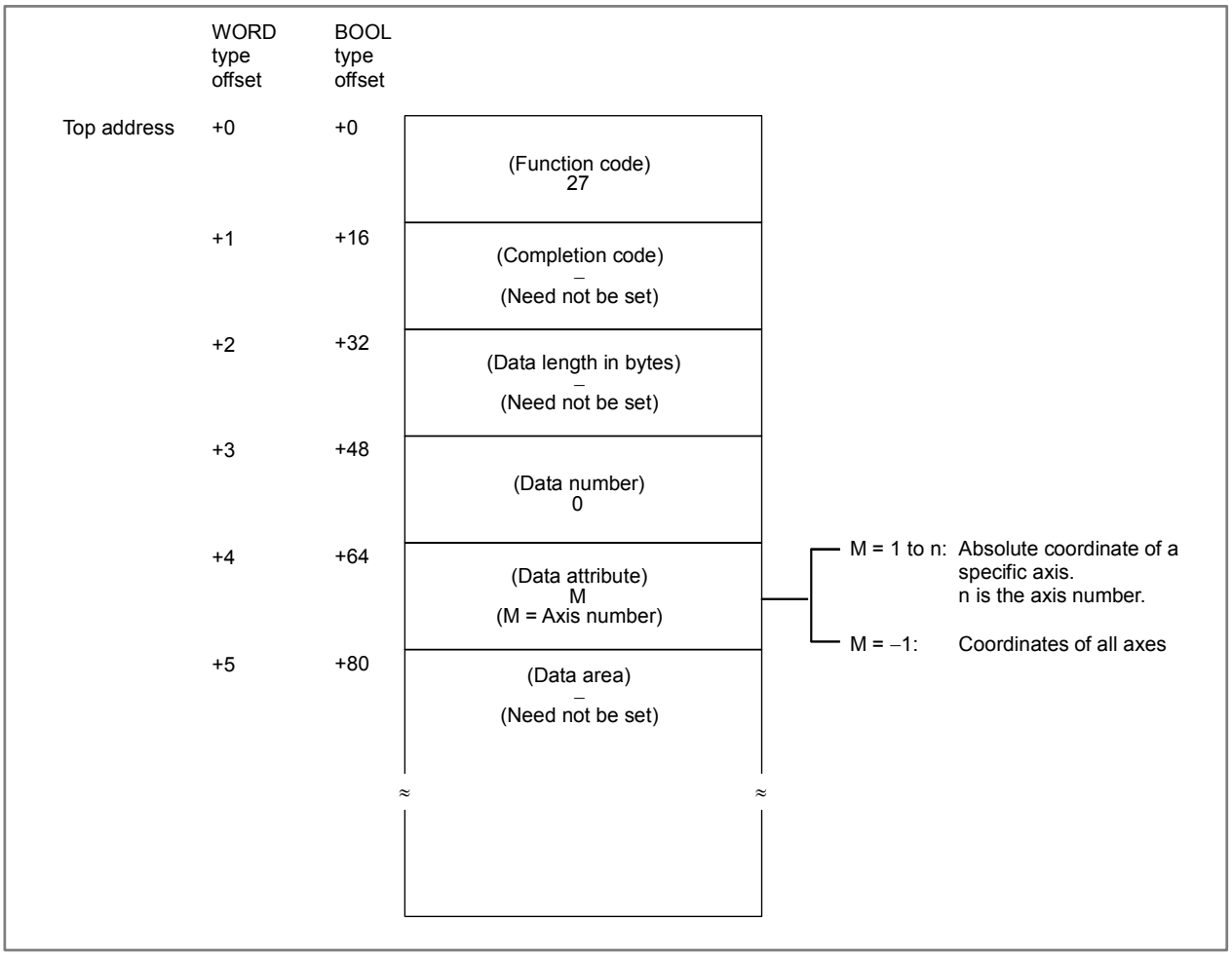
	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 26	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	Actual velocity for controlled axes	Unsigned binary <Data increments> • Input in mm 1 mm/min. • Input in inches 0.01 inch/min.

A.5.2 Reading the Absolute Position (Absolute Coordinates) of Controlled Axes (High-speed Response)

[Description]

The absolute coordinates of the CNC-controlled axes for movement can be read. The absolute coordinates indicate those after cutter compensation or tool length compensation.

[Input data structure]



[Completion codes]

0: The absolute coordinates of the controlled axes have been read normally.

4: Data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 27	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n, n is the number of axes specified.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Absolute coordinate of the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

			Value
+5	+80	Absolute coordinate of the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
+7	+112	Absolute coordinate of the second axis (4 bytes)	
+9	+144	Absolute coordinate of the third axis (4 bytes)	
+11	+176	Absolute coordinate of the fourth axis (4 bytes)	

[Output data unit]

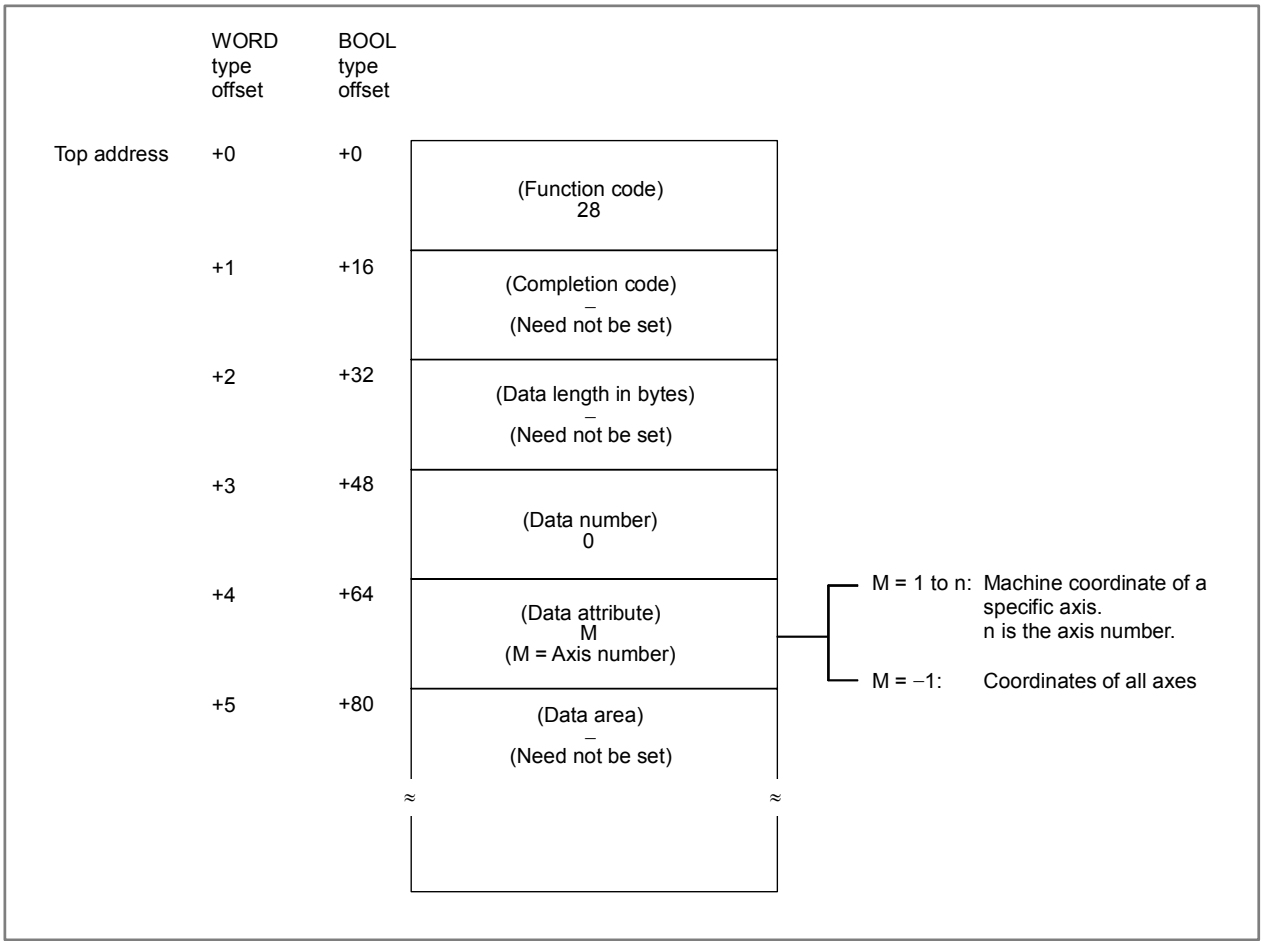
		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

A.5.3 Reading the Machine Position (Machine Coordinates) of Controlled Axes (High-speed Response)

[Description]

The machine coordinates of CNC-controlled axes for movement can be read. The read value equals the machine coordinate indicated on the current position display screen displayed in the CNC. (This screen can be displayed by pressing the function key POS.)

[Input data structure]



⚠CAUTION

When an inch machine is used in metric input, or when a millimeter machine is used in inch input, the machine position that is read with bit 0 of parameter No. 3104 set to 1 differs from the value indicated by the CNC. In this case, therefore, the value read through the ladder must be calculated (converted).

[Completion codes]

0: The machine coordinates of the controlled axes have been read normally.

4: Data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of the controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 28	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n, n is the number of axes specified.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Machine coordinate of the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

				Value
	+5	+80	Machine coordinate of the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Machine coordinate of the second axis (4 bytes)	
	+9	+144	Machine coordinate of the third axis (4 bytes)	
	+11	+176	Machine coordinate of the fourth axis (4 bytes)	

[Output data unit]

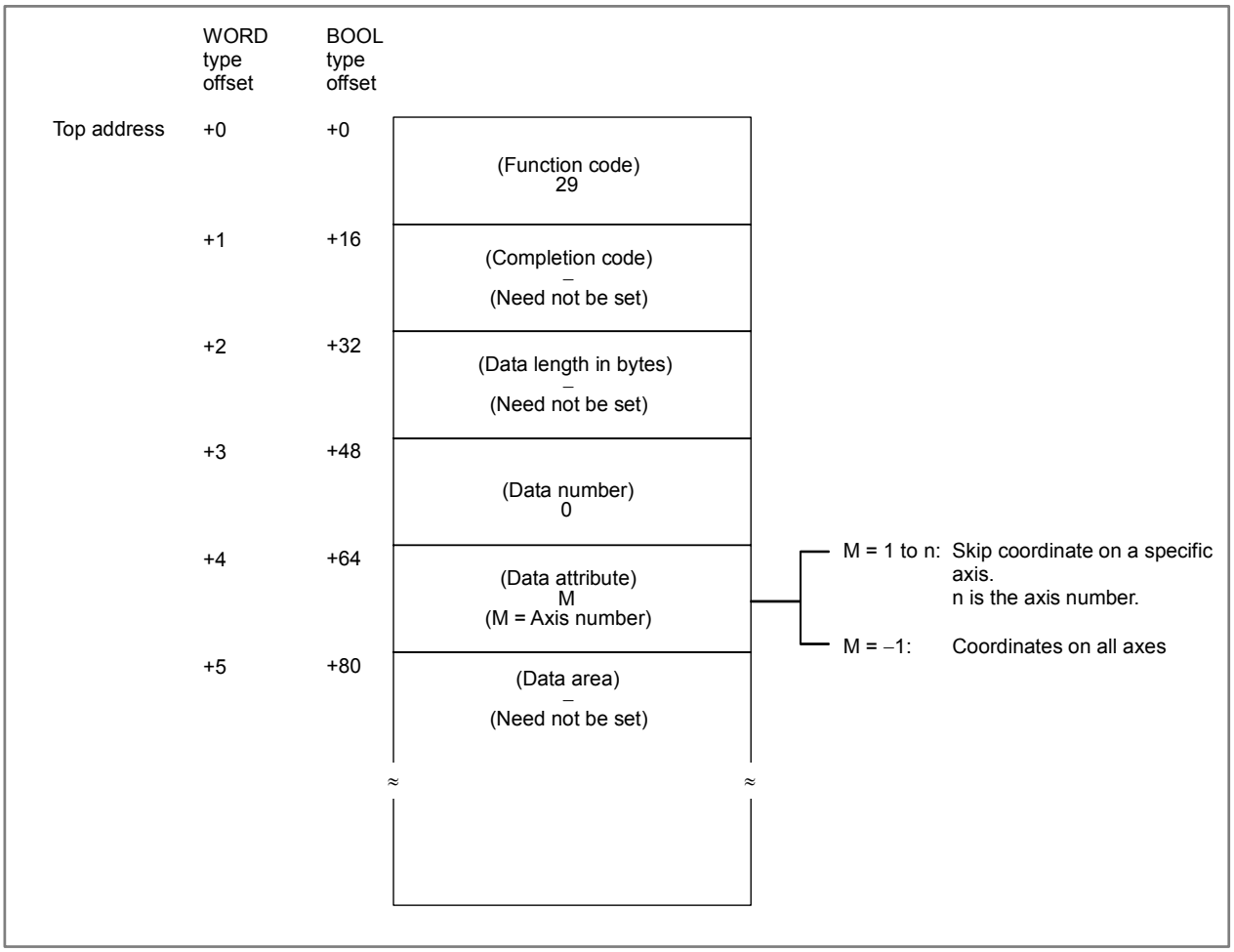
		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

A.5.4 Reading a Skip Position (Stop Coordinates of Skip Operation (G31)) of Controlled Axes (High-speed Response)

[Description]

When a block of the skip operation (G31) is executed by the CNC and the skip signal goes on to stop the machine, the absolute coordinates of the stop position on the axes of movement can be read.

[Input data structure]



[Completion codes]

0: The coordinates of the skip stop position for the controlled axes have been read normally.

4: Data specified for the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 29	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n, n is the number of axes specified.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Skip coordinate of the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

				Value
	+5	+80	Skip coordinate of the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Skip coordinate of the second axis (4 bytes)	
	+9	+144	Skip coordinate of the third axis (4 bytes)	
	+11	+176	Skip coordinate of the fourth axis (4 bytes)	

[Output data unit]

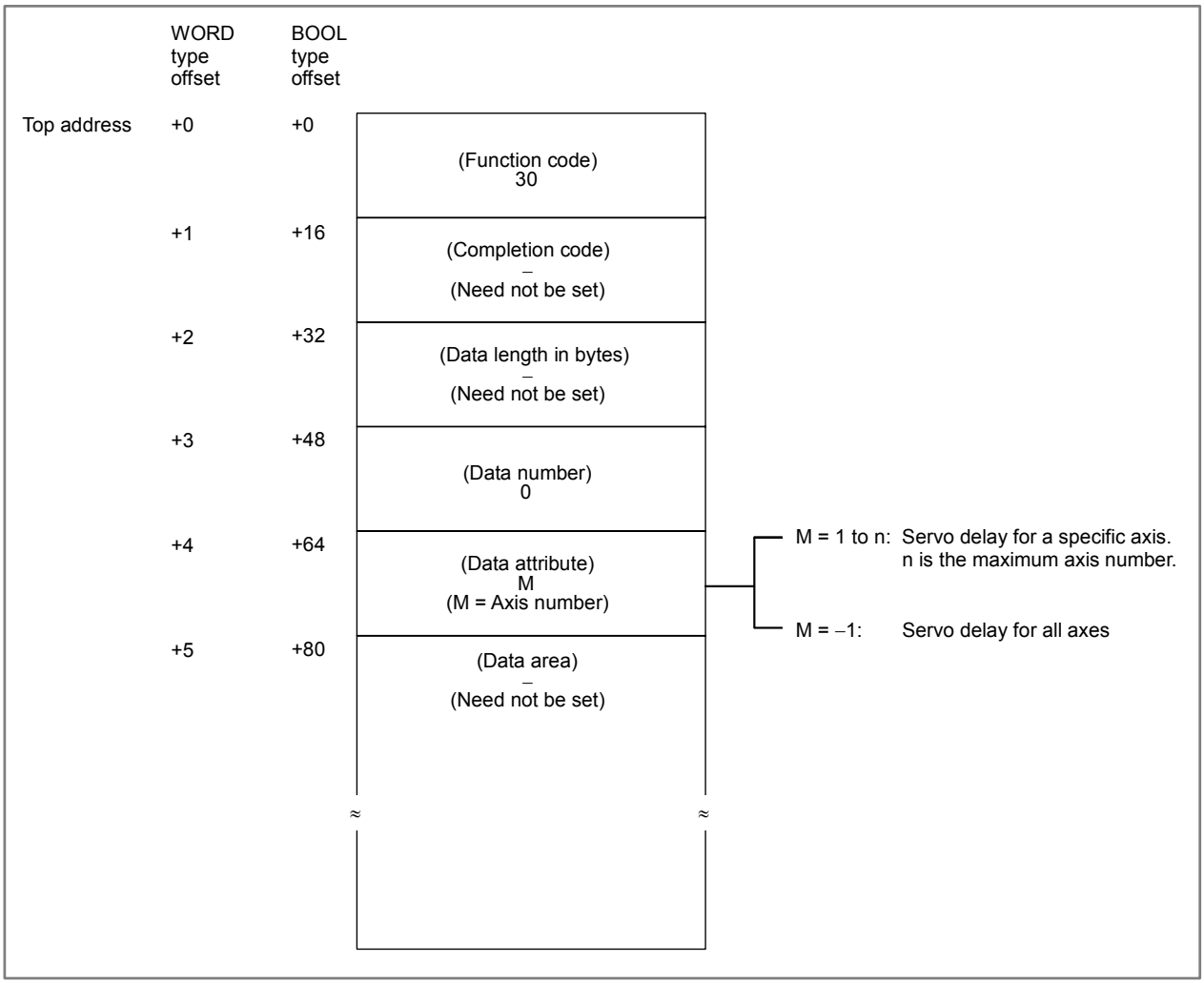
		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

A.5.5 Reading the Servo Delay for Controlled Axes (High-speed Response)

[Description]

The servo delay, which is the difference between the specified coordinates of CNC-controlled axes and the actual servo position, can be read.

[Input data structure]



[Completion codes]

0: The servo delay for the controlled axes have been read normally.
4: The data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 30	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n, n is the number of axes specified.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Servo delay for the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

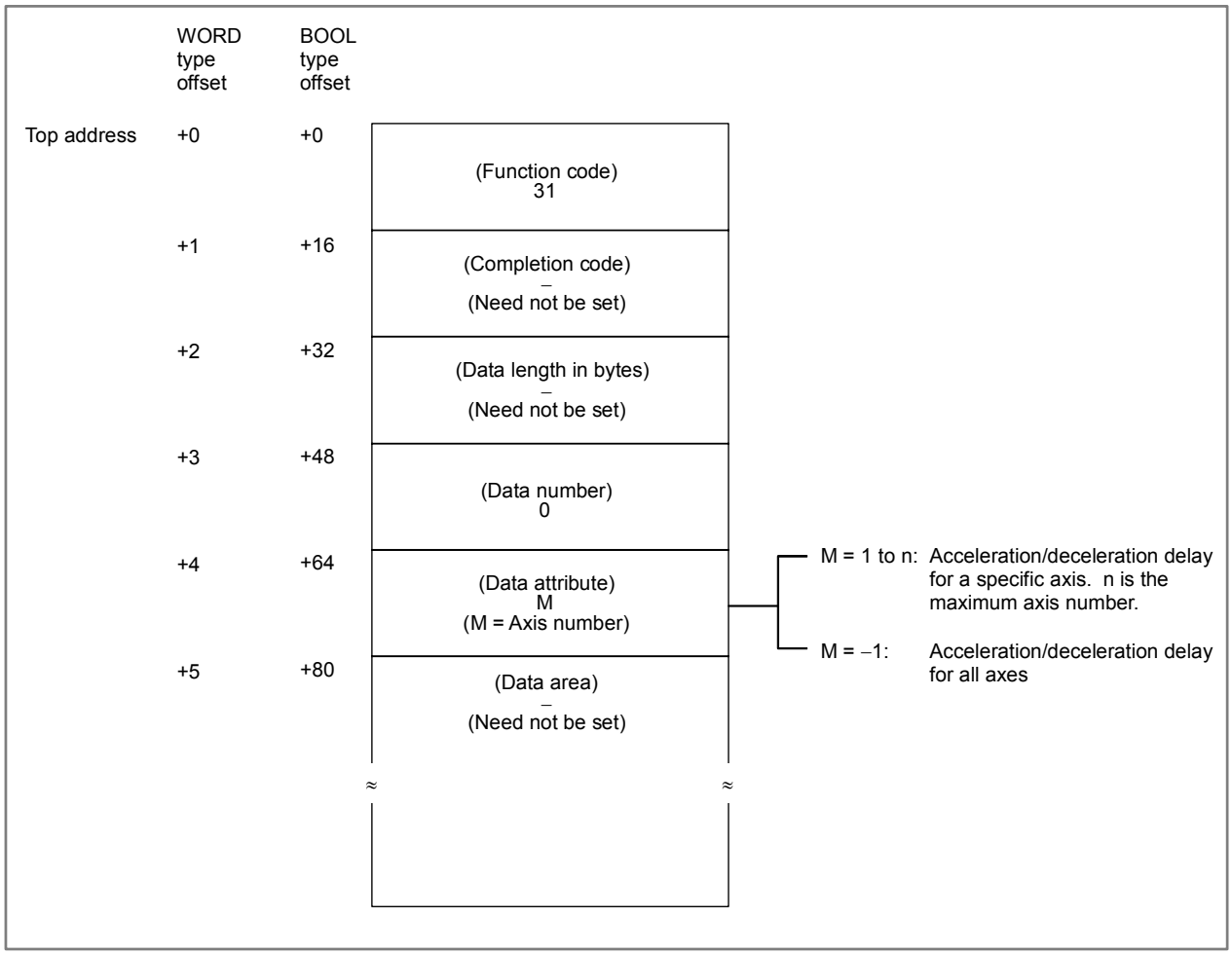
				Value
	+5	+80	Servo delay for the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Servo delay for the second axis (4 bytes)	
	+9	+144	Servo delay for the third axis (4 bytes)	
	+11	+176	Servo delay for the fourth axis (4 bytes)	

A.5.6 Reading the Acceleration/Deceleration Delay on Controlled Axes (High-speed Response)

[Description]

The acceleration/deceleration delay, which is the difference between the coordinates of controlled axes programmed in the CNC and the position after acceleration/deceleration is performed, can be read.

[Input data structure]



[Completion codes]

0: The acceleration/deceleration delay for the control axis has been read normally.

4: The data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 31	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n, n is the number of axes specified.)	
	+3	+48	(Data number) _	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Acceleration/deceleration delay for the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

				Value
	+5	+80	Acceleration/deceleration delay for the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Acceleration/deceleration delay for the second axis (4 bytes)	
	+9	+144	Acceleration/deceleration delay for the third axis (4 bytes)	
	+11	+176	Acceleration/deceleration delay for the fourth axis (4 bytes)	

[Output data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

A.5.7 Reading the Feed Motor Load Current Value (A/D Conversion Data) (High-speed Response)

[Description]

The digital value converted from the load current of the CNC-controlled axis can be read.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 34
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (N: Type of analog voltage)
	+4	+64	(Data attribute) M (M = 1 to 8: Axis specification)
	+5	+80	(Data area) – (Need not be set)

(a) Type of analog voltage (data number)

N	Type of analog voltage
0	(reserved)
2	Load information for the CNC-controlled axes

(b) Specifying a CNC-controlled axis (data attribute)
Specify a CNC-controlled axis number for which the voltage conversion data for the load current is to be read.

⚠CAUTION
There is no general-purpose analog input on *i* series. If you need such a function, use the I/O Link analog input module.

[Completion codes]

0: A/D conversion data has been read normally.
3: The data specified for the data number is invalid.
4: The data specified for the data attribute is invalid, or the specified axis number is greater than the number of controlled axes.
6: No analog input module is connected.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 34	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) N (Input data)	
	+4	+64	(Data attribute) M (Input data)	Value
	+5	+80	A/D conversion data (2 bytes) (For CNC controlled axis load information)	Binary number from 0 to ±6554

(a) A/D conversion data (AD) of CNC controlled axis load information
The load current for the specified CNC controlled axis is converted into analog voltage, the input to the A/D converter to output a digital data.
The value actually set in the AD field is obtained from the following formula:

$$(AD) \times \frac{N}{6554} = \text{Load current } [A_{peak}]$$

AD = A/D conversion data [Value read by the window function (±)]
N = Nominal current limit for the amplifier corresponding to the motor

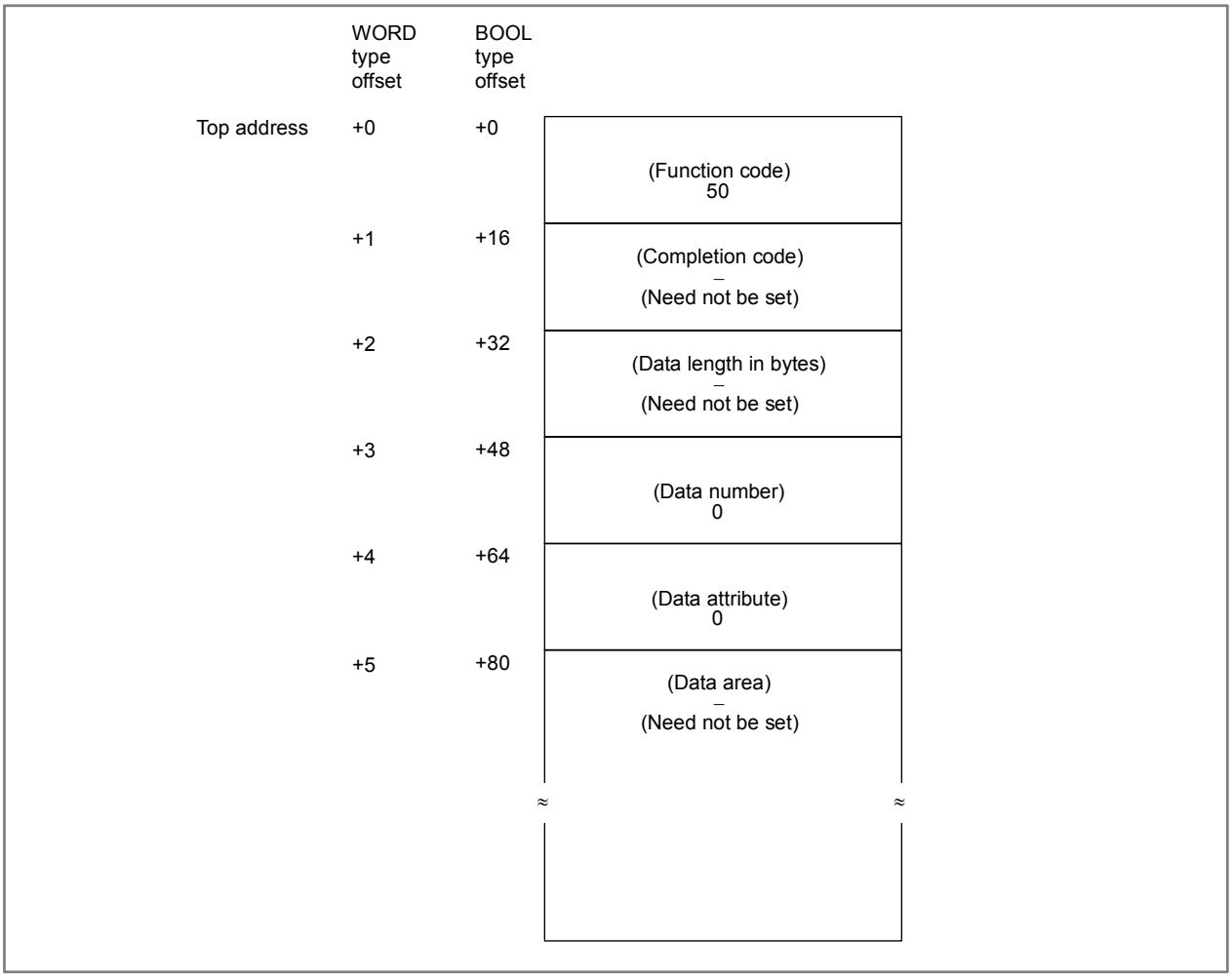
NOTE
For the nominal current limits, see the descriptions of the control motor.

A.5.8 Reading the Actual Spindle Speed (High-speed Response)

[Description]

The actual speed of the spindle can be read from the CNC.

[Input data structure]



[Completion codes]

0: The actual speed of the spindle has been read normally.

[Output data structure]

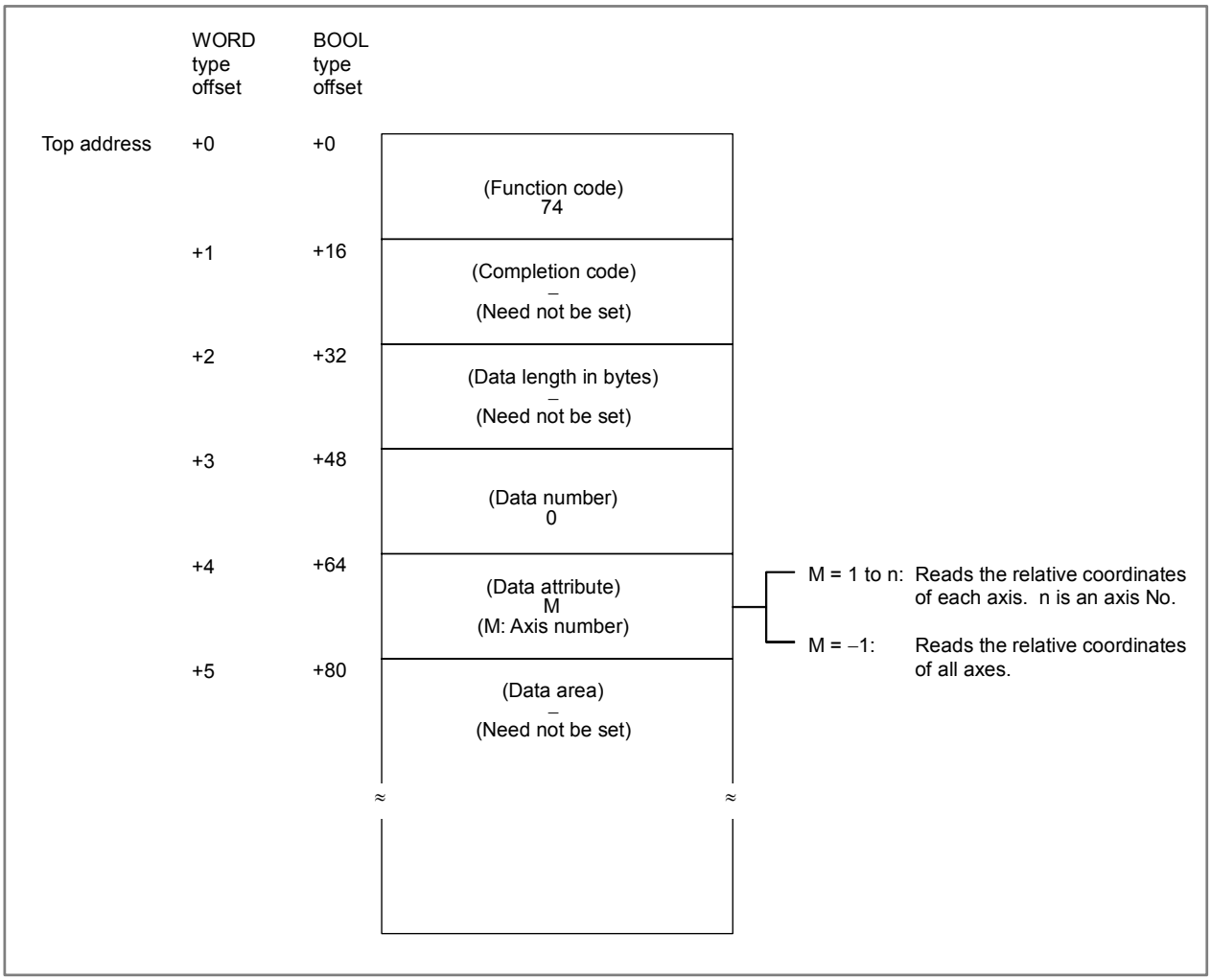
	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 50	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	Actual spindle speed	Unsigned binary <Data unit> min^{-1}

A.5.9 Reading the Relative Position on a Controlled Axis (High-speed Response)

[Description]

The relative coordinates of the machine moving along an axis controlled by the CNC can be read.

[Input data structure]



[Completion codes]

0: The relative coordinates on the controlled axis have been read normally.

4: The specified data attribute is invalid. That is, a value other than –1 and 1 to n (number of axes) was specified, or the specified axis No. was greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 74	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n. n is the number of specified axes.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Relative coordinates on the specified controlled axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

				Value
	+5	+80	Relative coordinates on the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Relative coordinates on the second axis (4 bytes)	
	+9	+144	Relative coordinates on the third (4 bytes)	
	+11	+176	Relative coordinates on the fourth axis (4 bytes)	

[Output data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

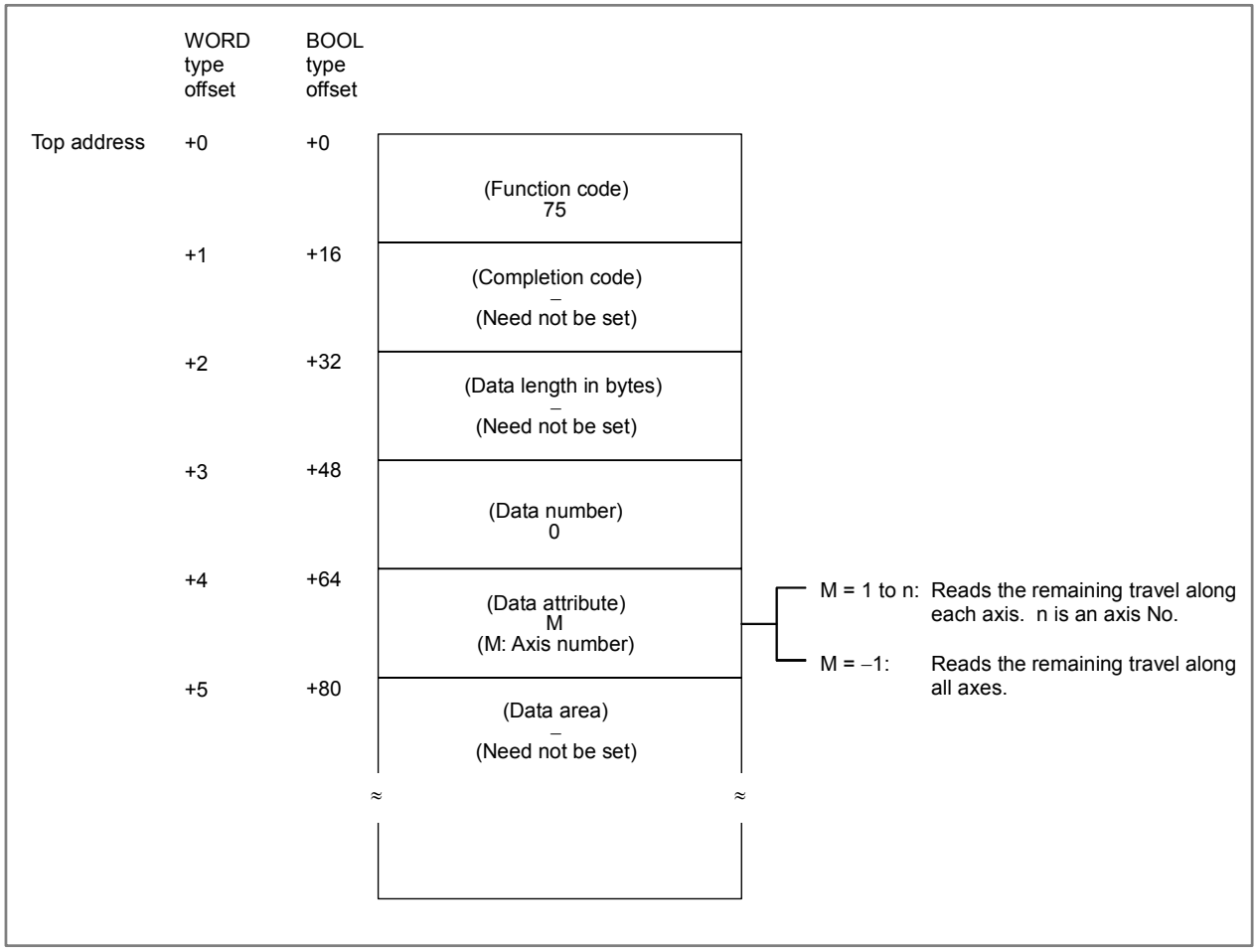
Double values can be read for a machining center system or when radius specification is used for the relevant axis of a lathe system.

A.5.10 Reading the Remaining Travel (High-speed Response)

[Description]

The remaining travel of the machine along an axis controlled by the CNC can be read. The read value equals the remaining travel indicated on the current position display screen on the CNC. (This screen can be called by pressing the function key <POS>.)

[Input data structure]



[Completion codes]

0: The remaining travel along the controlled axis has been read normally.

4: The specified data attribute is invalid. That is, a value other than –1 and 1 to n (number of axes) was specified, or the specified axis No. was greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 75	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 4*n. n is the number of specified axes.)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Remaining travel along the specified controlled axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)

When the number of controlled axes is 4

				Value
	+5	+80	Remaining travel along the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement.)
	+7	+112	Remaining travel along the second axis (4 bytes)	
	+9	+144	Remaining travel along the third axis (4 bytes)	
	+11	+176	Remaining travel along the fourth axis (4 bytes)	

[Output data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.0005	0.00005
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.00005	0.000005

In the above table, when a machining center system is used or the radius is specified for the corresponding axis of the lathe system, double the value is read.

A.5.11 Reading Actual Spindle Speeds (High-speed Response)

(1) Actual spindle speed

[Description]

This function reads the actual speed of the No.1 to No.4 serial spindles.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 138
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) M (M = Spindle number)
	+5	+80	(Data area) – (Need not be set)

M = 1 to n: Read spindles on each axis.
(n is the spindle number.)
–1: Read spindles on No.1 and No.2 axes
–2: Read spindles on No.1 to No.3 axes
–3: Read spindles on No.1 to No.4 axes

[Completion codes]

0: The actual spindle speed was read successfully.
4: The spindle speed in 'Data Attribute' has wrong values, that is, a value outside of the range -1 to -(n - 1) or 1 to n (n: number of spindles).

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 138	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) L (L = 4 × n)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Actual speed of specified spindle	Signed binary <Data unit> min^{-1}
	+7	+112		

Or, 4 spindles:

				Value
	+5	+80	Actual speed of No.1 spindle	Signed binary <Data unit> min^{-1}
	+7	+112	Actual speed of No.2 spindle	
	+9	+144	Actual speed of No.3 spindle	
	+11	+176	Actual speed of No.4 spindle	
	+13	+208		

(2) Position coder-less actual spindle speed

[Description]

This function reads the actual spindle speed (position coder-less actual spindle speed) obtained by calculating the spindle motor speed of the No.1 to No.4 serial spindles.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 138
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) M (M = Spindle number + 10)
	+5	+80	(Data area) – (Need not be set)
	+6	+96	

M = 11 to
(10 + n): Read spindles on each axis.
(n is the spindle number.)
−11: Read spindles on No.1 and No.2 axes
−12: Read spindles on No.1 to No.3 axes
−13: Read spindles on No.1 to No.4 axes

[Completion codes]

0: The actual spindle speed was read successfully.
4: The spindle speed in 'Data Attribute' has wrong values, that is, a value outside of the range -11 to -(9 + n) or 11 to (10 + n) (n: number of spindles).

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 138	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) L (L = 4 × n)	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Position coder-less actual spindle speed	Signed binary <Data unit> min^{-1}
	+7	+112		

Or, 4 spindles:

				Value
	+5	+80	Position coder-less actual No.1 spindle speed	Signed binary <Data unit> min^{-1}
	+7	+112	Position coder-less actual No.2 spindle speed	
	+9	+144	Position coder-less actual No.3 spindle speed	
	+11	+176	Position coder-less actual No.4 spindle speed	
	+13	+208		

A.5.12 Entering Torque Limit Data for the Digital Servo Motor (Low-speed Response)

[Description]

Torque limit values for the digital servo motor can be entered.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 152	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) M (M: 1 to n)	M = 1 to n: Axis No.
				Value
	+5	+80	Torque limit data (1 byte) The high-order byte is always set to 0.	Unsigned binary <Unit: %> Values from 0 to 255 correspond to 0% to 100%.

⚠CAUTION

Calculate the torque limit data assuming that the short time rated value is 100%.

Example: To specify a torque limit of 50%, enter 128.

[Completion codes]

0: Torque limit data has been entered normally.
4: The specified data attribute is invalid. That is, a value other than 1 to n (number of axes) was specified, or the specified axis No. was greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 152	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 2 (Input data)	
	+3	+48	(Data number) 1 (Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Torque limit data (1 byte): Input data The high-order byte is always set to 0.	Unsigned binary <Unit: %> Values from 0 to 255 correspond to 0% to 100%.

A.5.13 Reading Load Information of the Spindle Motor (Serial Interface) (High-speed Response)

[Description]

Load information of the serial spindle can be read.
The equation to normalize the load information is shown below

$$\text{Load}(\%) = \frac{L}{32767} \times \lambda$$

L: Data read from the window

λ: The percentage of the maximum output of the motor to the continuous rated output of the motor (When the maximum output is 180% and the continuous rated output is 100%, the percentage is 180.)

⚠CAUTION
The "λ" is equal to the value of parameter No. 4127.

[Input data structure]

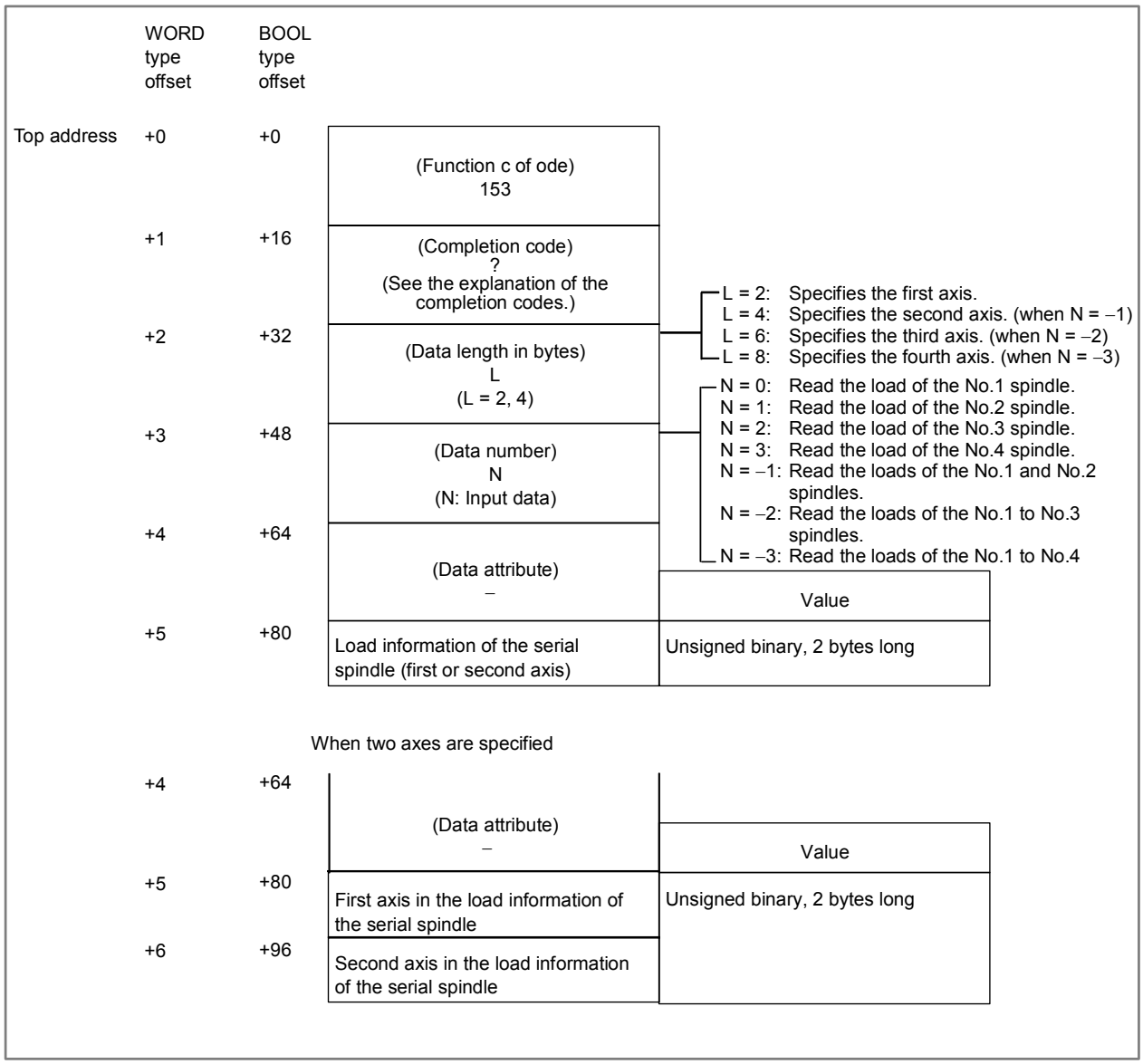
	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 153
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (Need not be set)
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) – (Need not be set)
	≈		≈

N = 0: Read the load of the No.1 spindle.
1: Read the load of the No.2 spindle.
2: Read the load of the No.3 spindle.
3: Read the load of the No.4 spindle.
–1: Read the loads of the No.1 and No.2 spindles.
–2: Read the loads of the No.1 to No.3 spindles.
–3: Read the loads of the No.1 to No.4 spindles.

[Completion codes]

0: Load information of the serial spindle has been read normally.

[Output data structure]



When three axes are specified

		(Data attribute) –	Value
+4	+64		
+5	+80	First axis in the load information of the serial spindle	Unsigned binary, 2 bytes long
+6	+96	Second axis in the load information of the serial spindle	
+7	+112	Third axis in the load information of the serial spindle	

When four axes are specified

		(Data attribute) –	Value
+4	+64		
+5	+80	First axis in the load information of the serial spindle	Unsigned binary, 2 bytes long
+6	+96	Second axis in the load information of the serial spindle	
+7	+112	Third axis in the load information of the serial spindle	
+8	+128	Fourth axis in the load information of the serial spindle	

A.5.14 Reading the servo data of the control axes (High-speed Response)

[Description]

This function can read the following information of servo motor.

- Actual speed (rev / min)
- Thermal simulation data (OVC data)
- Torque command

The read "Torque command" is normalized from -6554 to 6554. The value 6554 corresponds to the maximum current of servo amplifier.
Applying the following formula to this value, you can determine the ratio of the torque command to the maximum current of amplifier.

Ratio (%) = [data] * 100 / 6554

Applying the following formula, you can also determine the torque command (Apeak).

Torque command (Apeak) = [data] * [maximum current of amplifier] / 6554

[Input data structure]

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (Need not to be set)	
	+2	+32	(Data length in bytes) - (Need not to be set)	
	+3	+48	(Data number) N	N = 11: Actual speed (min^{-1}) 14: Thermal simulation data (OVC data) 15: Torque command
	+4	+64	(Data attribute) M (M=Axis number)	M = 1 to n: Axis number -1: All axes
	+5	+80	(Data area) - (Need not to be set)	

[Completion codes]

0: Normal completion
3: The data number is invalid.
4: Data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes).

[Output data structure]

Reading the actual speed for one axis (N=11)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=4*n)	The n means the number of read parameters.
	+3	+48	(Data number) N=11 (Input data)	
	+4	+64	(Data attribute) M (Input data)	
	+5	+80	Actual speed (4 bytes)	Signed binary format in 4 bytes length Data unit: min^{-1}
	+7	+112		

[Output data structure]

Reading the actual speed for all axes (N=11, Example for 3 controlled-axes)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=4*n)	The n means the number of read parameters.
	+3	+48	(Data number) N=11 (Input data)	
	+4	+64	(Data attribute) -1	
	+5	+80	Actual speed for 1st axis (4 bytes)	Signed binary format in 4 bytes length Data unit: min^{-1}
	+7	+112	Actual speed for 2nd axis (4 bytes)	
	+9	+144	Actual speed for 3rd axis (4 bytes)	
	+11	+176		

[Output data structure]

Reading the thermal simulation data for one axis (N=14)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=2*n)	
	+3	+48	(Data number) N=14 (Input data)	
	+4	+64	(Data attribute) M (Input data)	
	+5	+80	Thermal simulation data (2 bytes)	Signed binary format in 2 bytes length Data unit: % The OVC alarm will happen when this value is 100 %.
	+6	+96		

[Output data structure]

Reading the thermal simulation data for all axes (N=14, Example for 3 controlled-axes)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=2*n)	
	+3	+48	(Data number) N=14 (Input data)	
	+4	+64	(Data attribute) -1	
	+5	+80	Thermal simulation data for 1st axis (2 bytes)	Signed binary format in 2 bytes length Data unit: % The OVC alarm will happen when this value is 100 %.
	+6	+96	Thermal simulation data for 2nd axis (2 bytes)	
	+7	+112	Thermal simulation data for 3rd axis (2 bytes)	
	+8	+128		

[Output data structure]

Reading the torque command for one axis (N=15)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=2*n)	
	+3	+48	(Data number) N=15 (Input data)	
	+4	+64	(Data attribute) M (Input data)	
	+5	+80	Torque command (2 bytes)	Signed binary format in 2 bytes length Data: This data is nomalized from -6554 to 6554.
	+6	+96		

[Output data structure]

Reading the torque command for all axes (N=15, Example for 3 controlled-axes)

	WORD type offset	BOOL type offset		
Top Address	+0	+0	(Function code) 207	
	+1	+16	(Completion code) - (See above description)	
	+2	+32	(Data length in bytes) L (L=2*n)	
	+3	+48	(Data number) N=15 (Input data)	
	+4	+64	(Data attribute) -1	
	+5	+80	Torque command for 1st axis (2 bytes)	Signed binary format in 2 bytes length Data: This data is nomalized from -6554 to 6554.
	+6	+96	Torque command for 2nd axis (2 bytes)	
	+7	+112	Torque command for 3rd axis (2 bytes)	
	+8	+128		

A.5.15 Reading the Estimate Disturbance Torque Data (High-speed Response)

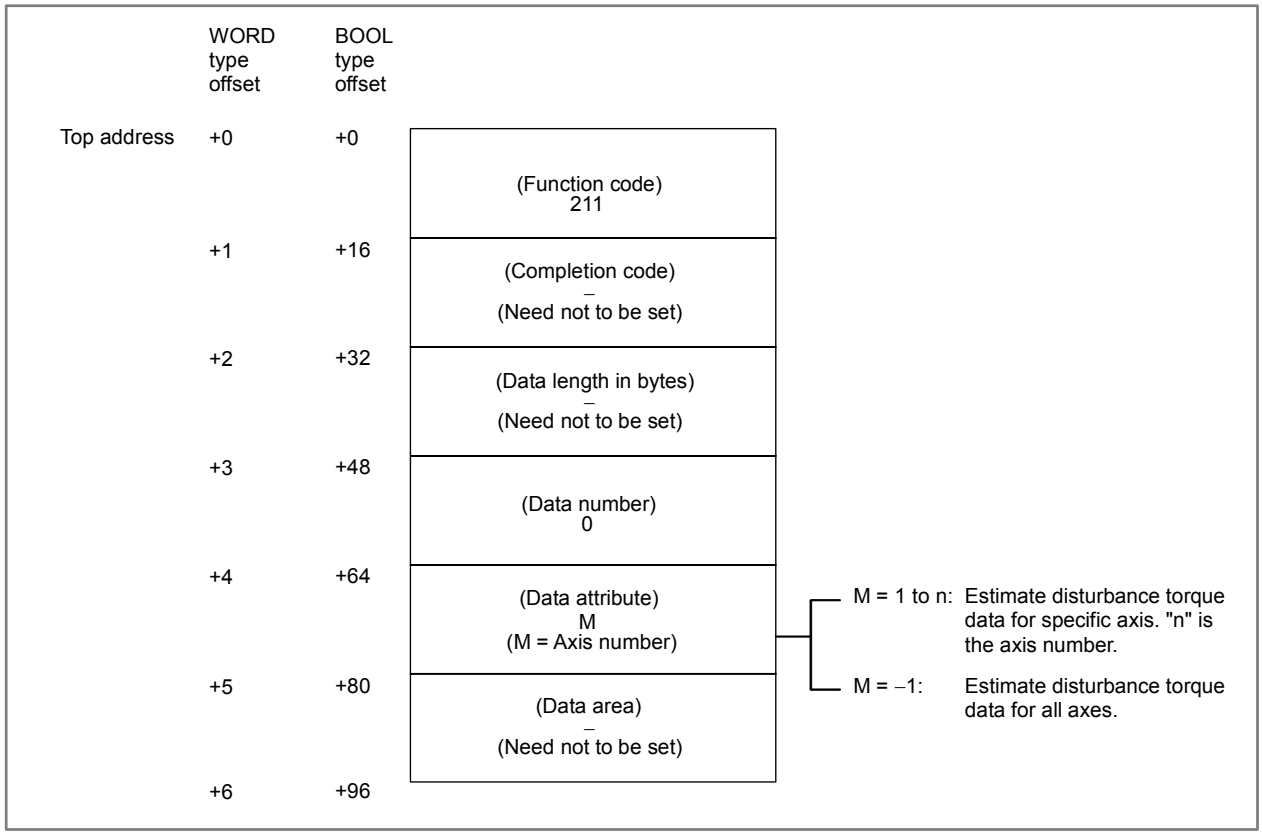
> **⚠CAUTION**
> The unexpected disturbance torque detection function option is required. For detailed settings of parameters and so forth, refer to the description of unexpected disturbance torque detection in the connection manual (functions).

(1) Servo axis

[Description]

The load torques except a necessary torque for acceleration / deceleration of the digital servo axis are read.

[Input data structure]



[Completion codes]

0: The estimate disturbance torque data have been read normally.
4: The data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 211	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) L (L = 2 × n, n is the number of axes specified.)	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Estimate disturbance torque data for the controlled axis specified (2 bytes)	(A negative value is represented in 2's complement.)
	+6	+96		

When the number of controlled axes is 4

				Value
	+5	+80	Estimate disturbance torque data for first axis (2 bytes)	Signed binary 2's complement.)
	+6	+96	Estimate disturbance torque data for second axis (2 bytes)	
	+7	+112	Estimate disturbance torque data for third axis (2 bytes)	
	+8	+128	Estimate disturbance torque data for fourth axis (2 bytes)	
	+9	+144		

(2) Spindle axis

[Description]

The load torques except a necessary torque for acceleration / deceleration of the serial spindle axis are read.

[Input data structure]

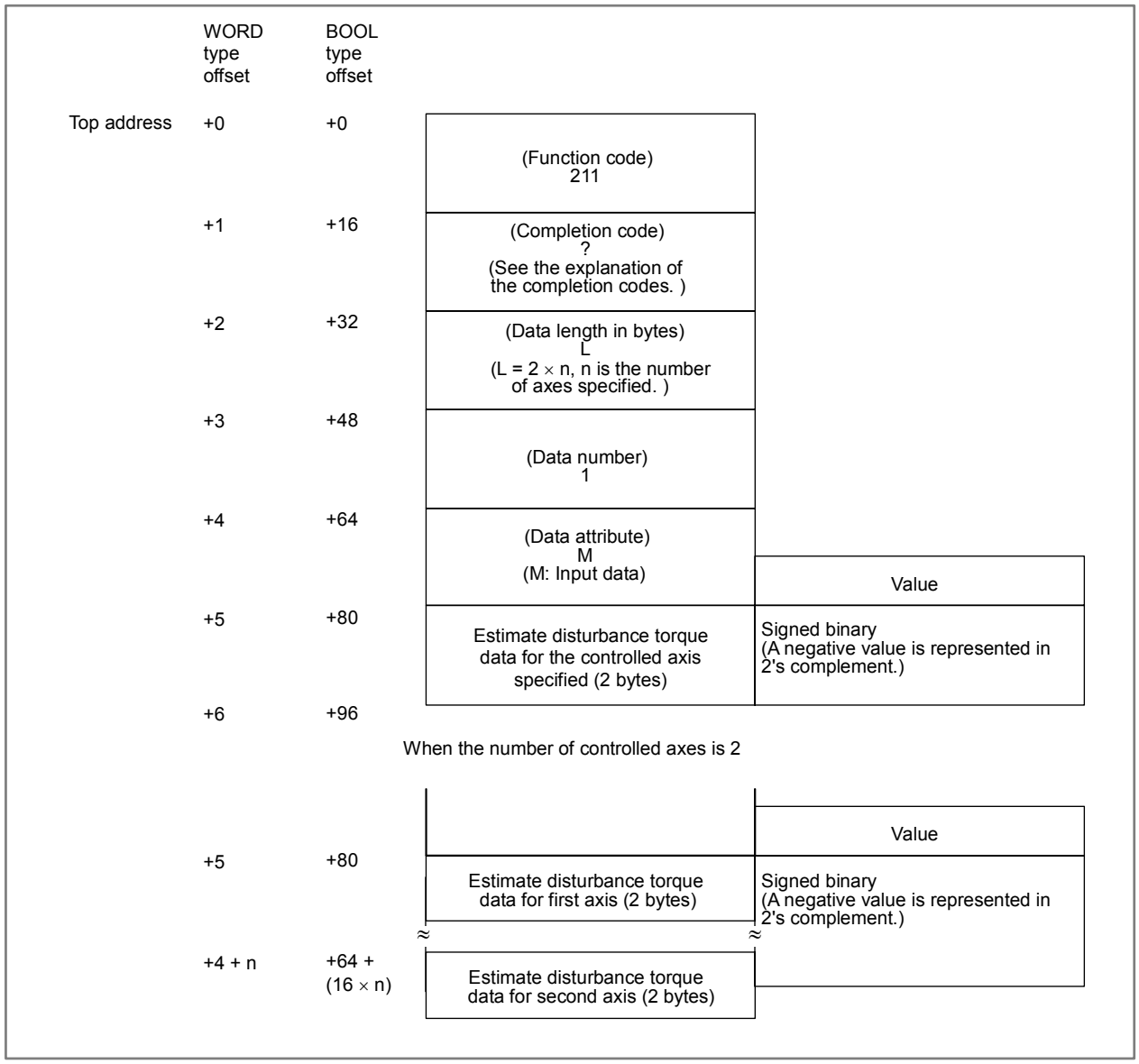
	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 211
	+1	+16	(Completion code) – (Need not to be set)
	+2	+32	(Data length in bytes) – (Need not to be set)
	+3	+48	(Data number) 1
	+4	+64	(Data attribute) M (M = Spindle number)
	+5	+80	(Data area) – (Need not to be set)
	+6	+96	

M = 1 to n: Read the load of each spindle.
(n is the spindle number.)
–1: Read the loads of the No.1 and No.2 spindles.
–2: Read the loads of the No.1 to No.3 spindles.
–3: Read the loads of the No.1 to No.4 spindles.

[Completion codes]

0: The estimate disturbance torque data have been read normally.
4: The data specified as the data attribute is invalid because it is neither -1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.

[Output data structure]



A.5.16 Reading Fine Torque Sensing Data (Statistical Calculation Results) (High-speed Response)

[Description]

This function reads the statistical calculation results (average value, maximum value, and distribution) in the fine torque sensing function.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 226
	+1	+16	(Completion code) (Need not be set)
	+2	+32	(Data length in bytes) (Need not be set)
	+3	+48	(Data number) N (N=Axis number)
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) (Need not be set)
	+6	+96	

N=1 to n: Read spindles on each axis.
(n is the spindle number.)

-1: Read all axes

[Completion codes]

0: The statistical calculation results were read successfully.
3: The fine torque sensing data in 'Data Attribute' has a wrong value, that is, a value outside of the range -1 or 1 to n (n: number of spindles).
6: The fine torque sensing option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 226	
	+1	+16	(Completion code) ? (See the above explanation.)	
	+2	+32	(Data length in bytes) L L=6×n	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Average value of target axis	Signed binary
	+6	+96	Maximum value of target axis	
	+7	+112	Distribution of target axis	
	+8	+128		

Or, 4 controlled axes:

				Value
	+5	+80	Average value of target axis 1	Signed binary (Output only for number of axes specified to parameter No. 6390 to 6363)
	+6	+96	Maximum value of target axis 1	
	+7	+112	Distribution of target axis 1	
	+8	+128	Average value of target axis 2	
	+9	+144	Maximum value of target axis 2	
	+10	+160	Distribution of target axis 2	
			:	
			:	
			:	
	+16	+256	Average value of target axis 4	
	+17	+272		

A.5.17 Reading Fine Torque Sensing Data (Store Data) (High-speed Response)

(1) Store counter

[Description]

This function reads the number of stored torque data items.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) ? (Need not be set)	
	+2	+32	(Data length in bytes) 2 (Need not be set)	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) M (M=Tool number)	M= 0 : Latest data counter = 1 : Sample data counter
	+5	+80	(Data area) - (Need not be set)	

[Completion codes]

0: The store counter was read successfully.
3: Incorrect data number, that is, a value other than 0 is specified.
4: The fine torque sensing data in 'Data Attribute' has wrong values, that is, a value other than 0 or 1.
6: The fine torque sensing option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) — (See the explanation above.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) 0 (Entered data)	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Store counter value	Unsigned binary
	+7	+112		

(2) Stored torque data (latest data)

[Description]

This function reads the latest stored data among stored torque data.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) — (Need not be set)	
	+2	+32	(Data length in bytes) — (Need not be set)	
	+3	+48	(Data number) N (N= Axis number + 10)	N : Number of axis to be read + 100
	+4	+64	(Data attribute) M (M=Data type)	M= 0 : Latest data counter = 1 : Sample data counter
	+5	+80	(Data area) — (Need not be set)	

[Completion codes]

0: The stored torque data (latest data) was read successfully.
3: Incorrect data number, that is, a value other than 11 to (10+n) (n: number of spindles) is specified.
4: The fine torque sensing data in 'Data Attribute' has a wrong value, that is, a value other than 0 or 1.
6: The fine torque sensing option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) L (L=2 or 0)	
	+3	+48	(Data number) N (Entered data)	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Latest stored data	Signed binary
	+6	+96		

NOTE

1 When data has not been stored, data is not output and processing ends successfully with L set to 0.

2 When sample data is selected by data attribute, the sample data corresponding to the latest stored data is output.

Example) When 10000 sample data items (data numbers 0 to 9999) and latest data items 5000 (data numbers 0 to 4999) are stored, data number 4999 in the latest data is output when data attribute M is set to "0", and data number 4999 in the sample data is output when data attribute M is set to "1".

3 When sample data is selected by data attribute, and there is no sample data corresponding to the latest stored data, data is not output, and processing ends successfully with L set to 0.

Example) When 5000 sample data items (data numbers 0 to 4999) and 10000 latest data items (data numbers 0 to 9999) are stored, data is not output, and processing ends successfully with L set to 0 when data attribute M is set to "1".

(3) Stored torque data (any data)

[Description]

This function reads the arbitrary data among stored torque data.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) ? (Need not be set)	
	+2	+32	(Data length in bytes) 6	
	+3	+48	(Data number) N (N= Axis number)	N=1 to n: Axis number (n is the spindle number.)
	+4	+64	(Data attribute) M (M=Data type)	M =0 : Latest data =1 : Sample data
	+5	+80	Data number n (Entered data)	
	+7	+112	Number of data items I (Entered data)	
	+8	+128	Data of number n	
	+9	+144	Data of number n+1	
			:	
	+10	+160		

NOTE

The valid range of data number n is calculated as follows:

$$0 \le n \le (524288 \times \frac{1}{a} \times \frac{1}{b}) - 1$$

where,

$$a = \begin{cases} 1: \text{Number of target axes 1} \\ 2: \text{Number of target axes 2} \\ 4: \text{Number of target axes 3 and 4} \end{cases}$$

$$b = \begin{cases} 1: \text{Sample data store function OFF} \\ 2: \text{Sample data store function ON} \end{cases}$$

The valid range of number of data items I is calculated as follows:

$$1 \le I \le 20$$

[Completion codes]

0: The stored torque data (any data) was read successfully.
2: Incorrect data length, that is, a value other than 6 is specified.
3: Incorrect data number, that is, a value other than 11 to (10+n) (n: number of spindles) is specified.
4: The fine torque sensing data in 'Data Attribute' has a wrong value, that is, a value other than 0 or 1.
5: Incorrect data area is specified. See Note for details of value ranges.
6: The fine torque sensing option has not been added on.

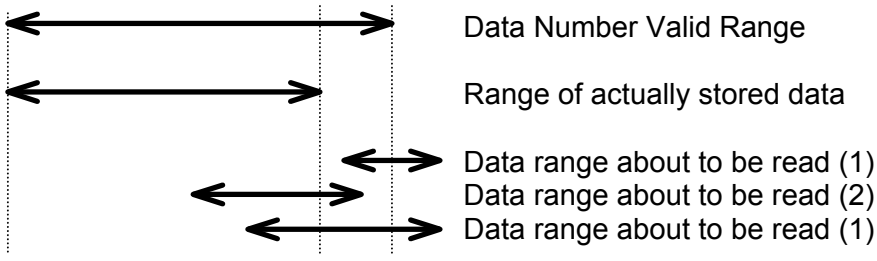
[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 232	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) L (L=6 + number of data items I × 2)	
	+3	+48	(Data number) N (Entered data)	
	+4	+64	(Data attribute) M (Entered data)	Value
	+5	+80	Data number n (Entered data)	Signed binary
	+7	+112	Number of data items I (Entered data)	
	+8	+128	Distribution of target axes	
	+9	+144	Number n data	
	+10	+160	Number n+1 data	
	+11	+176	Number n+2 data	
	+12	+192	:	
			:	
			:	
			Number n+I - 1 data	

NOTE

1 When the number of actually stored data items is exceeded even though data number n is in the valid range, data is not output and processing ends successfully by number of data items I set to 0.
Example) When the number of target axes is 2, and the sample data store function is enabled (parameter No.6350#2=1), data numbers 0 to 131071 are valid. However, if an attempt is made to read (example (1) in figure below) data from data number n = 131020 when the number of actually stored data items is 131000 (data numbers 0 to 130999), data is not output, and the number of data items I becomes 0.

2 When data number n is within the number of actually stored data items, and (n+I-1) exceeds the number of actually stored data items, data of the stored data items is output, and processing ends successfully. In this case, number of data items I is updated to the number of data items that was output.
Example) If an attempt is made to read (example (2) in figure below) number of data items I (120) from data number 130900 under the same conditions as in the example above, the data of data numbers 130900 to 130999 is output, and number of data items I becomes 100. Also, if an attempt is made to read (example (3) in figure below) number of data items I (120) from data number 130999 under the same conditions as in the example above, the data of data numbers 130990 to 130999 is output, and number of data items becomes 10.



A.5.18 Presetting the Relative Coordinate (Low-speed Response)

[Description]

The preset data is set to the relative coordinate controlled by CNC. If 0 is set as preset data it becomes to origin.
But it is impossible to write the value of preset data to the transferring axis. In the case of the preset of relative coordinate of all axes is executed by using this function, if only one axis is transferring, the preset of relative coordinate cannot be executed, neither.

[Input data structure]

Case of writing data on each axis.

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 249	
	+1	+16	(Completion code) – (Need not be set.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) 0	Value
	+4	+64	(Data attribute) M (M: Axis number)	M=1 to n: Write data on each (n is the axis number)
	+5	+80	Value of relative coordinate for the controlled axis specified (4 bytes)	Signed binary (A negative value is represented in 2's complement)
	+6	+96		

[Input data unit]

		Input system	Increment system	
			IS-B	IS-C
Machining center system		mm, deg	0.001	0.0001
		inch	0.0001	0.00001
Lathe system	Radius specification	mm, deg	0.001	0.0001
	Diameter specification		0.001	0.0001
	Radius specification	inch	0.0001	0.00001
	Diameter specification		0.0001	0.00001

Case of writing data on all axes (controlled axes are 4).

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 249	
	+1	+16	(Completion code) – (Need not be set.)	
	+2	+32	(Data length in bytes) 16	
	+3	+48	(Data number) 0	Value
	+4	+64	(Data attribute) M (M = −1)	−1 must be set
	+5	+80	Value of relative coordinate for the first axis (4 bytes)	Signed binary (A negative value is represented in 2's complement)
	+7	+112	Value of relative coordinate for the second axis (4 bytes)	
	+9	+144	Value of relative coordinate for the third axis (4 bytes)	
	+11	+176	Value of relative coordinate for the fourth axis (4 bytes)	

[Completion codes]

0: Success to set the value of relative coordinate.
4: Data specified for the data attribute is invalid because it is neither –1 nor a value from 1 to n (n is the number of axes). Alternatively, the specified axis number is greater than the number of controlled axes.
5: Relative coordinate is out of range.
13: Axis is moving now.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 249
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) L (Same as input data)
	+3	+48	(Data number) 0 (Same as input data)
	+4	+64	(Data attribute) M (M: Input data)
	+5	+80	Value of relative coordinate (4*n bytes)

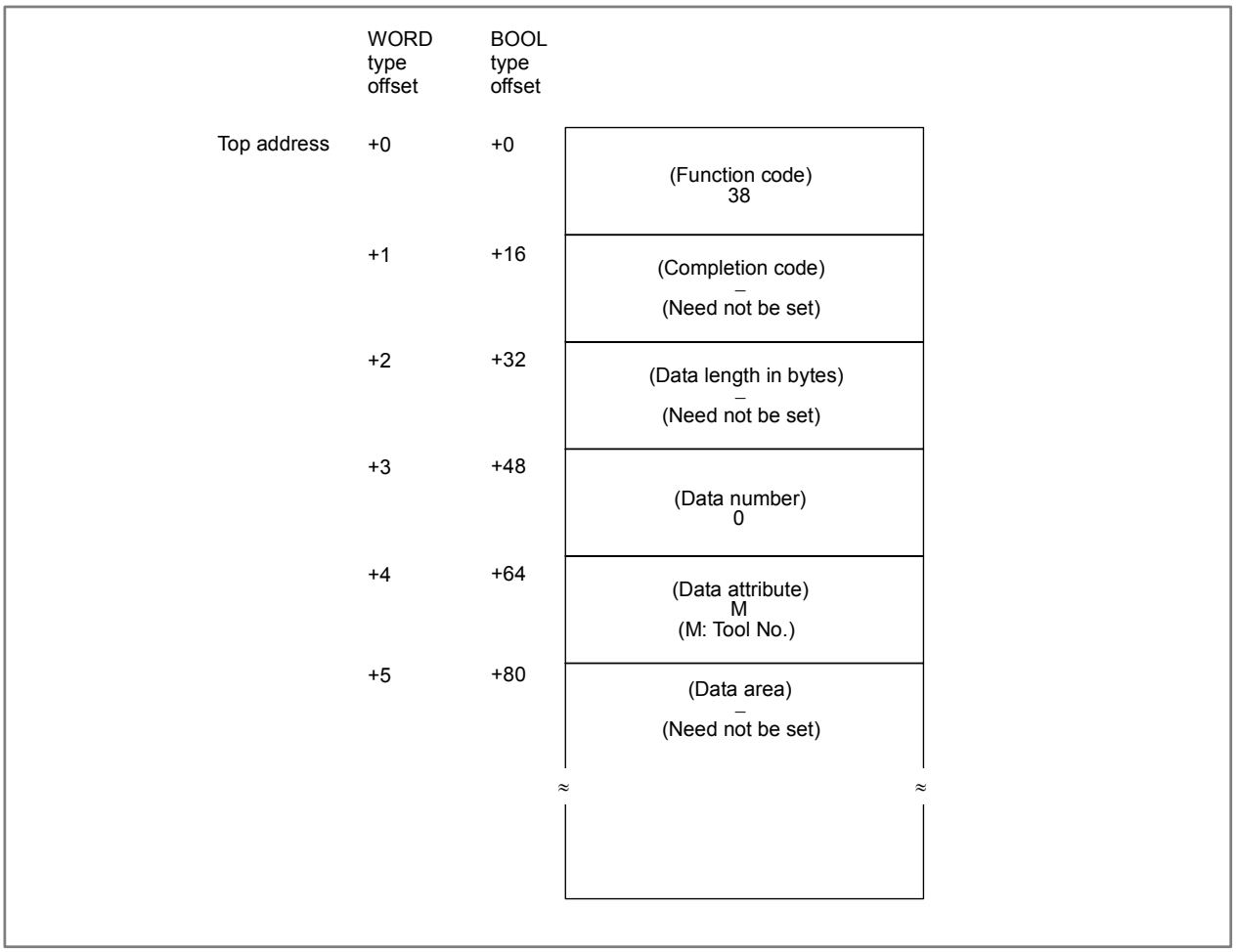
A.6 TOOL LIFE MANAGEMENT FUNCTION

A.6.1 Reading The Tool Life Management Data (Tool Group Number) (High-speed Response)

[Description]

By specifying a tool number, the number of the tool group to which the specified tool belongs can be read from tool life management data.

[Input data structure]



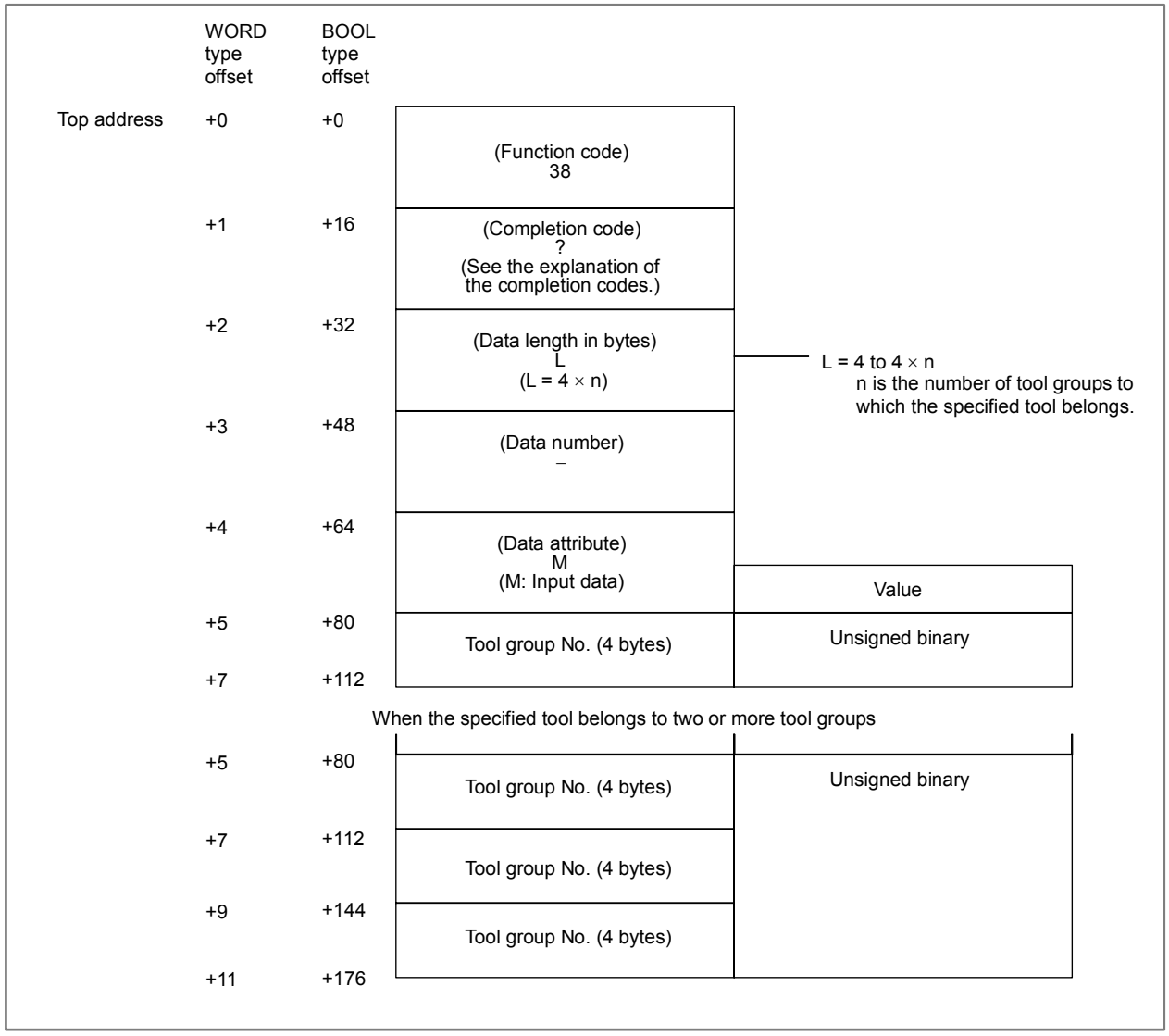
⚠CAUTION

If 0 is specified for the tool No., the number of the tool group currently used is read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.
If the same tool belongs to two or more tool groups, the number of all tool groups to which the tool belongs are displayed.

[Completion codes]

0: The tool group number has been read normally.
4: The value specified for the data attribute is invalid.
5: The specified tool number was not found.
6: The tool life management option has not been added.

[Output data structure]



NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.2 Reading Tool Life Management Data (Number of Tool Groups) (High-speed Response)

[Description]

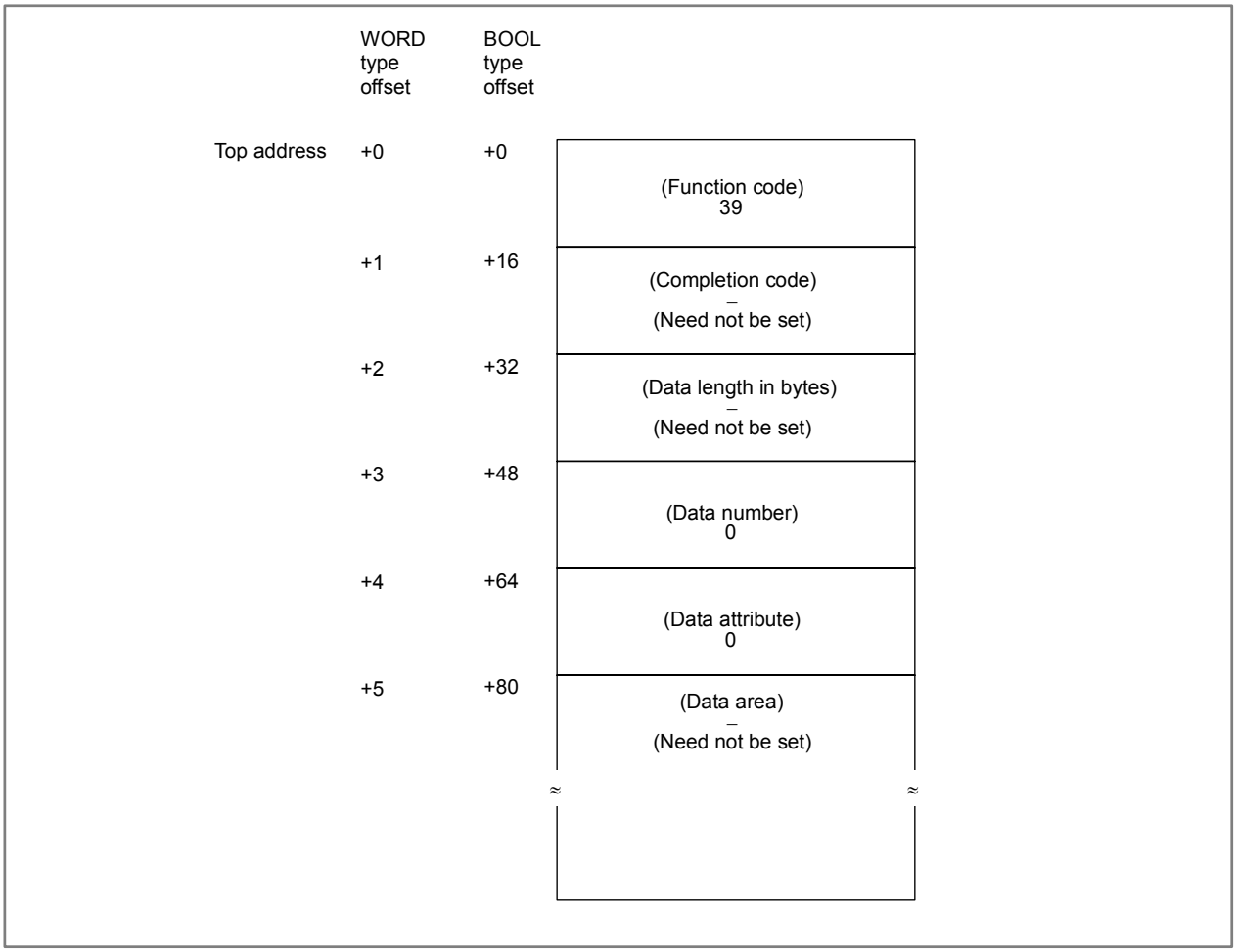
The number of tool groups in tool life management data can be read. The number of tool groups that can be registered varies depending on the setting of parameter 6800 of the CNC, as indicated in the following table.

Parameter 6800

GS2	GS1	Number of tool groups The numbers in parentheses apply when the additional option is used	
		M series	T series
0	0	1 to 16 (1 to 64)	1 to 16 (1 to 16)
0	1	1 to 32 (1 to 128)	1 to 32 (1 to 32)
1	0	1 to 64 (1 to 256)	1 to 64 (1 to 64)
1	1	1 to 128 (1 to 512)	1 to 16 (1 to 128)

M series: For Machining Centers T series: For Lathes

[Input data structure]



[Completion codes]

0: The number of tool group numbers has been read successfully.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 39	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) –	
	+4	+64	(Data attribute) –	Value
	+5	+80	Number of tool groups (4 bytes)	Unsigned binary

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.3 Reading Tool Life Management Data (Number of Tools) (High-speed Response)

[Description]

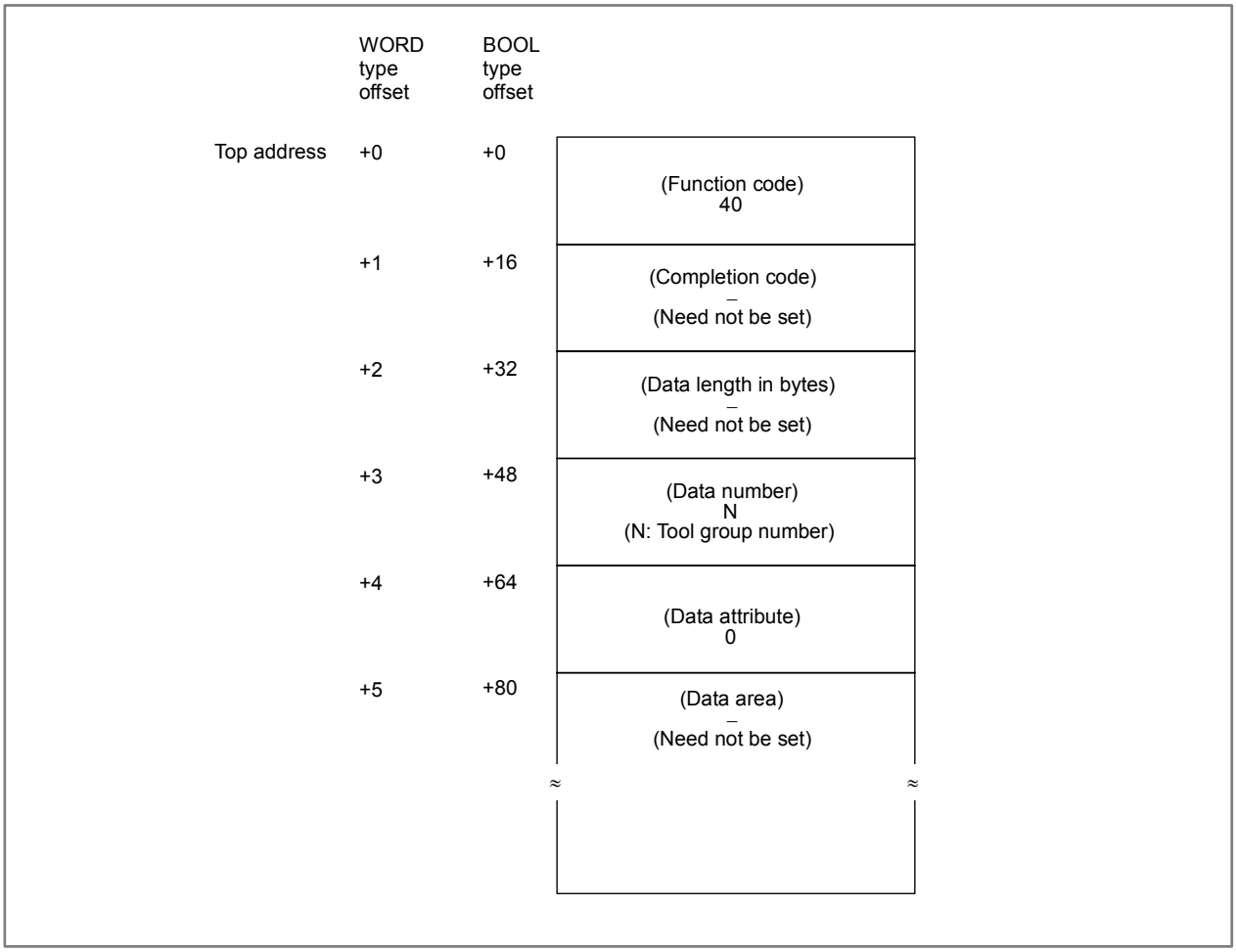
By specifying a tool group number, the number of tools that belong to the tool group can be read from tool life management data.
The number of tools that can be registered varies depending on the setting of parameter 6800 of the CNC, as indicated in the following table.

Parameter 6800

GS2	GS1	Number of tools The numbers in parentheses apply when the additional option is used	
		M series	T series
0	0	1 to 16 (1 to 32)	1 to 16 (1 to 32)
0	1	1 to 8 (1 to 16)	1 to 8 (1 to 16)
1	0	1 to 4 (1 to 8)	1 to 4 (1 to 8)
1	1	1 to 2 (1 to 4)	1 to 16 (1 to 4)

M series: For Machining Centers T series: For Lathes

[Input data structure]



> **⚠CAUTION**
> If 0 is specified for the tool group number, the number of tools that belong to the tool group currently used is read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.

[Completion codes]

0: The number of tools has been read normally.
3: The specified tool group number is invalid.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 40	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) –	Value
	+5	+80	Number of tools (4 bytes)	Unsigned binary

> **NOTE**
> 1 This cannot be used with those models that do not have the tool life management function.
> 2 Tool life management data is not applied to the T mode of the compound machining function.

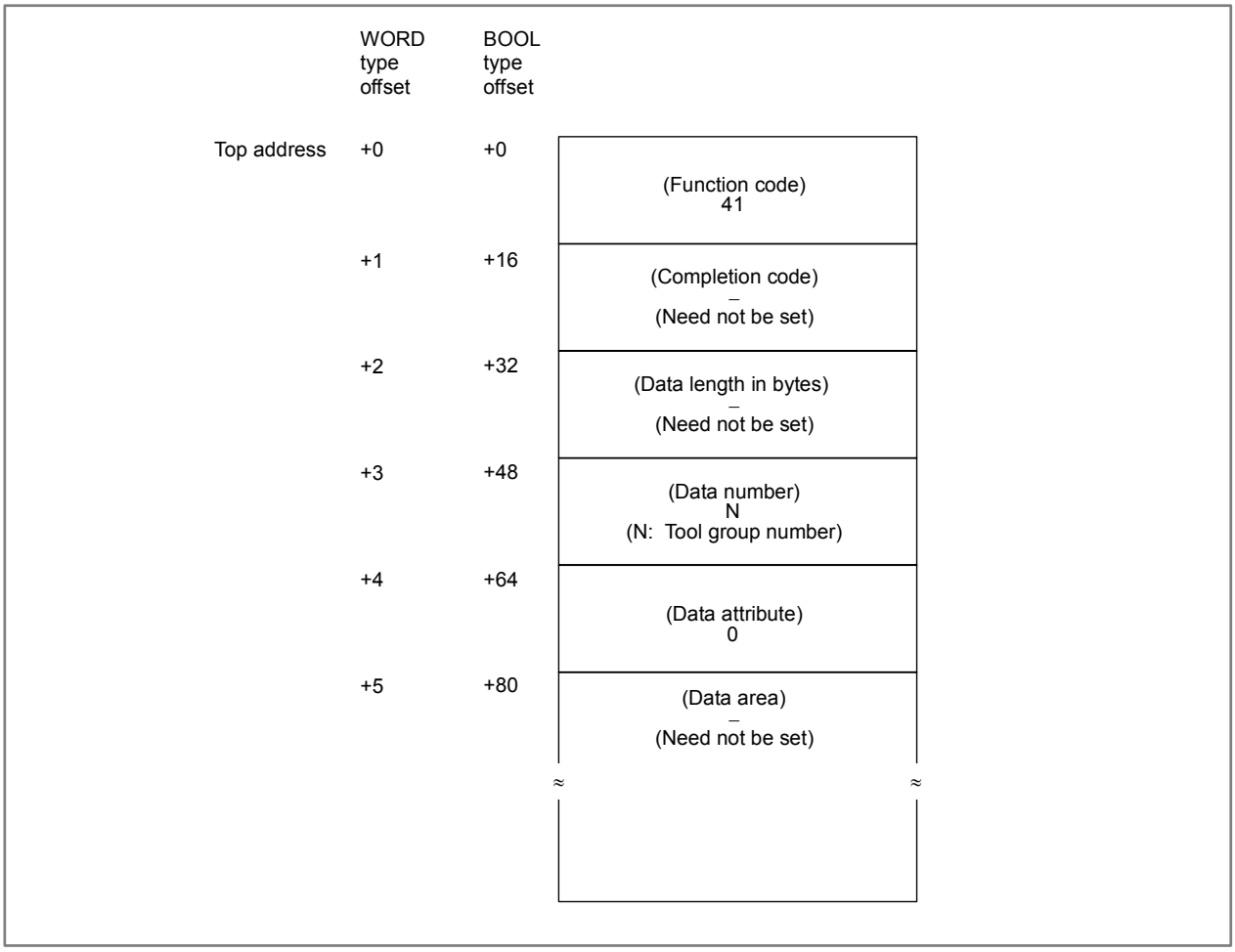
A.6.4 Reading Tool Life Management Data (Tool Life) (High-speed Response)

[Description]

By specifying a tool group number, the life of tools belonging to the tool group can be read from tool life management data.

Whether to display the tool life in minutes or the number of cycles is selected by bit 2 of parameter 6800 (LTM) for the CNC.

[Input data structure]



⚠CAUTION

If 0 is specified for the tool group number, the tool life of the tool group currently used is read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.

[Completion codes]

0: The tool life has been read normally.
3: The specified tool group number is invalid.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 41	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) –	Value
	+5	+80	Tool life (4 bytes)	Unsigned binary Unit: Time (minutes) or number of cycles

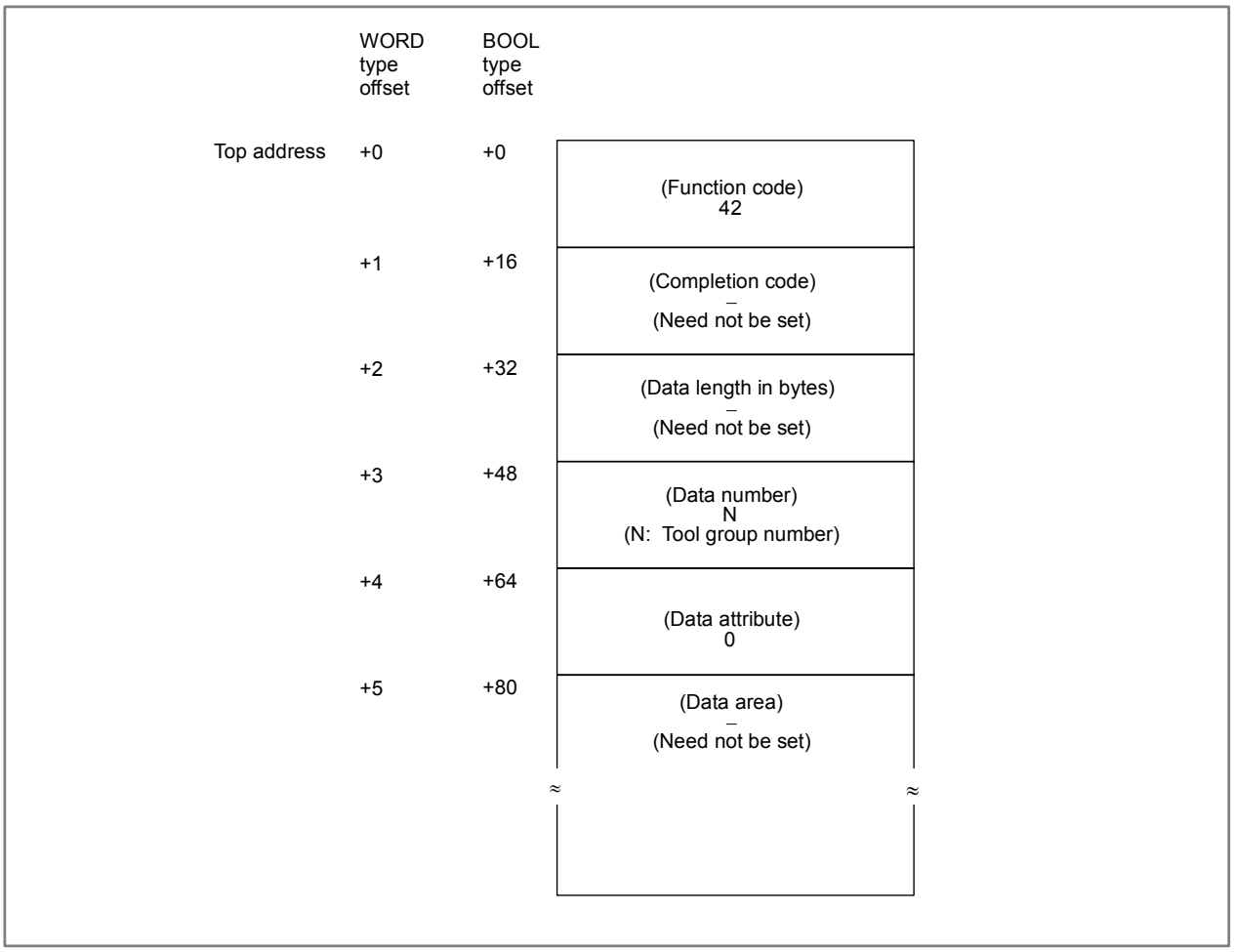
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.5 Reading Tool Life Management Data (Tool Life Counter) (High-speed Response)

[Description]

By specifying a tool group number, the tool life counter for the specified tool group can be read from tool life management data.

[Input data structure]



⚠CAUTION

If 0 is specified for the tool group number, the tool life counter for the tool group currently used is read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.

[Completion codes]

0: The tool life has been read normally.
3: The specified tool group number is invalid.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 42	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) –	Value
	+5	+80	Tool life counter (4 bytes)	Unsigned binary Unit: Time (minutes) or number of cycles

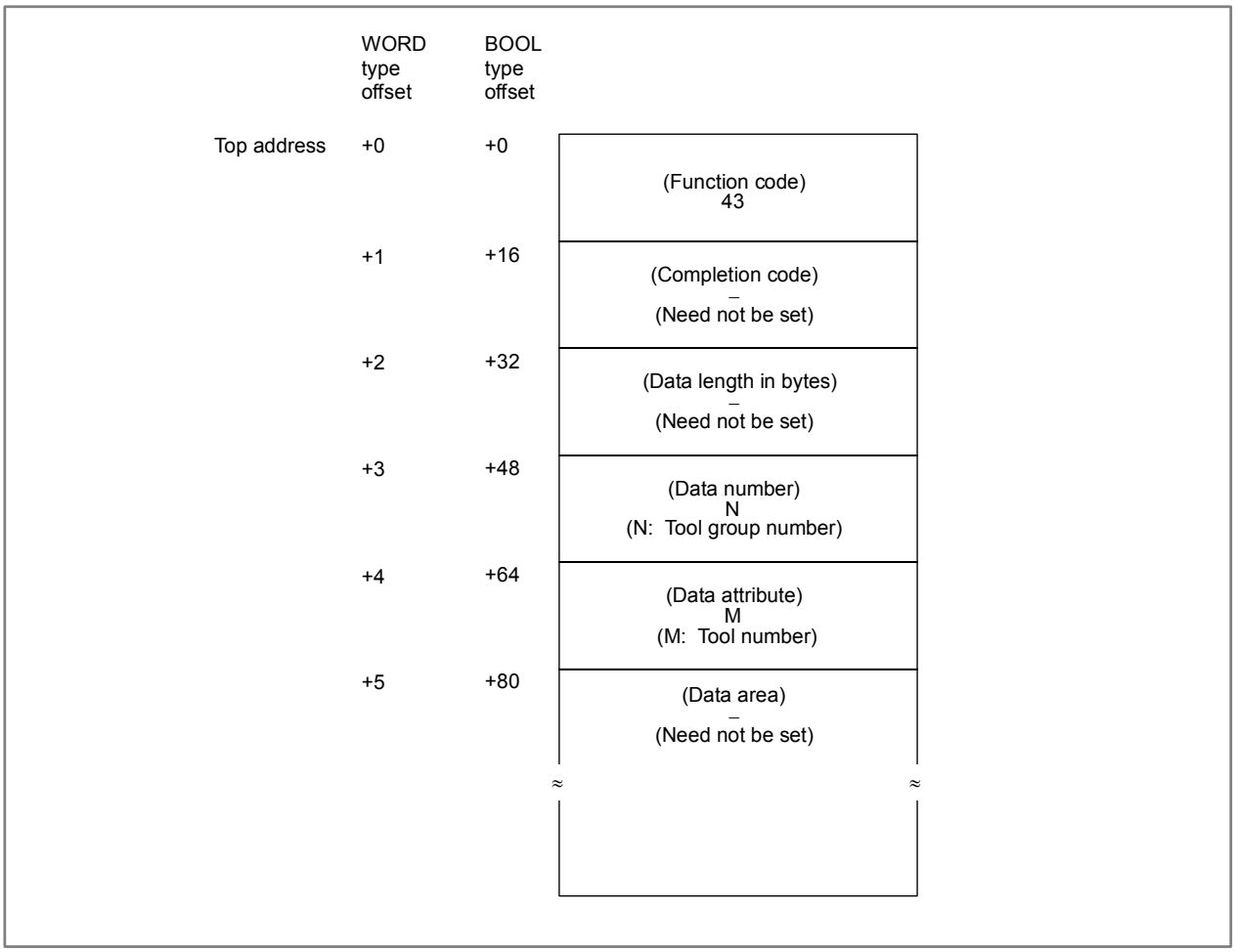
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.6 Reading Tool Life Management Data (Tool Length Compensation Number (1): Tool Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool No., the tool length compensation number for the specified tool can be read from tool life management data. This function is available only with the M series CNCs.

[Input data structure]



⚠CAUTION

If 0 is specified for both the tool group number and tool number, the numbers of the tool group and tool currently used are read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.
For the T series CNCs, 0 is always output.

[Completion codes]

0: The tool length compensation number has been read normally.
3: The specified tool group number is invalid.
4: The specified tool number is invalid.
5: The specified tool number was not found in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 43	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Tool length compensation number (4 bytes)	Unsigned binary

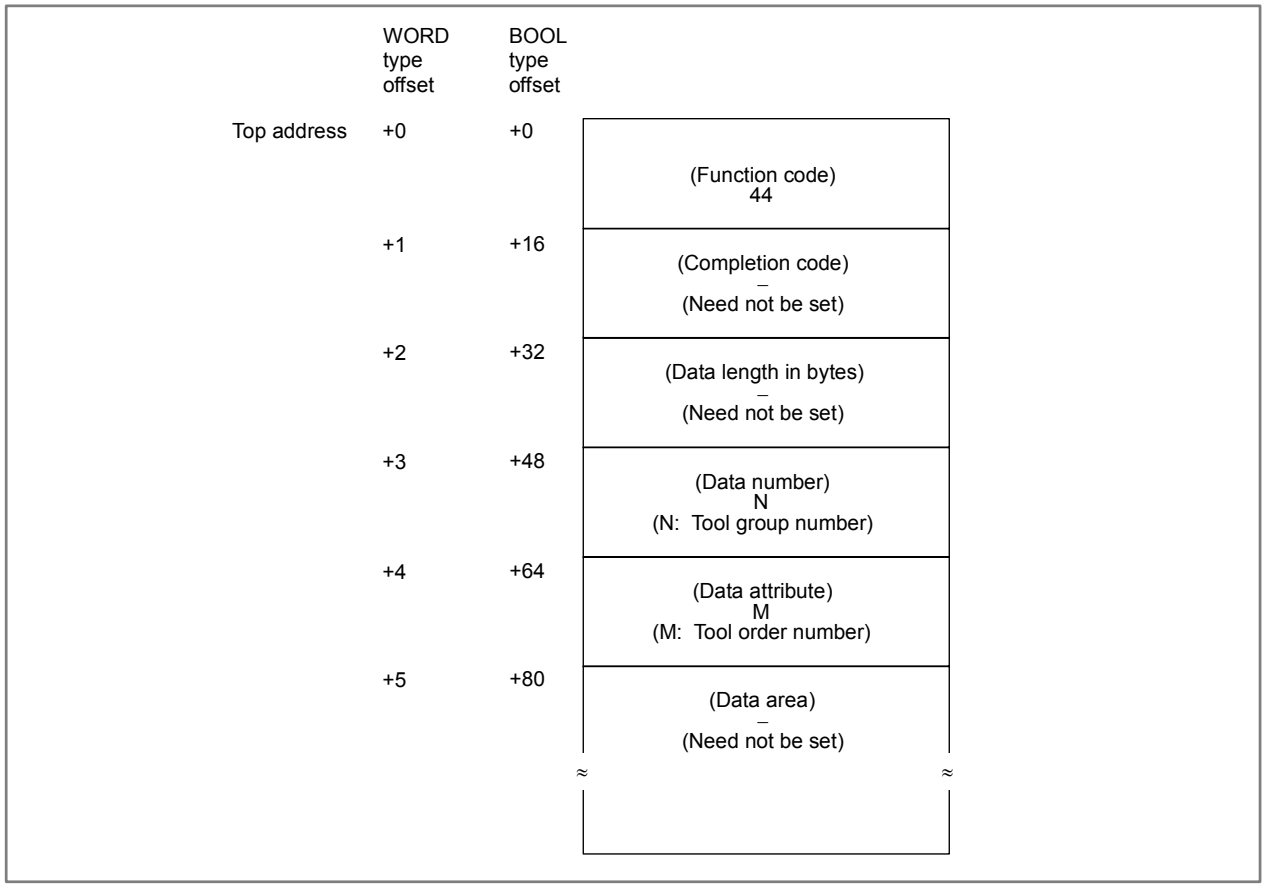
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.7 Reading Tool Life Management Data (Tool Length Compensation Number (2): Tool Order Number) (High-speed Response)

[Description]

By specifying a tool group number and tool order number, the tool length compensation number for the specified tool can be read from tool life management data. This function is available only with the M series CNCs.

[Input data structure]



⚠CAUTION

If 0 is specified for the tool group number, the number of the tool group currently used is read. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.
When 0 is specified for the tool order number, if the specified tool group has been used, the tool currently used is read. In this case, if the specified tool group has not been used, the first tool in the group is read.
For the T series CNCs, 0 is always output.

[Completion codes]

0: The tool length compensation number has been read normally.
3: The specified tool group number is invalid.
4: The specified tool order is invalid.
5: The tool having the specified tool order is not registered in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 44	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Tool length compensation number (4 bytes)	Unsigned binary

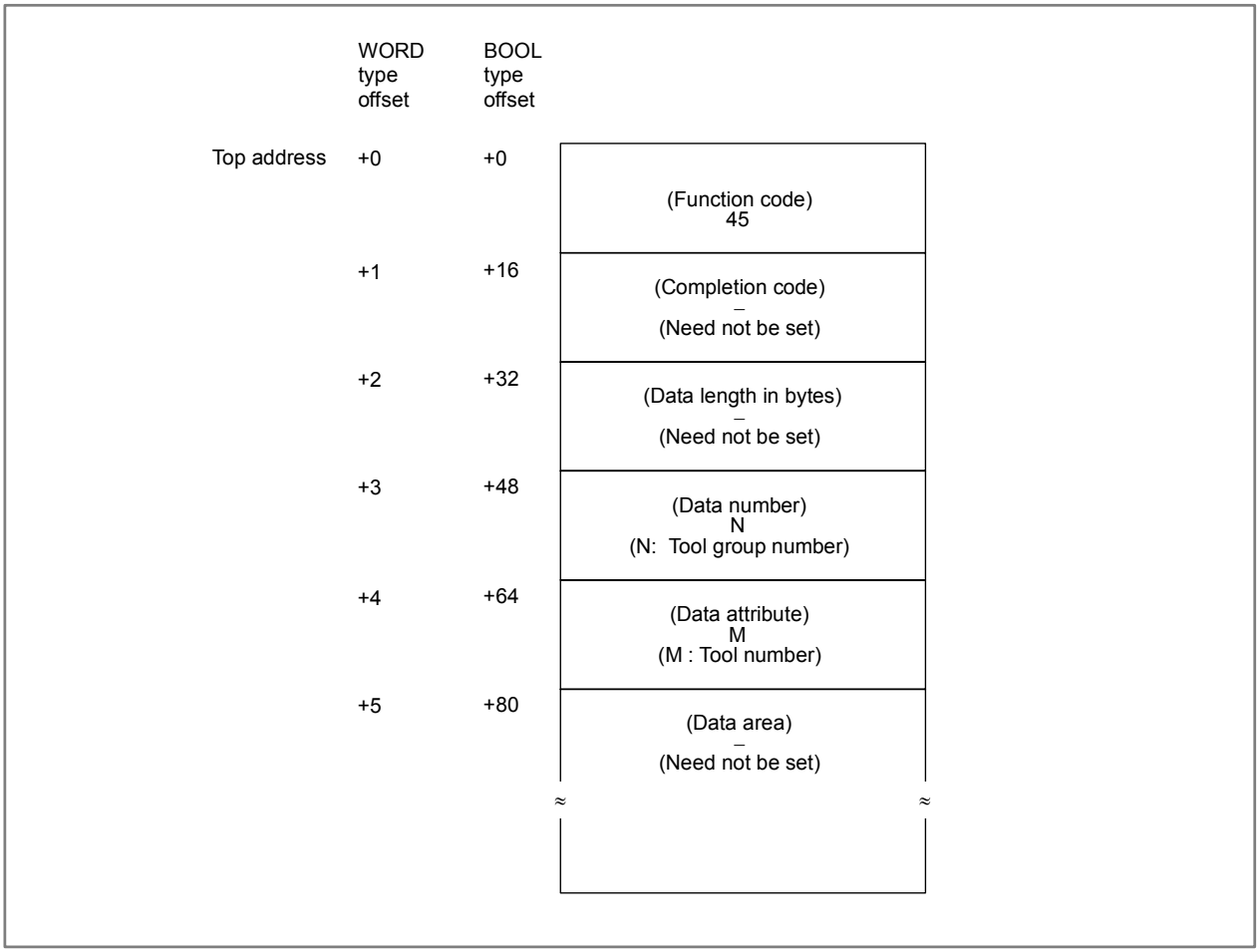
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.8 Reading Tool Life Management Data (Cutter Radius Compensation Number (1): Tool Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool number, the cutter compensation number for the specified tool can be read from tool life management data. This function is available only with the M series CNCs.

[Input data structure]



⚠CAUTION

If 0 is specified for both tool group number and tool number, the numbers of the tool group and tool currently used are read. If a tool group number has not been specified since the power to the CNC was turned on, 0 is output.
For the T series CNCs, 0 is always read.

[Completion codes]

0: The cutter compensation number has been read normally.
3: The specified tool group number is invalid.
4: The specified tool number is invalid.
5: The specified tool number was not found in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 45	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Cutter compensation number (4 bytes)	Unsigned binary

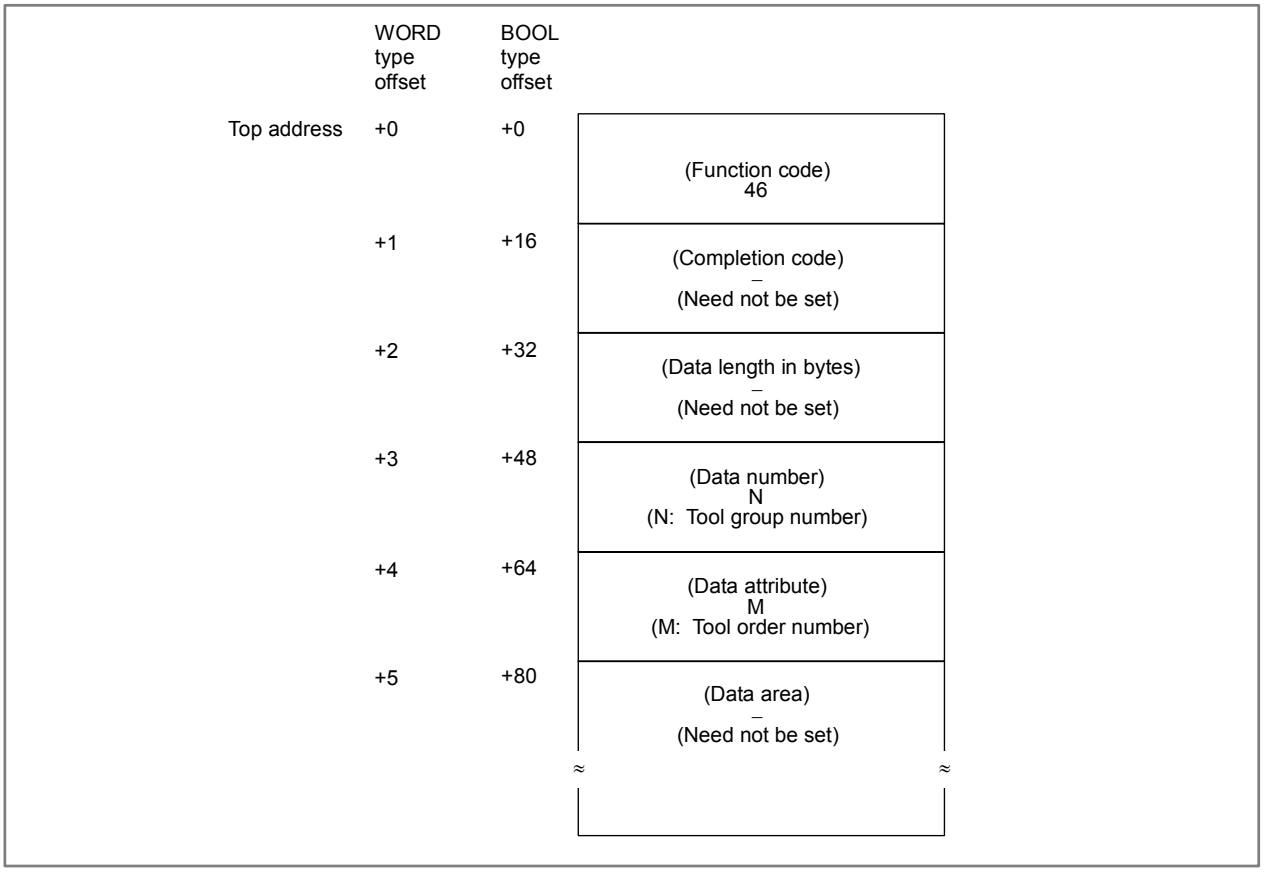
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.9 Reading Tool Life Management Data (Cutter Radius Compensation Number (2): Tool Order Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool order number, the cutter compensation number for the specified tool can be read from tool life management data. This function is available only with the M series CNCs.

[Input data structure]



⚠CAUTION

If 0 is specified for the tool group number, the number of the tool group currently used is referenced. In this case, if a tool group number has not been specified since the power to the CNC was turned on, 0 is output.

When 0 is specified for the tool order number, if the specified tool group has been used, the tool currently used is read. In this case, if the specified tool group has not been used, the first tool in the group is referred to.

For the T series CNCs, 0 is always output.

[Completion codes]

0: The cutter compensation number has been read normally.
3: The specified tool group number is invalid.
4: The specified tool order number is invalid.
5: The tool having the specified tool order is not registered in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 46	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Cutter compensation number (4 bytes)	Unsigned binary

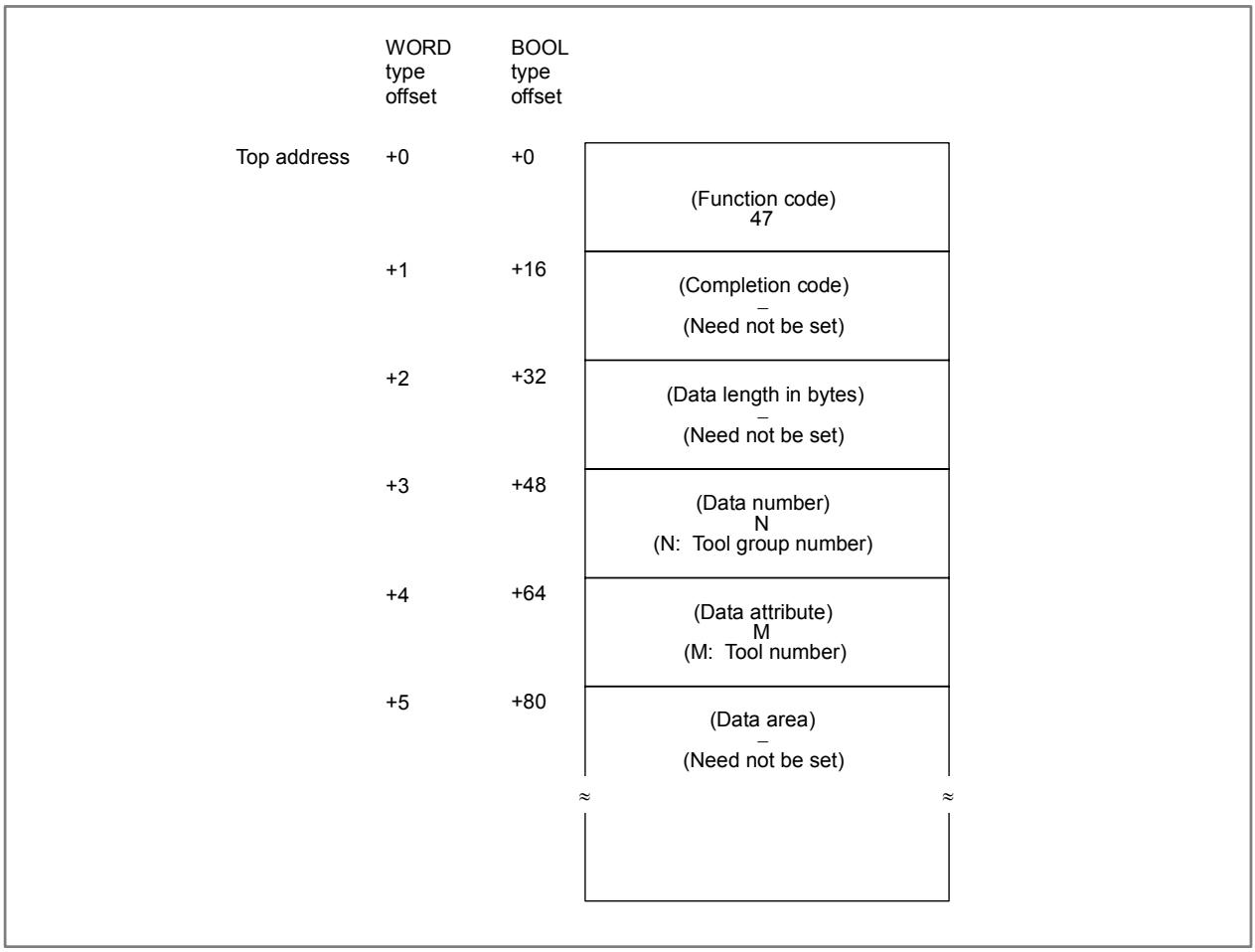
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.10 Reading Tool Life Management Data (Tool Information (1): Tool Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool number, the information for the specified tool can be read from tool life management data.

[Input data structure]



⚠CAUTION

If “0” is specified for both tool group number and tool number, the numbers of the tool group and tool currently used are referenced.

If neither a tool group number nor a tool number has been specified since the power to the CNC was turned on, “0” is output.

[Completion codes]

0: The tool status information has been read normally.
3: The specified tool group number is invalid.
4: The specified tool number is invalid.
5: The specified tool number was not found in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 47	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Tool status information (4 bytes)	0: See "CAUTION" on the previous page. 1: The tool is registered. 2: The tool has reached the end of its life. 3: The tool was skipped. The three high-order bytes are fixed to 0.

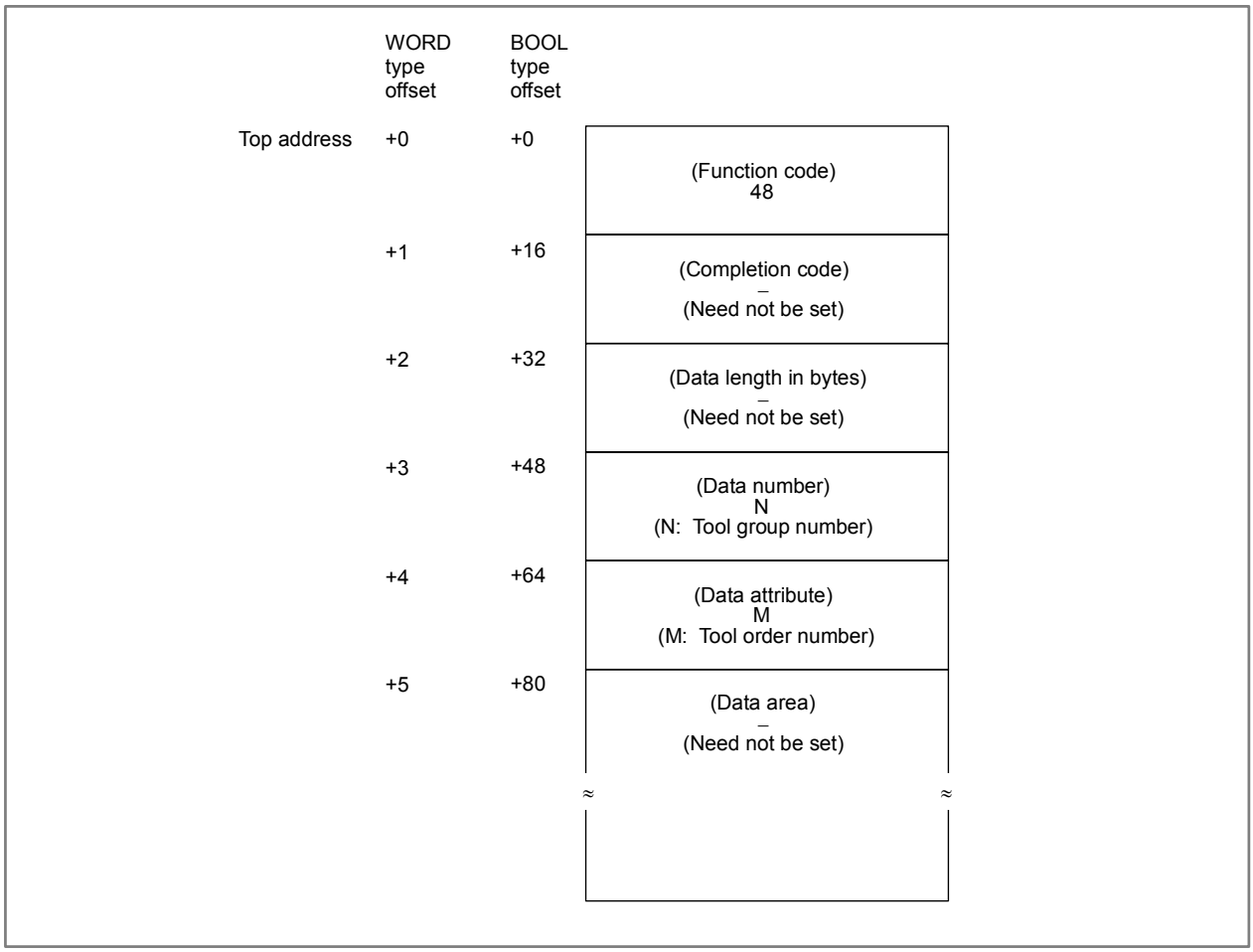
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.11 Reading Tool Life Management Data (Tool Information (2): Tool Order Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool order number, the information for the specified tool can be read from tool life management data.

[Input data structure]



⚠CAUTION

If "0" is specified for the tool group number, the number of the tool group currently used is read. If a tool group number has not been specified since the power to the CNC was turned on, "0" is output.
When "0" is specified for the tool order number, if the specified tool group has ever been used, the tool currently used is read. In this case, if the specified tool group has not been used, the first tool in the group is referred to.

[Completion codes]

0: The tool status information has been read normally.
3: The specified tool group number is invalid.
4: The specified tool order number is invalid.
5: The tool having the specified tool order is not registered in the specified tool group.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 48	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Tool status information (4 bytes)	0: See "Caution" on the previous page. 1: The tool is registered. 2: The tool has reached the end of its life. 3: The tool was skipped. The three high-order bytes are fixed to 0.

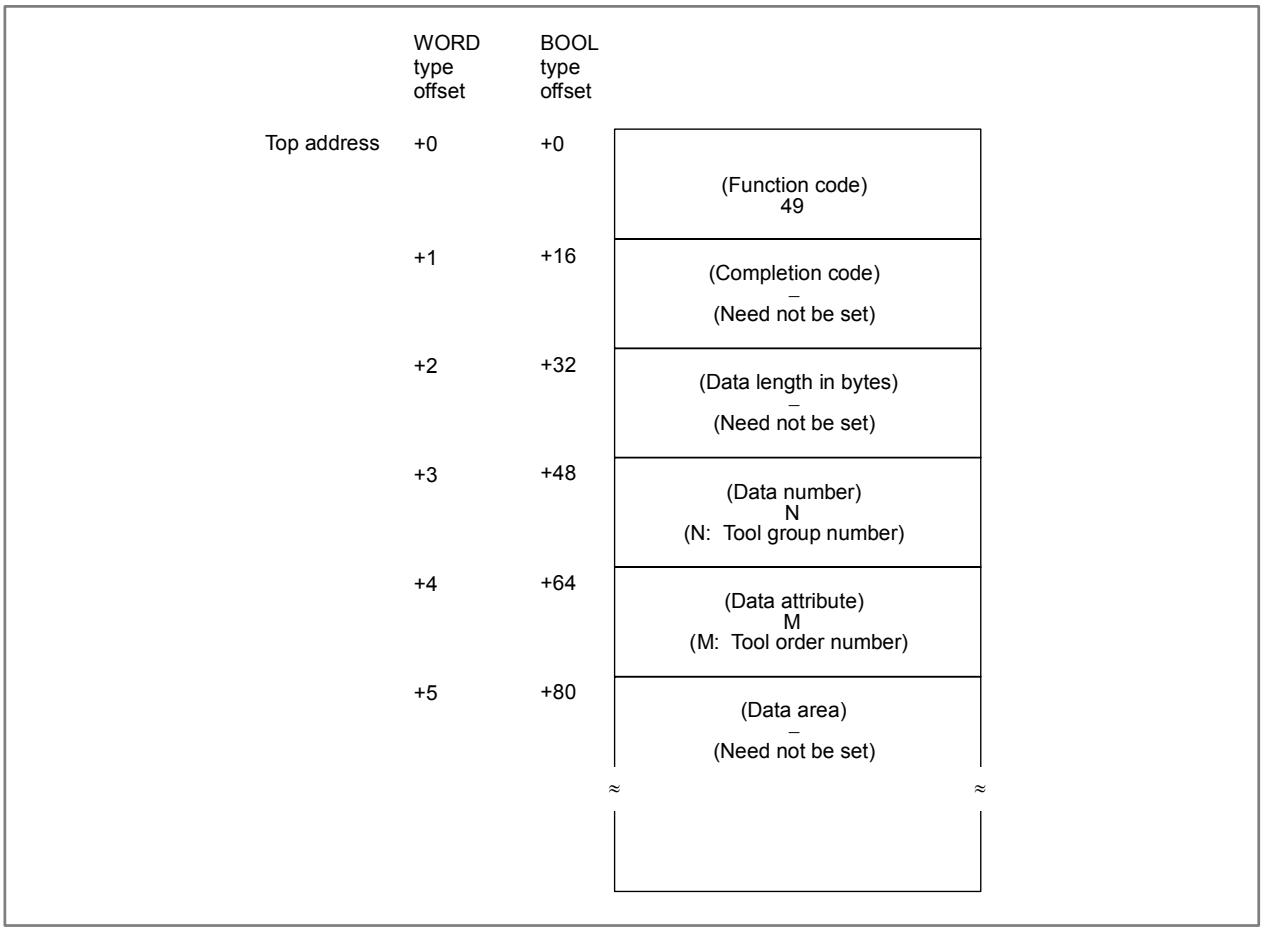
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.12 Reading Tool Life Management Data (Tool Number) (High-speed Response)

[Description]

By specifying a tool group number and a tool order number, the number of the corresponding tool can be read from tool life management data.

[Input data structure]



⚠CAUTION

When "0" is specified for the tool group number, the tool group currently used is referenced. If neither a tool group number or a tool number has been specified since the power to the CNC was turned on, however, "0" is output for the tool group number.
When "0" is specified for the tool order number, if the specified tool group has been used, the tool currently used is referred to. If the specified tool group has not been used, the first tool in the group is referenced.

[Completion codes]

0: The tool number has been read normally.
3: The specified tool group number is invalid.
4: The specified tool order number is invalid.
6: The tool life management option has not been added.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 49	
	+1	+16	(Completion code) ? (See the explanation of the completion codes.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Input data)	
	+4	+64	(Data attribute) M (M: Input data)	Value
	+5	+80	Tool number (4 bytes)	Unsigned binary

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.13 Reading the Tool Life Management Data (Tool Life Counter Type) (High-speed Response)

[Description]

This function gets the Tool life counter type of specified tool group in the Tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 160
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (N : Tool group number)
	+4	+64	(Data attribute) 0
	+5	+80	(Data area) – (Need not be set)

⚠CAUTION

About Tool group number (in 'Data number'):
"0" as Tool group number indicates the Tool group currently used.
When Tool group has never specified since power - on, "0" of Tool group number results "0" as counter type.
"0" of counter type will be returned on T series.

[Completion codes]

0: Success to read the Tool life counter type.
3: The Tool group number is out of range from 0 to 512, or exceeds the maximum number of registered Tool group.
6: No option for Tool life management.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 160	
	+1	+16	(Completion code) ? (See the explanation above)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) N (Same as input data)	
	+4	+64	(Data attribute) _ (Same as input data)	Value
	+5	+80	Tool life counter type (2 bytes)	0 : No counter type 1 : Frequency 2 : Real time (in minute)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.14 Registering Tool Life Management Data (Tool Group) (Low-speed Response)

[Description]

This function registers the Tool group in Tool life management data, with Tool number, length of life and Tool life counter type. On T series, the Tool life counter type will be specified by the NC parameter "LTM" (No.6800#2), and this function cannot set/change the counter type.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 163	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 8	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) M (M = Tool number)	Value
	+5	+80	Tool group number (2 bytes)	Unsigned binary 1–512
	+6	+96	Tool life counter type (2 bytes)	1: Frequency 2: Real time in minutes
	+7	+112	Tool life (4 bytes)	Unsigned binary 1–9999 (Frequency) 1–4300 (Real time in minutes: 1count/4s) Note 1–1090 (Real time in minutes: 1count/1s) Note

NOTE

The tool life count is changed by parameter 6801#5.
6801#5 = 0: the life is counted every 4seconds.
6801#5 = 1: the life is counted every 1second.

[Completion codes]

0: Success to register the tool group.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered Tool group.
4: The tool number in 'Data attribute' has wrong value.
5: The length of tool life in 'Data area' is out of range. The tool life counter type does not match on T series.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 163
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 8 (Same as input data)
	+3	+48	(Data number) _ (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool group number (2 bytes) (Same as input data)
	+6	+96	Tool life counter type (2 bytes) (Same as input data)
	+7	+112	Tool life (4 bytes) (Same as input data)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.15 Writing Tool Life Management Data (Tool Life) (Low-speed Response)

[Description]

This function sets the length of tool life of the specified tool group in the tool life management data.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 164	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) 0	Value
	+5	+80	Tool life (4 bytes)	Unsigned binary 1–9999 (Frequency) 1–4300 (Real time minutes: 1count/4s) Note 1–1090 (Real time minutes: 1count/1s) Note Case of the tool life management data B 1–999999 (Frequency) 1–100000 (Real time in minutes)

NOTE

The tool life count is changed by parameter 6801#5.
6801#5 = 0: the life is counted every 4seconds.
6801#5 = 1: the life is counted every 1second.

[Completion codes]

0: Success to set the length of tool life.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
5: The length of tool life is out of range.
6: No option for the tool life management.
13: The data of the currently selected tool group or the next tool group cannot be rewritten. An attempt was made to rewrite the data of the currently selected tool group or the next group.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 164
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) _ (Same as input data)
	+5	+80	Length of Tool life (4 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.16 Writing Tool Life Management Data (Tool Life Counter) (Low-speed Response)

[Description]

This function sets the tool life counter in the specified tool group in the tool life management data.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 165	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) 0	Value
	+5	+80	Tool life counter (4 bytes)	Unsigned binary 1–9999 (Frequency) 1–4300 (Real time minutes: 1count/4s) Note 1–1090 (Real time minutes: 1count/1s) Note Case of the tool life management data B 1–999999 (Frequency) 1–100000 (Real time in minutes)

NOTE

The tool life count is changed by parameter 6801#5.
6801#5 = 0: the life is counted every 4seconds.
6801#5 = 1: the life is counted every 1second.

[Completion codes]

0: Success to set the tool life counter.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
5: The value for tool life counter is out of range.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 165
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) – (Same as input data)
	+5	+80	Tool life counter(4 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.17 Writing Tool Life Management Data (Tool Life Counter Type) (Low-speed Response)

[Description]

This function sets the tool life counter type of specified tool group in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 166	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) 0	
	+5	+80	Tool life counter type (2 bytes)	Value 1: Frequency 2: Real time in minutes

[Completion codes]

0: Success to set the tool life counter type.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
5: The value for tool life counter type is wrong.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 166
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 2 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) – (Same as input data)
	+5	+80	Tool life counter type (2 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.18 Writing Tool Life Management Data (Tool Length Compensation Number (1): Tool Number) (Low-speed Response)

[Description]

This function sets the tool length compensation number of the specified tool group in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 167	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool number)	Value
	+5	+80	Tool length compensation number (4 bytes)	Unsigned binary 1-255

[Completion codes]

0: Success to set the tool length compensation number.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool number in 'Data attribute' has wrong value.
5: The tool number is not found in the tool group.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 167
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool length compensation number (4 bytes)

⚠CAUTION
The effective value for tool length compensation number depends on tool compensation number available on CNC.

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.19 Writing Tool Life Management Data (Tool Length Compensation Number (2): Tool Order Number) (Low-speed Response)

[Description]

This function sets the tool length compensation number of the tool of the specified tool order number in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 168	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool order number)	Value
	+5	+80	Tool length compensation number (4 bytes)	Unsigned binary 1-255

[Completion codes]

0: Success to set the tool length compensation number.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool order number is wrong.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 168
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool length compensation number (4 bytes)

⚠CAUTION
The effective value for tool length compensation number depends on tool compensation number available on CNC.

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.20 Writing Tool Life Management Data (Cutter Compensation Number (1): Tool Number) (Low-speed Response)

[Description]

This function sets the cutter compensation number of the specified tool group in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 169	
	+1	+16	(Completion code) _ (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool number)	Value
	+5	+80	Cutter compensation number (4 bytes)	Unsigned binary 1-255

[Completion codes]

0: Success to set the cutter compensation number.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool number in 'Data attribute' has wrong value.
5: The tool number is not found in the tool group.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 169
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Cutter compensation number (4 bytes)

⚠CAUTION
The effective value for Cutter compensation number depends on tool compensation number available on CNC.

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.21 Writing Tool Life Management Data (Cutter Compensation Number (2): Tool Order Number) (Low-speed Response)

[Description]

This function sets the cutter compensation number of the tool of the specified tool order number in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 170	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool order number)	Value
	+5	+80	Cutter compensation number (4 bytes)	Unsigned binary 1-255

[Completion codes]

0: Success to set the cutter compensation number.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool order number is wrong.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 170
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Cutter compensation number (4 bytes)

⚠CAUTION
The effective value for cutter compensation number depends on tool compensation number available on CNC.

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.22 Writing the Tool Life Management Data (Tool Information (1): Tool Number) (Low-speed Response)

[Description]

This function sets the Tool information of the specified Tool group in the Tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 171	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool number)	Value
	+5	+80	Tool information (2 bytes)	1: Tool state clear 2: Tool state skip
	+6	+96		

[Completion codes]

0: Success to set the tool information.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool number in 'Data attribute' has wrong value.
5: The tool number is not found in the tool group.
6: No option for tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 171
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 2 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool information (2 bytes)
	+6	+96	

This function changes tool condition as shown below.

Command	Before call	After call
clear	Skip (#) Skip (#) Consumed (*)	Unused ( ) In use (@) Unused ( )
skip	Unused ( ) In use (@) Consumed (*)	Skip (#) Skip (#) Skip (*)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.23 Writing the Tool Management Data (Tool Information (2): Tool Order Number) (Low-speed Response)

[Description]

This function sets the tool information of the tool of the specified tool order number in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 172	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 2	
	+3	+48	(Data number) N (N : Tool group number)	
	+4	+64	(Data attribute) M (M : Tool order number)	Value
	+5	+80	Tool information (2 bytes)	1: Tool status clear 2: Tool status skip

[Completion codes]

0: Success to set the tool information.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool order number is wrong.
6: No option for Tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 172
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 2 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool information (2 bytes)

This function changes tool condition as shown below.

Command	Before call	After call
clear	Skip (#) Skip (#) Consumed (*)	Unused ( ) In use (@) Unused ( )
skip	Unused ( ) In use (@) Consumed (*)	Skip (#) Skip (#) Skip (*)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.24 Writing Tool Life Management Data (Tool Number) (Low-speed Response)

[Description]

This function registers a tool to the specified tool group in the tool life management data.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 173	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool order number)	Value
	+5	+80	Tool number (4 bytes)	Unsigned binary 1–9999

[Completion codes]

0: Success to register the tool number.
3: The tool group number is out of range from 1 to 512, or exceeds the maximum number of registered tool group.
4: The tool order number is wrong.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 173
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+5	+80	Tool number (4 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.25 Reading The Tool Life Management Data (Tool Group Number) (8-digit tool number) (High-speed Response)

[Description]

This function reads the tool group number to which the tool number is currently registered. This function supports 8 digits tool number.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 200
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) 0
	+4	+64	(Data attribute) M (M = Tool number)
	+6	+96	(Data area) – (Need not be set)

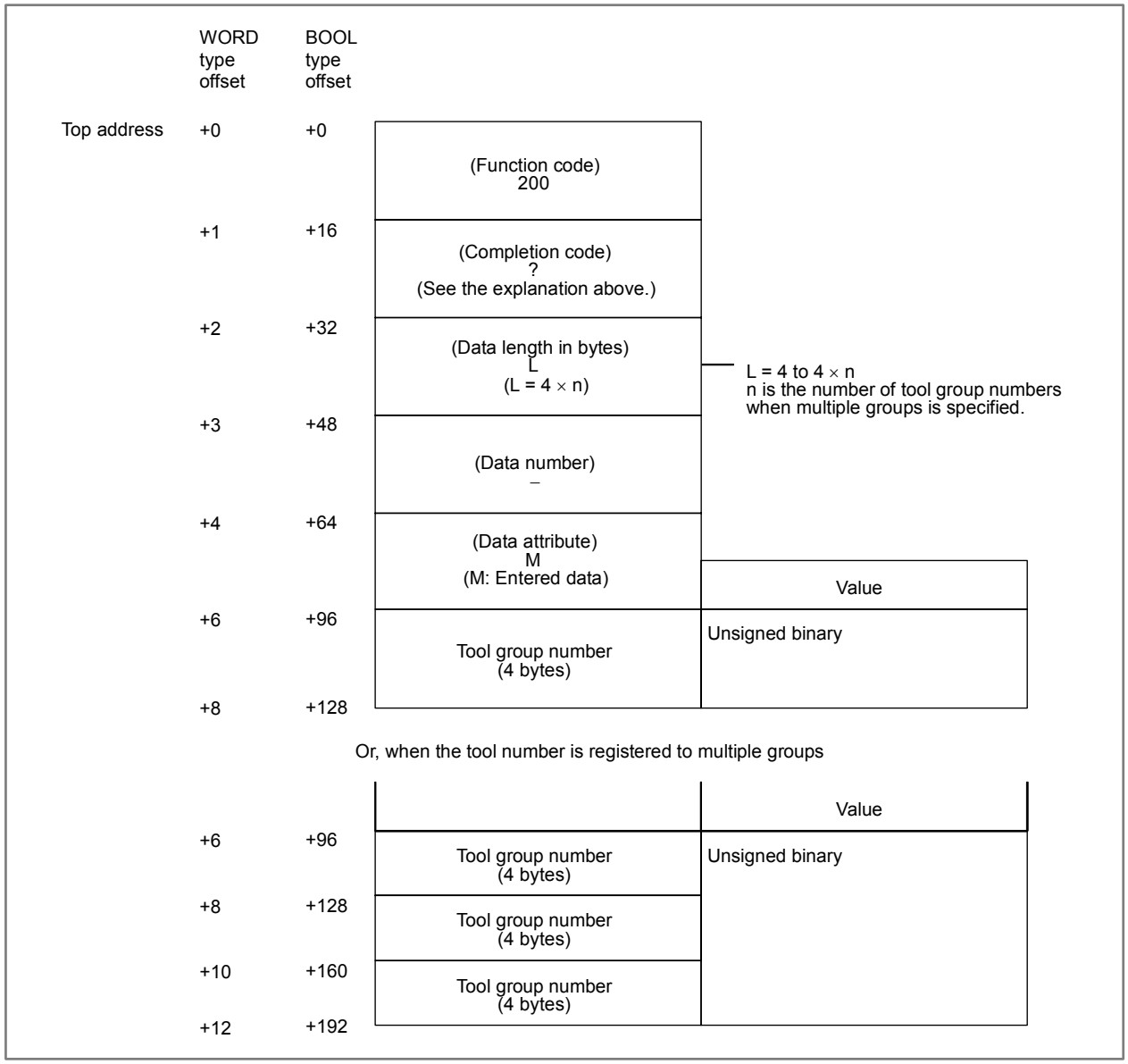
NOTE

When the tool number is set to "0", the tool group number of the currently used tool is read.
If a tool group number is not specified after the power is turned ON, tool group number "0" is read.
Also, if a tool number is registered to two or more tool group numbers, the tool group numbers of all tool groups to which the tool number is registered are read.

[Completion codes]

0: The tool group number was read successfully.
4: The tool number in 'Data Attribute' has a wrong value.
5: The tool number is not registered.
6: The tool life management option has not been added on.

[Output data structure]



NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.26 Reading Tool Life Management Data (Tool Information (1): Tool Number) (8-digit tool number) (High-speed Response)

[Description]

This function reads the tool information (status) according to the specified tool group number and tool number.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 201
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (N : Tool group number)
	+4	+64	(Data attribute) M (M : Tool number)
	+6	+96	(Data area) – (Need not be set)
	+8	+128	

⚠CAUTION

When the tool group number and tool number are set to "0", the currently used tool group and tool number are referenced.

If a tool group number is not specified after the power is turned ON, tool group number "0" is read.

[Completion codes]

0: The tool information was read successfully.
3: The specified tool group number is incorrect.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 201	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Entered data)	
	+4	+64	(Data attribute) M (M: Entered data)	Value
	+6	+96	Tool status information (4 bytes)	0: See "CAUTION" on the previous page. 1: Tool is registered. 2: End of tool life. 3: Tool skipped. All above three must be “0”.
	+8	+128		

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.27 Registering Tool Life Management Data (Tool Group Number) (8-digit tool number) (Low-speed Response)

[Description]

This function registers the tool group number to tool life management data. Set the tool number, life value and life counter type to the specified tool group. On the T series, since the life counter type is specified by CNC parameter LTM (No. 6800#2), it cannot be set nor changed here.

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 202	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 8	
	+3	+48	(Data number) 0	
	+4	+64	(Data attribute) M (M = Tool number)	Value
	+6	+96	Tool group number (2 bytes)	Unsigned binary 1–512
	+7	+112	Tool life counter type (2 bytes)	1: Frequency 2: Real time in minutes
	+8	+128	Tool life (4 bytes)	Unsigned binary 1–9999 (Frequency) 1–4300 (Real time minutes: 1count/4s) Note 1–1090 (Real time minutes: 1count/1s) Note Case of the tool life management data B 1–999999 (Frequency) 1–100000 (Real time in minutes)
	+10	+160		

NOTE

The tool life count is changed by parameter 6801#5.
6801#5 = 0: the life is counted every 4seconds.
6801#5 = 1: the life is counted every 1second.

[Completion codes]

0: The tool group was registered successfully.
3: The tool group number exceeded the range 1 to 512 or maximum number of registered groups.
4: The tool number in 'Data Attribute' has a wrong value.
5: The tool life value is out-of-range. On the T series, the tool life counter type is different.
6: The tool life management option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 202
	+1	+16	(Completion code) ? (See the explanation above.)
	+2	+32	(Data length in bytes) 8 (Same as input data)
	+3	+48	(Data number) – (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+6	+96	Tool group number (2 bytes) (Same as input data)
	+7	+112	Tool life counter type (2 bytes) (Same as input data)
	+8	+128	Tool life value (4 bytes) (Same as input data)
	+10	+160	

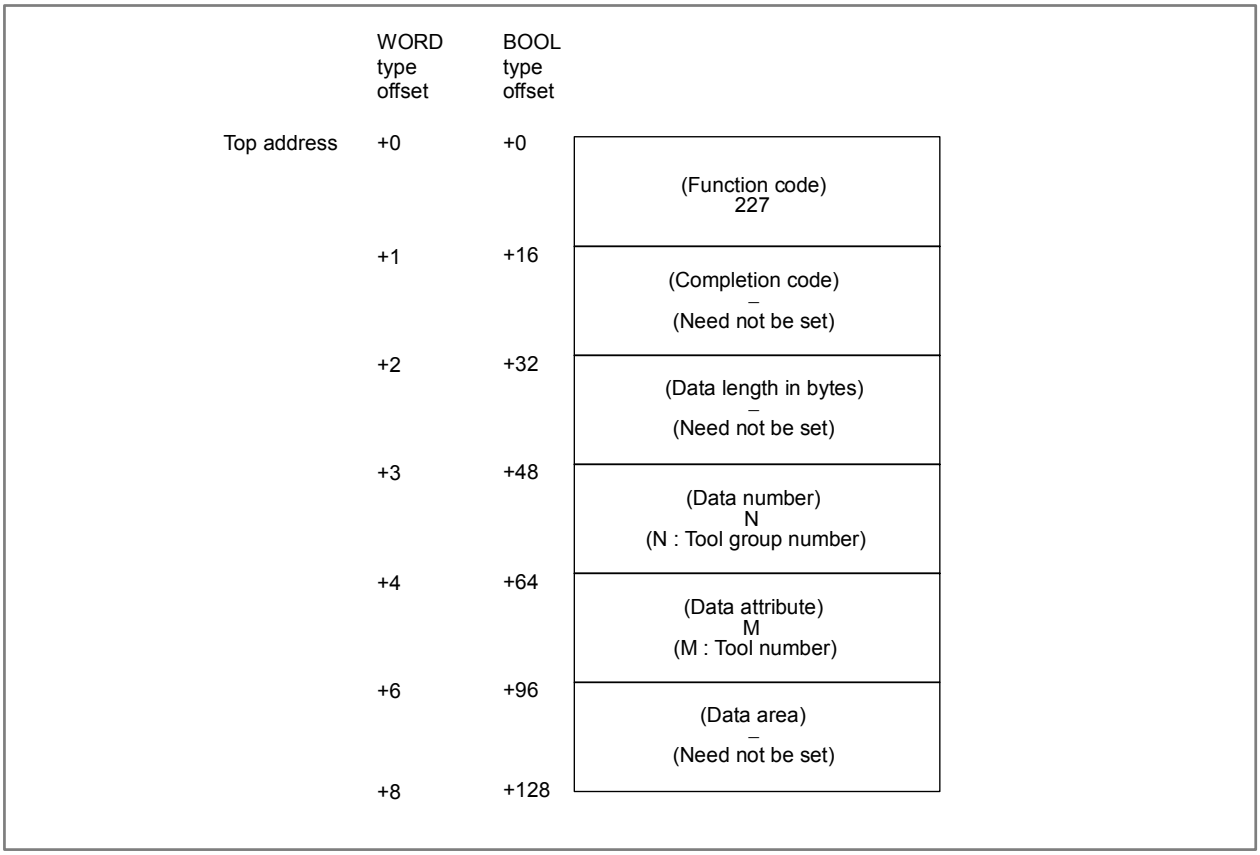
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.28 Reading Tool Life Management Data (Tool Length Compensation Number (1): Tool Number) (8-digit tool number) (High-speed Response)

[Description]

This function reads the tool length compensation number according to the specified tool group number and tool number. (M series only)

[Input data structure]



⚠CAUTION

When the tool group number and tool number are set to "0", the currently used tool group and tool number are referenced.

If a tool group number is not specified after the power is turned ON, tool group number "0" is read. "0" is always read on the T series.

[Completion codes]

0: The tool length compensation number was read successfully.
3: The specified tool group number is incorrect.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 227	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Entered data)	
	+4	+64	(Data attribute) M (M: Entered data)	Value
	+6	+96	Tool length compensation number (4 bytes)	Unsigned binary
	+8	+128		

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.29 Reading Tool Life Management Data (Cutter Compensation Number (1): Tool Number) (8-digit tool number) (High-speed Response)

[Description]

This function reads the cutter compensation number according to the specified tool group number and tool number. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 228
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) N (N : Tool group number)
	+4	+64	(Data attribute) M (M : Tool number)
	+6	+96	(Data area) – (Need not be set)
	+8	+128	

⚠CAUTION

When the tool group number and tool number are set to "0", the currently used tool group and tool number are referenced.

If a tool group number is not specified after the power is turned ON, tool group number "0" is read. "0" is always read on the T series.

[Completion codes]

0: The cutter compensation number was read successfully.
3: The specified tool group number is incorrect.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 228	
	+1	+16	(Completion code) ? (See the explanation above.)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N: Entered data)	
	+4	+64	(Data attribute) M (M: Entered data)	Value
	+6	+96	Cutter compensation number (4 bytes)	Unsigned binary
	+8	+128		

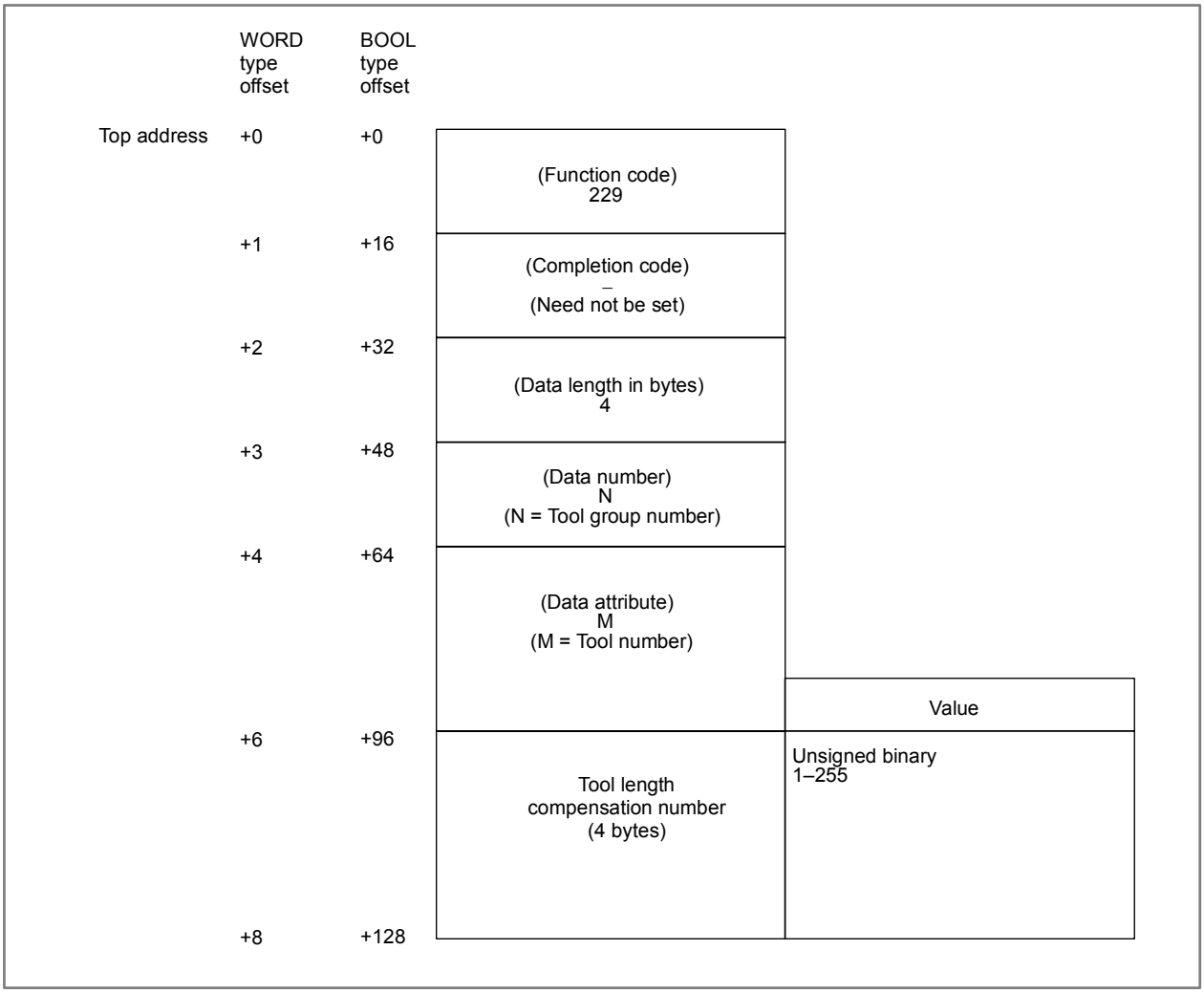
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.30 Writing Tool Life Management Data (Tool Length Compensation Number (1): Tool Number) (8-digit tool number) (Low-speed Response)

[Description]

This function sets the tool length compensation number of a specified tool group in the tool life management data. (M series only)

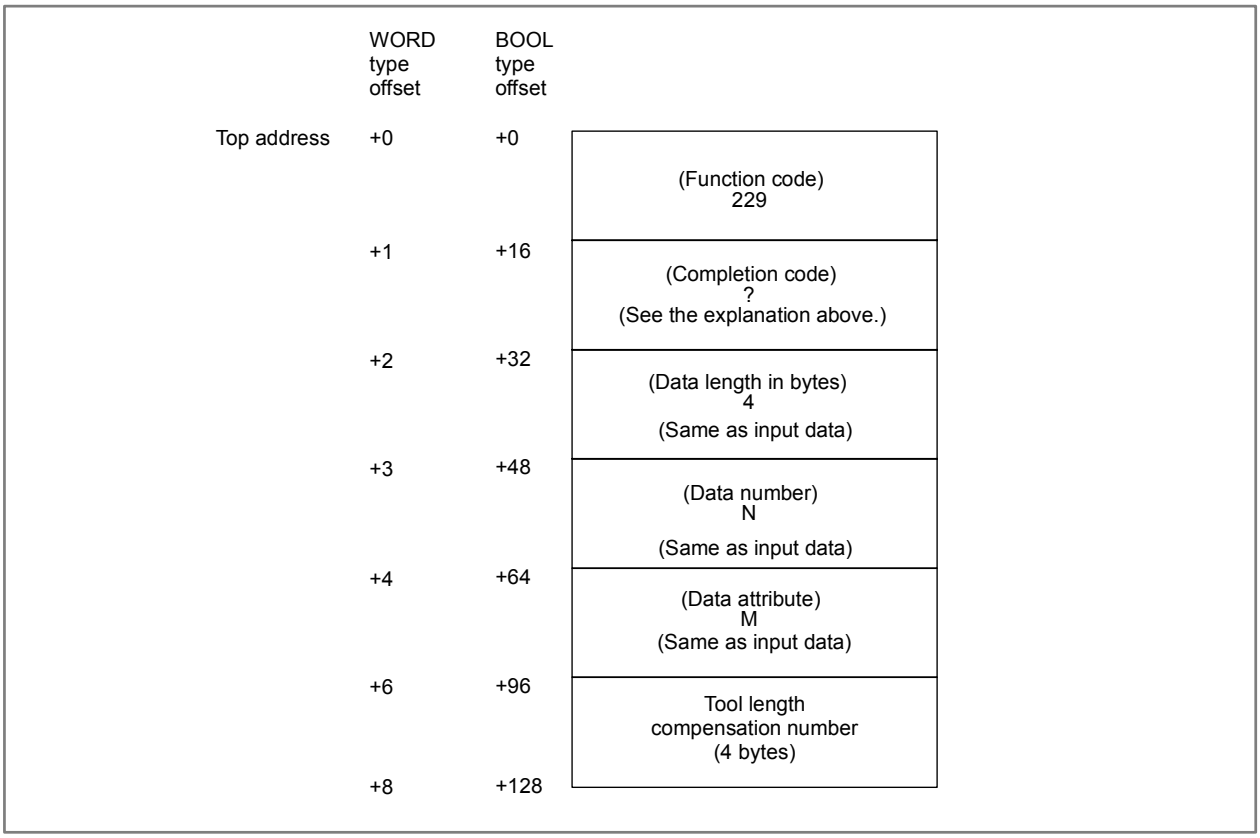
[Input data structure]



[Completion codes]

0: The tool length compensation number was written successfully.
3: The specified tool group number is incorrect.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]



⚠CAUTION
The effective value for tool length compensation number depends on tool compensation number available on CNC.

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.31 Writing Tool Life Management Data (Cutter Compensation Number (1): Tool Number) (8-digit tool number) (Low-speed Response)

[Description]

This function sets the cutter compensation number of a tool belonging to a specified tool group in the tool life management data. (M series only)

[Input data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 230	
	+1	+16	(Completion code) – (Need not be set)	
	+2	+32	(Data length in bytes) 4	
	+3	+48	(Data number) N (N = Tool group number)	
	+4	+64	(Data attribute) M (M = Tool number)	
				Value
	+6	+96	Cutter compensation number (4 bytes)	Unsigned binary 1–255
	+8	+128		

[Completion codes]

0: The cutter compensation number was written successfully.
3: The tool group number exceeded the range 1 to 512 or maximum number of registered groups.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 230
	+1	+16	(Completion code) ? (See the explanation above.)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) M (Same as input data)
	+6	+96	Cutter compensation number (4 bytes)
	+8	+128	

⚠CAUTION
The effective value for Cutter compensation number depends on tool compensation number available on CNC.

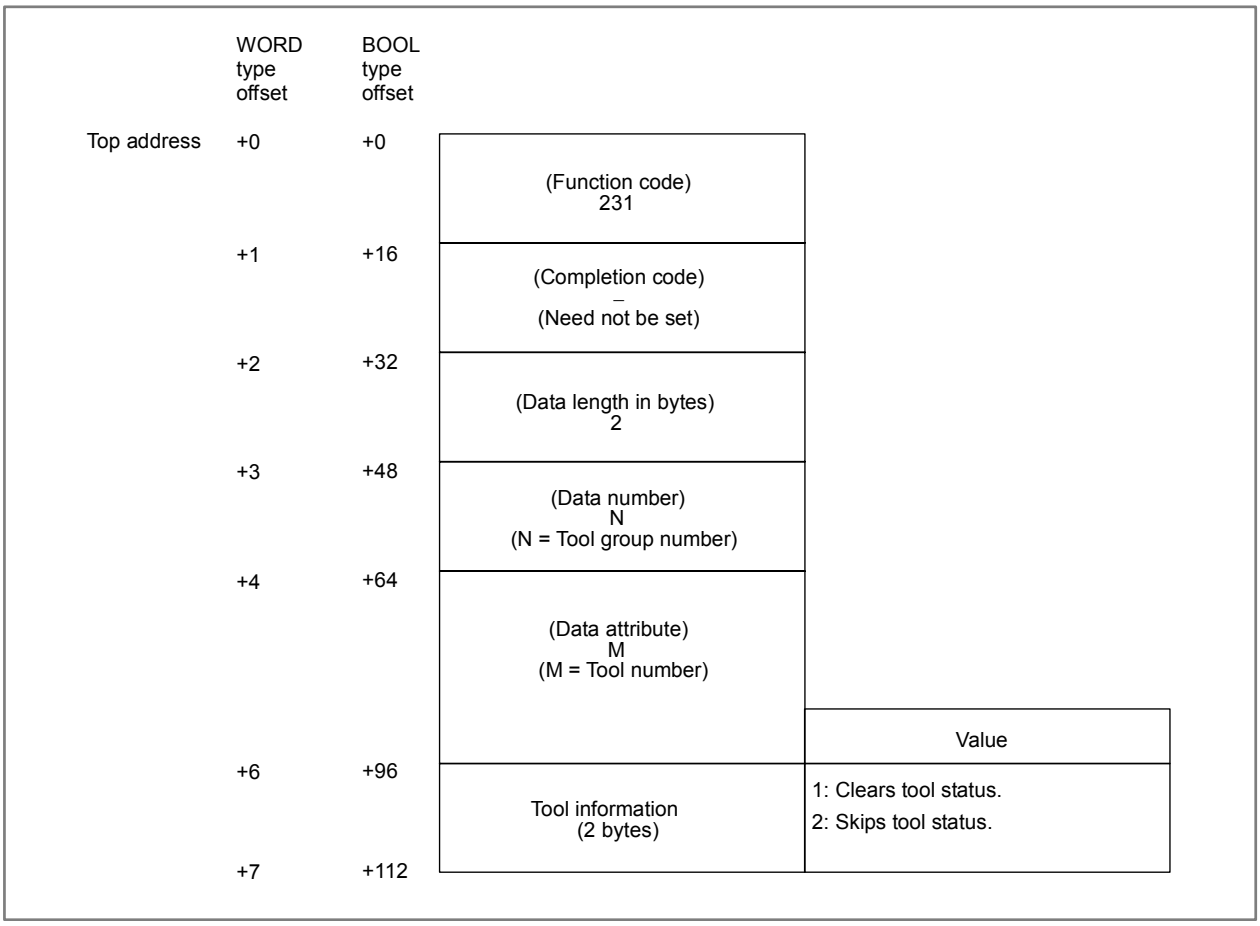
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.32 Writing the Tool Life Management Data (Tool Information (1): Tool Number) (8-digit tool number) (Low-speed Response)

[Description]

This function sets the tool information of a tool belonging to a specified tool group in the tool life management data. (M series only)

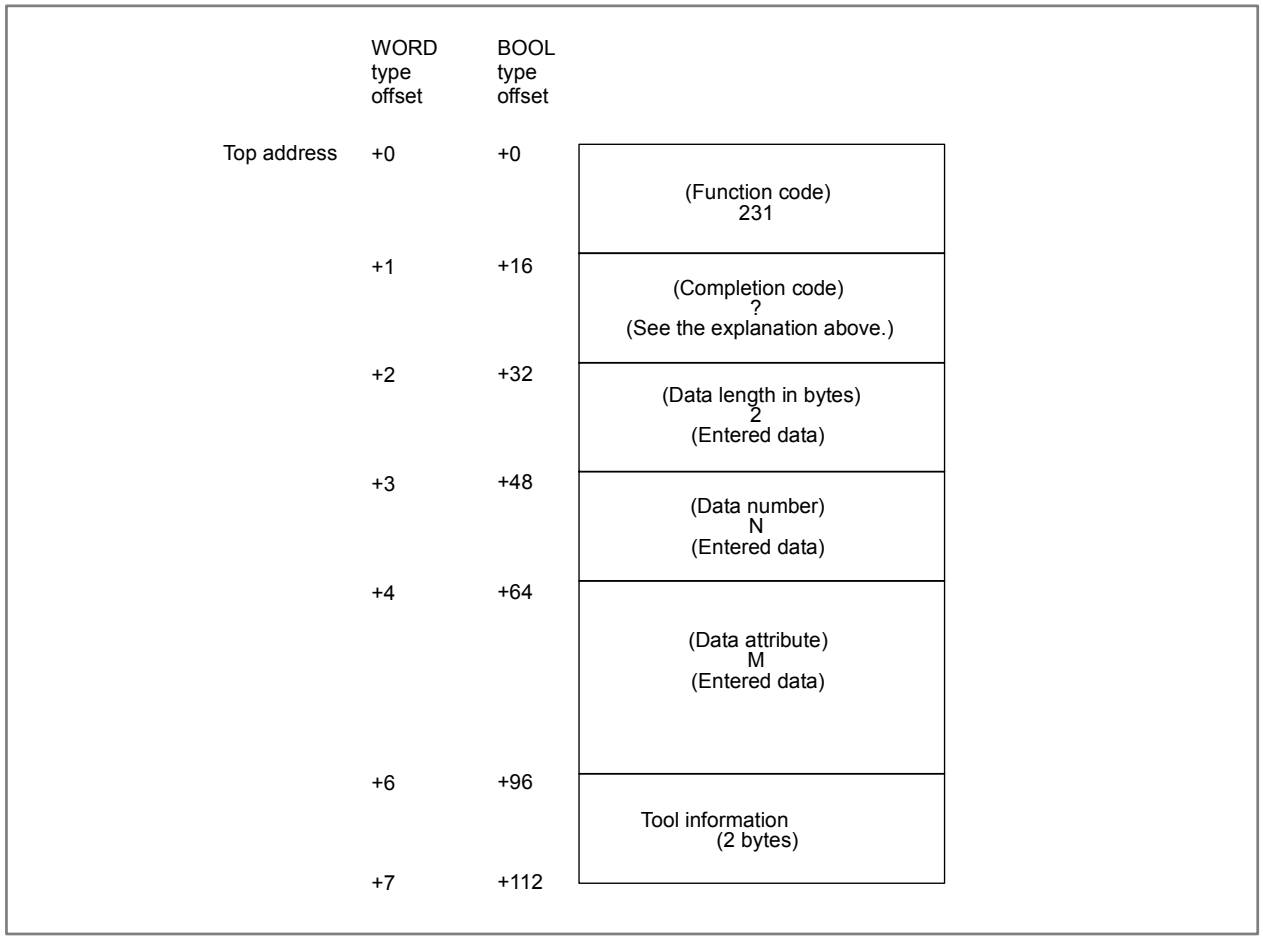
[Input data structure]



[Completion codes]

0: The tool information was written successfully.
3: The tool group number exceeded the range 1 to 512 or maximum number of registered groups.
4: The specified tool number is incorrect.
5: The specified tool number is not registered to the specified tool group.
6: The tool life management option has not been added on.

[Output data structure]



The following table shows how the tool status changes before and after this function is specified.

Command	Before call	After call
clear	Skip (#) Skip (#) Used (*)	Unused ( ) In use (@) Unused ( )
skip	Unused ( ) In use (@) Used (*)	Skip (#) Skip (#) Skip (*)

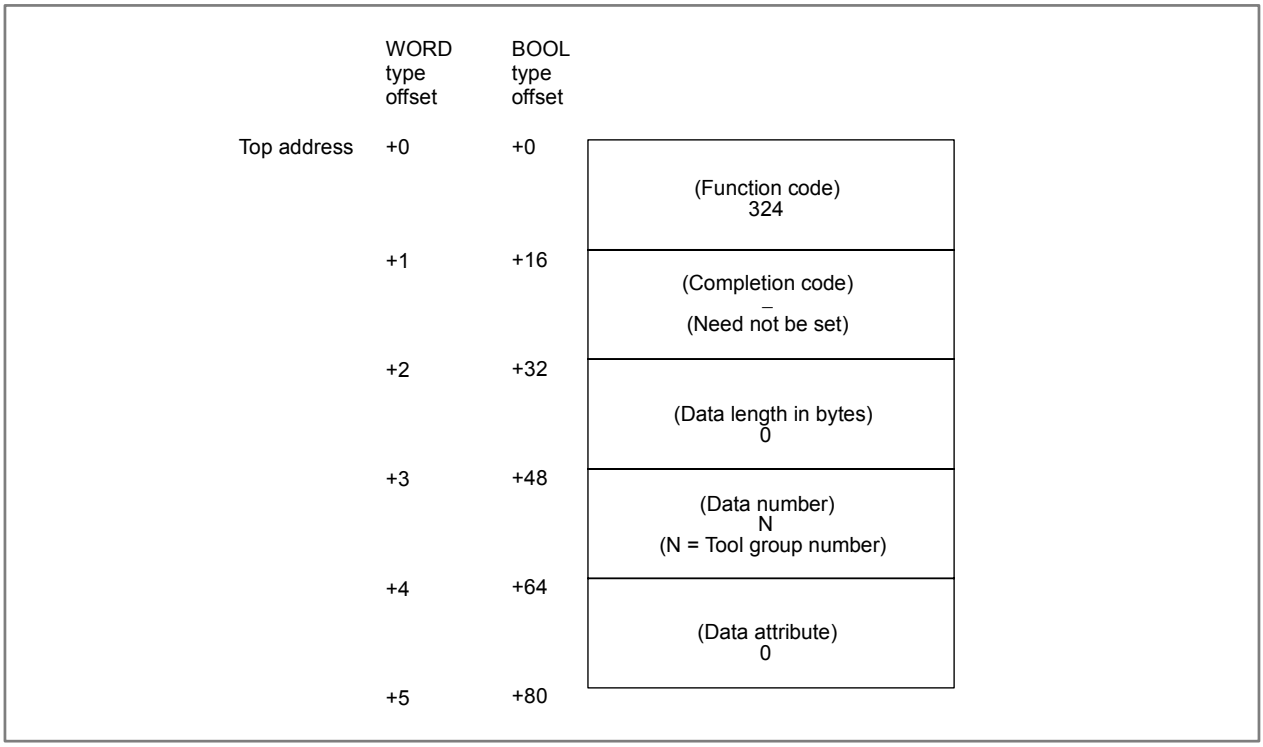
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.33 Deleting Tool life Management Data (Tool Group) (Low-speed Response)

[Description]

This function deletes the specified tool group in the tool life management data. In short, it makes the condition that Tool group is not registered.

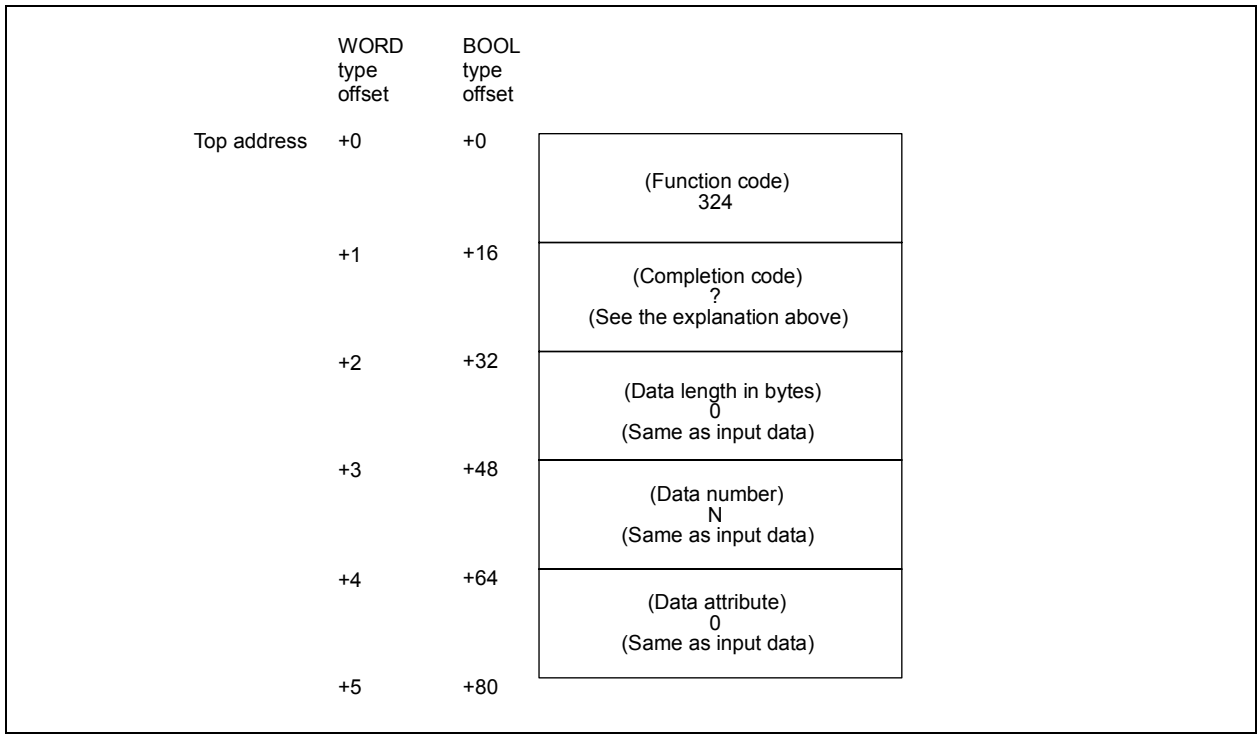
[Input data structure]



[Completion codes]

0: Success to delete the tool group number.
3: The tool group number is out of range from 1 to 512,or exceeds the maximum number of registered tool group.
6: No option for the tool life management.

[Output data structure]



NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.34 Deleting Tool life Management Data (Tool Data) (Low-speed Response)

[Description]

The function deletes the tool data of the tool of the specified tool order number in the tool life management data. (M series only)

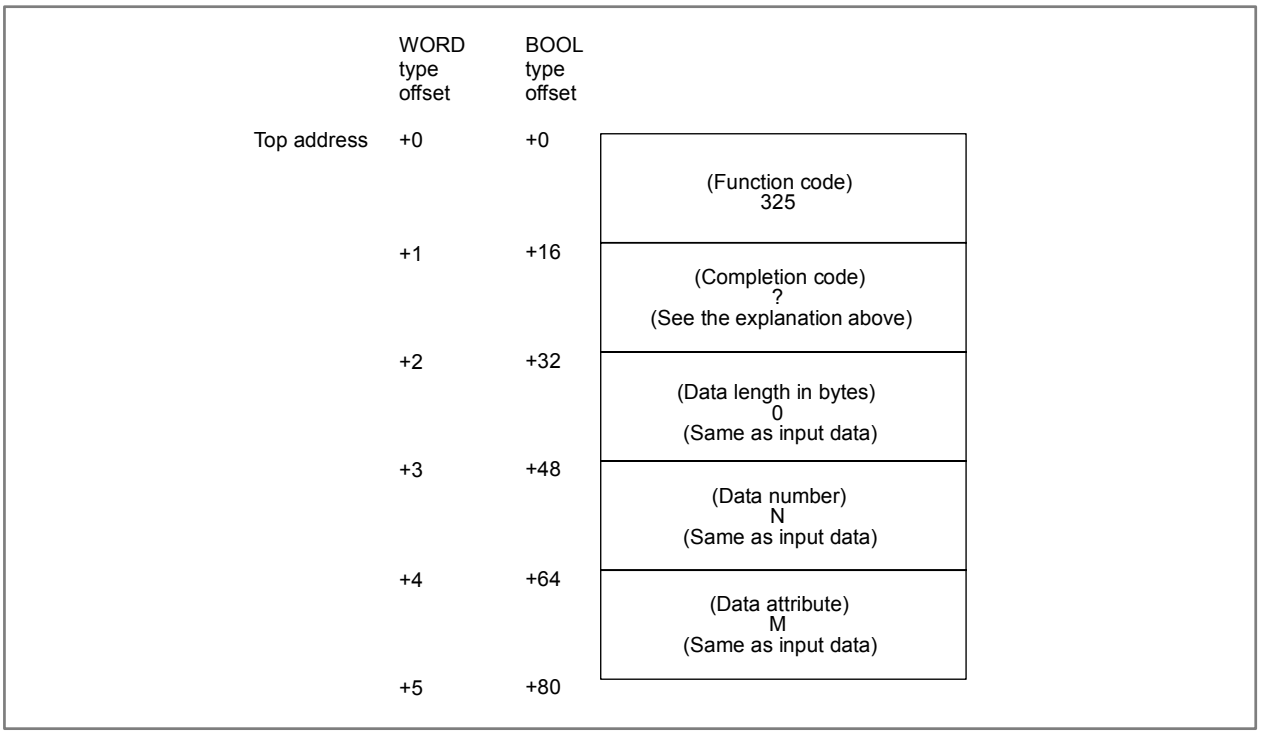
[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 325
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 0
	+3	+48	(Data number) N (N = Tool group number)
	+4	+64	(Data attribute) M (M = Tool order number)
	+5	+80	

[Completion codes]

0: Success to delete the tool group number.
3: The tool group number is out of range from 1 to 512,or exceeds the maximum number of registered tool group.
4: The tool order number is wrong.
6: No option for the tool life management.

[Output data structure]



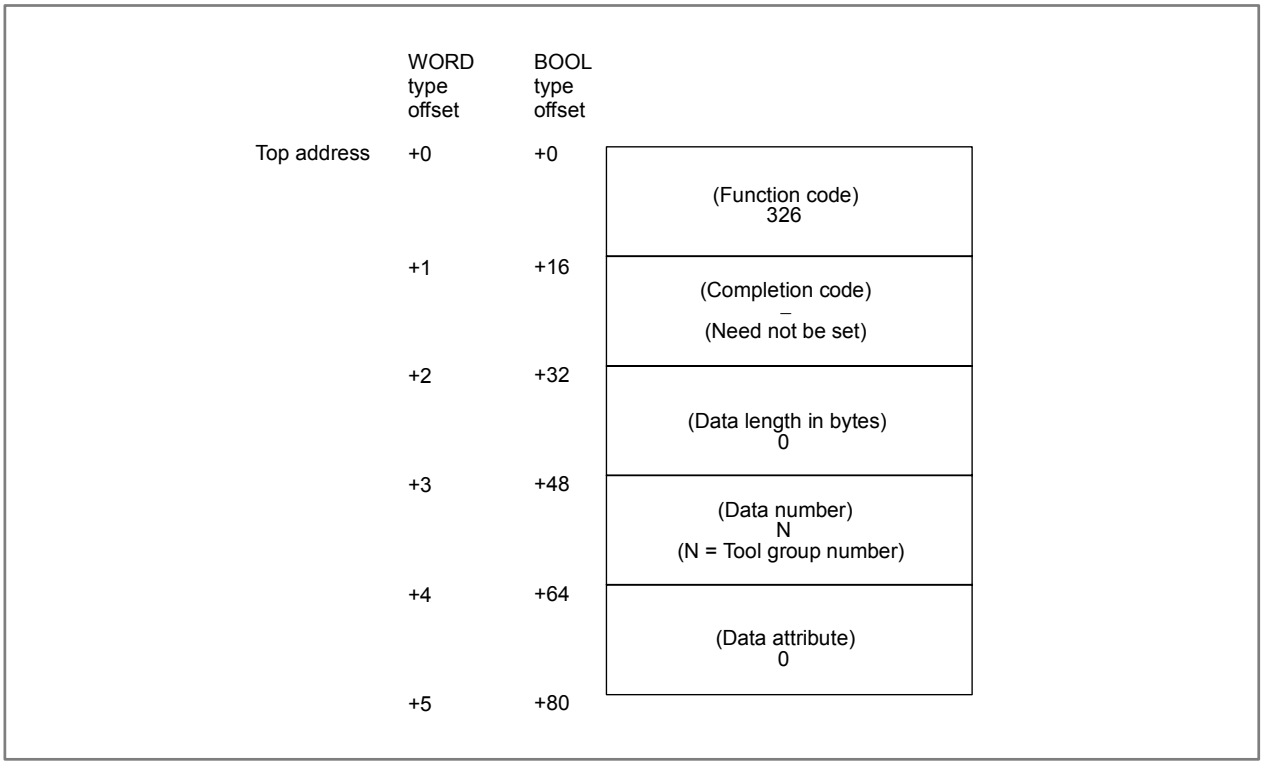
NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.35 Clearing Tool Life Management Data (Tool Life Counter and Tool Information) (Low-speed Response)

[Description]

This function clears the tool life counter and all tool information of the specified tool group in the tool life management data.

[Input data structure]



[Completion codes]

0: Success to clear the tool life counter and the tool information.
3: The tool group number is out of range from 1 to 512,or exceeds the maximum number of registered tool group.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 326
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 0 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) 0 (Same as input data)
	+5	+80	

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.

A.6.36 Writing Tool Life Management Data (Arbitrary Group Number) (Low-speed Response)

[Description]

This function sets arbitrary group number of the specified tool group in the tool life management data.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 327
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 4
	+3	+48	(Data number) N (N = Tool group number)
	+4	+64	(Data attribute) 0
	+5	+80	Arbitrary group number (4 bytes)

NOTE

Writing the tool life Management Data (Arbitrary group number) is available for tool life management data B.

[Completion codes]

0: Success to set the arbitrary group number.
3: The tool group number is out of range from 1 to 512,or exceeds the maximum number of registered tool group.
5: Arbitrary group number is out of range.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 327
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) 0 (Same as input data)
	+5	+80	Arbitrary group number (4 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.6.37 Writing Tool Life Management Data (Remaining Tool Life) (Low-speed Response)

[Description]

This function sets the length of remaining tool life of the specified tool group in the tool life management data.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 328
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 4
	+3	+48	(Data number) N (N = Tool group number)
	+4	+64	(Data attribute) 0
	+5	+80	Remaining tool life (4 bytes)

NOTE

Writing the tool life Management Data (Remaining tool life) is available for tool life management data B.

[Completion codes]

0: Success to set the length of remaining tool life.
3: The tool group number is out of range from 1 to 512,or exceeds the maximum number of registered tool group.
5: Remaining tool life is out of range.
6: No option for the tool life management.

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 328
	+1	+16	(Completion code) ? (See the explanation above)
	+2	+32	(Data length in bytes) 4 (Same as input data)
	+3	+48	(Data number) N (Same as input data)
	+4	+64	(Data attribute) 0 (Same as input data)
	+5	+80	Remaining tool life (4 bytes)

NOTE
1 This cannot be used with those models that do not have the tool life management function.
2 Tool life management data is not applied to the T mode of the compound machining function.
And compound machining function is not applied to the function of tool life management data B.

A.7 TOOL MANAGEMENT FUNCTIONS

Command of special magazine at multipath system

The tool management data and the magazine management table are common to all paths. But the spindle position table and wait position table are individual at each path. When the spindle position or the wait position at each path will be specified as the magazine through PMC window, the magazine number is depend on the following table.

	Spindle position			
	1st	**2nd**	**3rd**	**4th**
1st Path	111 (11)	112 (12)	113 (13)	114 (14)
2nd Path	211	212	213	214
3rd Path	311	312	313	314

	Wait position			
	1st	**2nd**	**3rd**	**4th**
1st Path	121 (21)	122 (22)	123 (23)	124 (24)
2nd Path	221	222	223	224
3rd Path	321	322	323	324

NOTE
The CNC can control a maximum of four axes per path.

A.7.1 Exchange of Tool Management Data Numbers in a Magazine Management Table (Low-speed Response)

[Description]

This function exchanges tool management data number in a pair of pots of specified magazine.

When the magazine number is set to 11 - 14(1 - 4th spindle position) or 21 - 24(1 - 4th waiting position), the pot number may be dummy number. It is necessary for specifying the 2nd or 3rd spindle/wait position at the multipath system to specify the path number by the place of 100. For example, the magazine number 213 means the 3rd spindle position at the 2nd path. Especially, the 1st spindle position can be specified without the third of hundreds (The magazine number 22 and 122 indicates the same magazine).

When tool management function oversize tools support option is effective, if changed tools are interfered with other tools, completion code 5 and detail completion code 27 are returned. Refer to note of the following description for details.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 329
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 8
	+3	+48	(Data number) – (Need not be set)
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data number 2) – (Need not be set)
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	Magazine number 1 (2 bytes)
	+8	+128	Pot number 1 (2 bytes)
	+9	+144	Magazine number 2 (2 bytes)
	+10	+160	Pot number 2 (2 bytes)

[Completion codes]

0: Normal end.
5: Specified magazine number / pot number is not registered / Tool interference
6: No required option.

[Detailed completion codes]

This code is 0 except completion code is 5.
Completion code = 5

21: Error of magazine No.1
22: Error of pot No.1
24: Error of magazine No.2
25: Error of pot No.2
27: Interfere with other tools

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 329
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 8
	+3	+48	(Data number) –
	+4	+64	(Data attribute) –
	+5	+80	(Data number 2) –
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	Magazine number 1 (2 bytes)
	+8	+128	Pot number 1 (2 bytes)
	+9	+144	Magazine number 2 (2 bytes)
	+10	+160	Pot number 2 (2 bytes)

A.7.2 Search of empty pot (Low-speed Response)

[Description]

This function searches the nearest empty pot, which is expressed, as tool management data is 0, from specified pot in the magazine. Spindle and waiting position are not searched as empty pot.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 330
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Search direction
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	Data area (4 bytes) (Need not be set)

[Search direction]

–1: Descending order
0: No order (The nearest pot is searched)
1: Ascending order

[Completion codes]

0: Normal end
3: Error of magazine number/pot number
4: Error of search direction
6: No required option
8: No empty pot

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3
21: Error of magazine number
22: Error of pot number

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 330
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 4
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Search direction
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	Searched magazine number (2 bytes)
	+8	+128	Searched pot number (2 bytes)

When this function is executed without specifying search direction, the pot found in ascending direction is output if empty pots are found at the same distance away from the specified pot in both orders.

A.7.3 New-register of a Tool Management Data (Low-speed Response)

[Description]

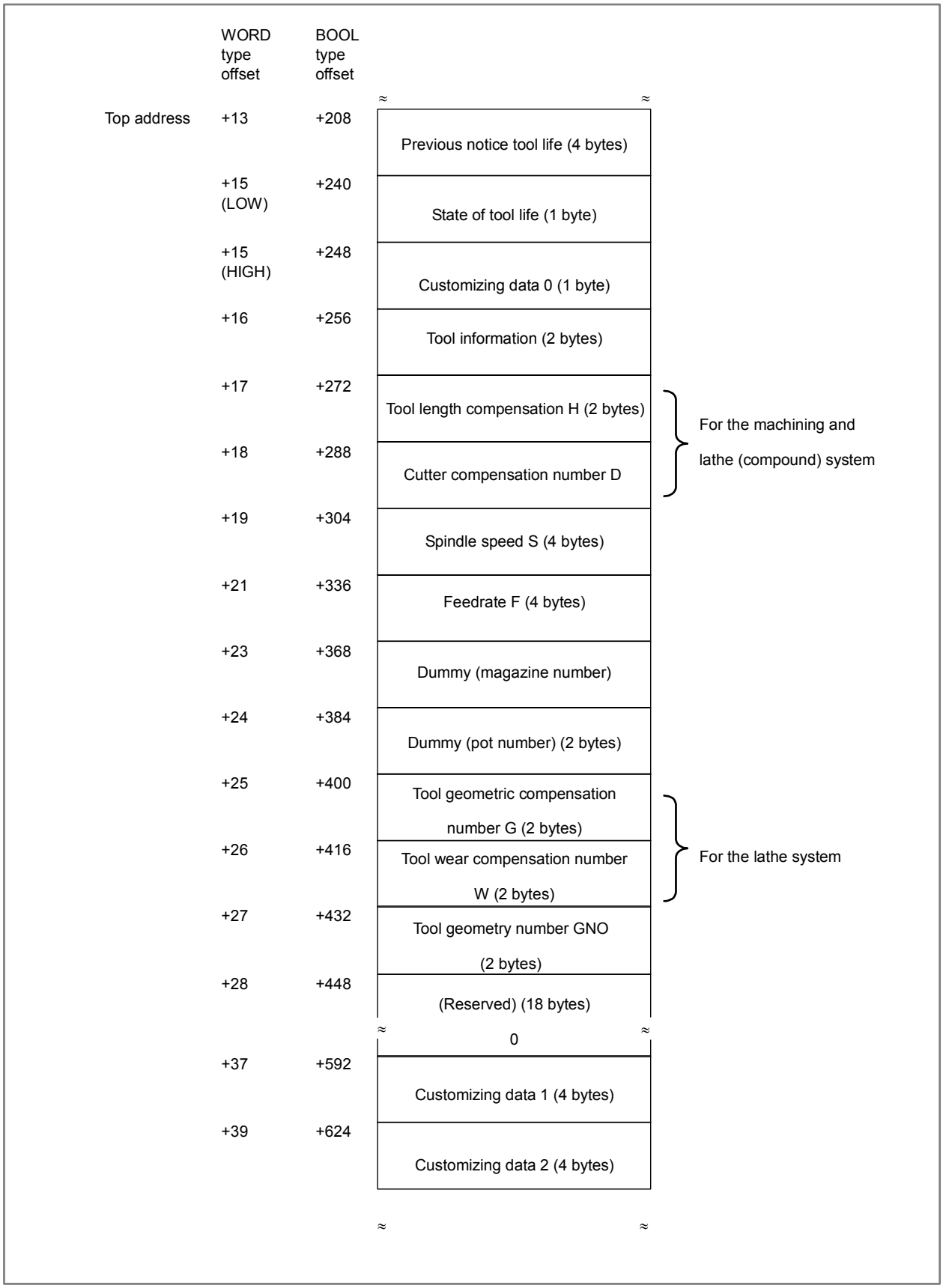
This function registers new tool data according to specified magazine and pot number. The tool data was registered at the first vacant area from the top of the database in NC system.
Vacant area means that tool management data of the area is ignored. (It is equal to the state that bit 0 of tool information is 0.) If there is no vacant area, completion code 8 is returned.
If a set of specified magazine and pot number is allocated, completion code 13 is returned.
When tool management function oversize tools support is effective, if a set of specified magazine and pot number is the empty pot but interferes with other tools because of tool geometry, then completion code 13 is returned.

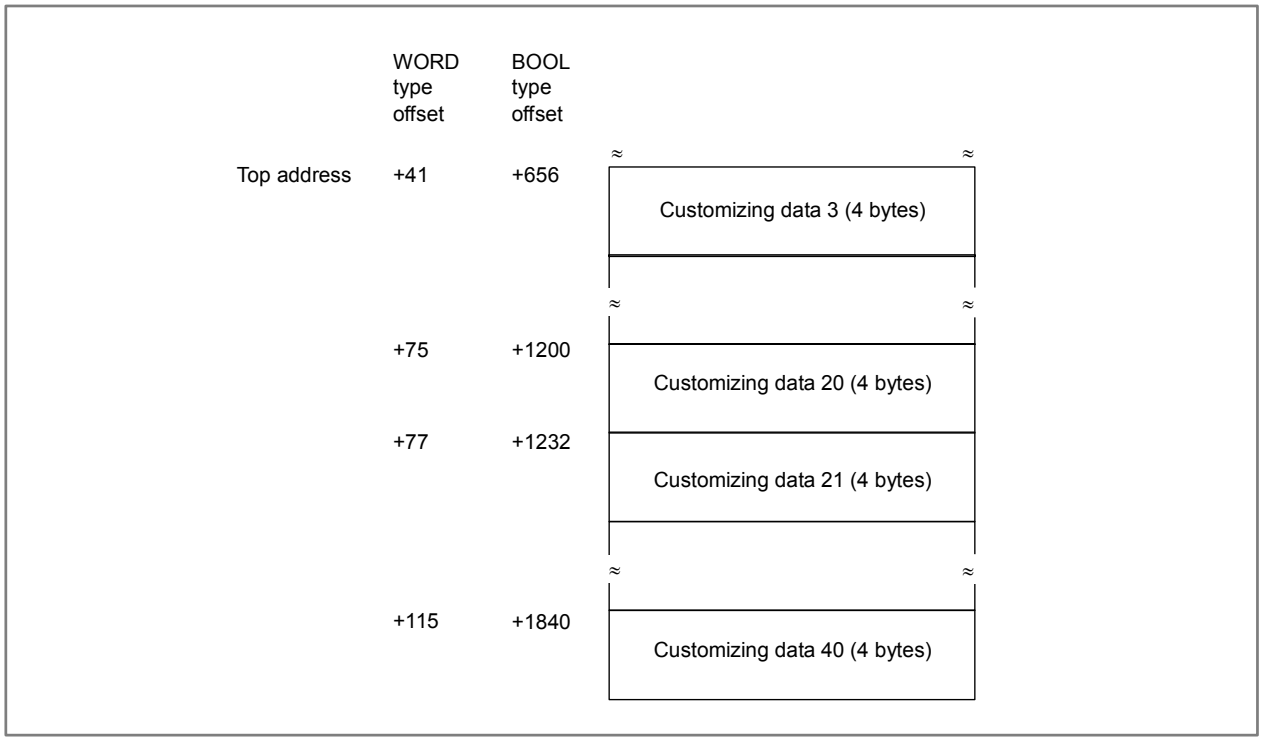
NOTE
The data length varies depending on whether the "tool management function customized data extension (5 to 20)" and "tool management function customized data extension (5 to 40)" options are present or not.

(a) Data length
76: When there is no option
140: When the "tool management function customized data extension (5 to 20)" option is present
220: When the "tool management function customized data extension (5 to 40)" option is present

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 331
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in byte) 76, 140, or 220
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	Tool type number (4 bytes)
	+9	+144	Tool life counter (4 bytes)
	+11	+176	Maximum of tool life (4 bytes)
	+13	+208	
	≈		≈


WORD type offset
BOOL type offset
Top address
+13
+208
Previous notice tool life (4 bytes)
+15 (LOW)
+240
State of tool life (1 byte)
+15 (HIGH)
+248
Customizing data 0 (1 byte)
+16
+256
Tool information (2 bytes)
+17
+272
Tool length compensation H (2 bytes)
+18
+288
Cutter compensation number D
For the machining and lathe (compound) system
+19
+304
Spindle speed S (4 bytes)
+21
+336
Feedrate F (4 bytes)
+23
+368
Dummy (magazine number)
+24
+384
Dummy (pot number) (2 bytes)
+25
+400
Tool geometric compensation number G (2 bytes)
+26
+416
Tool wear compensation number W (2 bytes)
For the lathe system
+27
+432
Tool geometry number GNO (2 bytes)
+28
+448
(Reserved) (18 bytes)
0
+37
+592
Customizing data 1 (4 bytes)
+39
+624
Customizing data 2 (4 bytes)



[Completion codes]

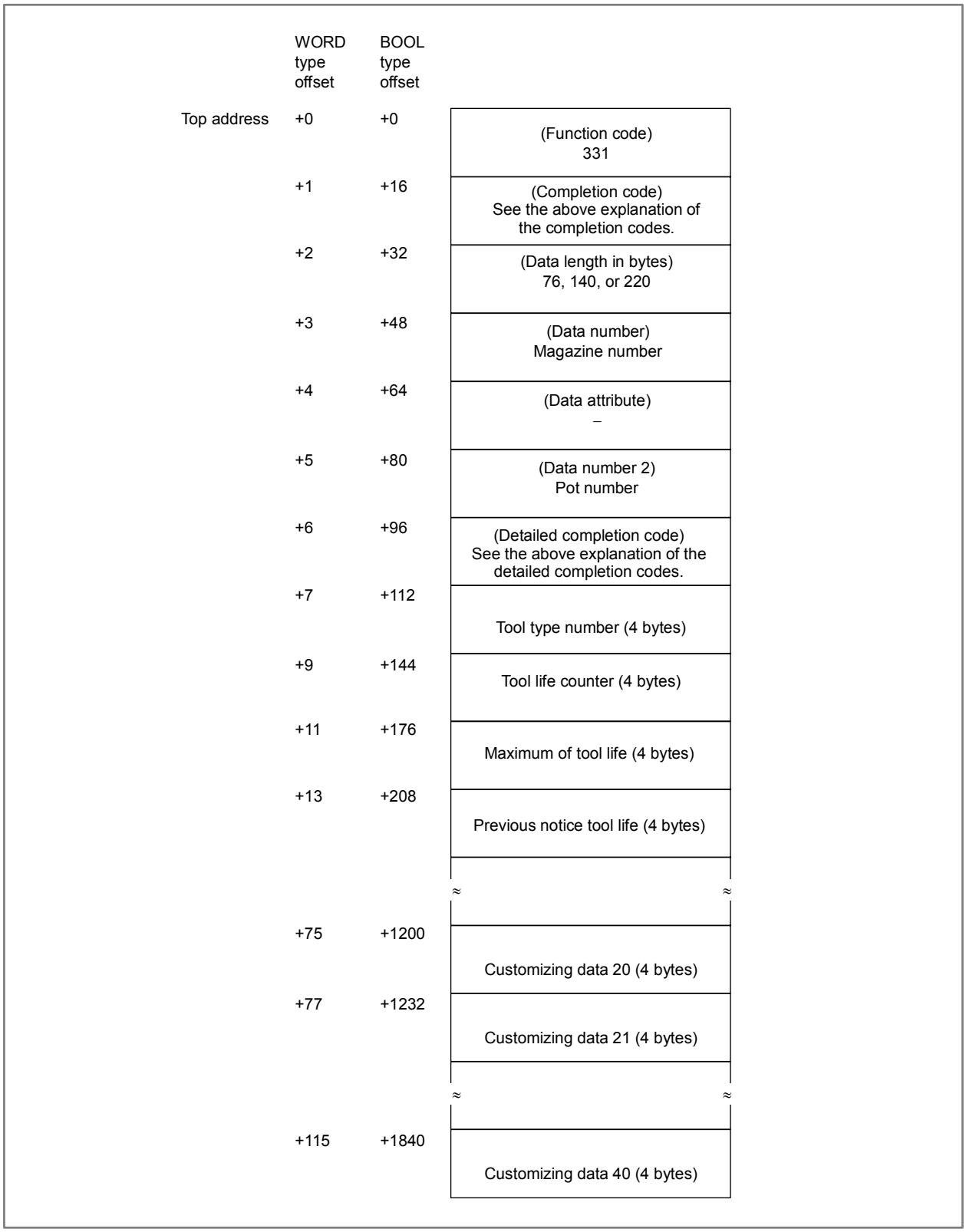
0: Normal end
2: Error of data length
3: Error of magazine or pot number
5: There is an error in data area / Tool interference
6: No required option
7: Protected area
8: There is no space
13: Already allocated

[Detailed completion codes]

This code is 0 except completion code is 5.
Completion code = 5

1: Error of tool type number
2: Error of tool life counter
3: Error of maximum of tool life
4: Error of previous notices tool life
5: Error of state of tool life
7: Error of tool information
8: Error of tool length compensation number (H) (Machining and Lathe (Compound) system)
9: Error of cutter compensation number (D) (Machining and Lathe (Compound) system)
10: Error of spindle speed (S)
11: Error of feedrate (F)
12: Error of tool geometric compensation number (G) (Lathe system)
13: Error of tool wears compensation number (W) (Lathe system)
14: Error of tool geometry number (GNO)
27: Interfere with other tools
31 to 50: Error of customizing data 1 - 20
51 to 70: Error of customizing data 21 - 40

[Output data structure]



A.7.4 Writing a Tool Management Data (Low-speed Response)

[Description]

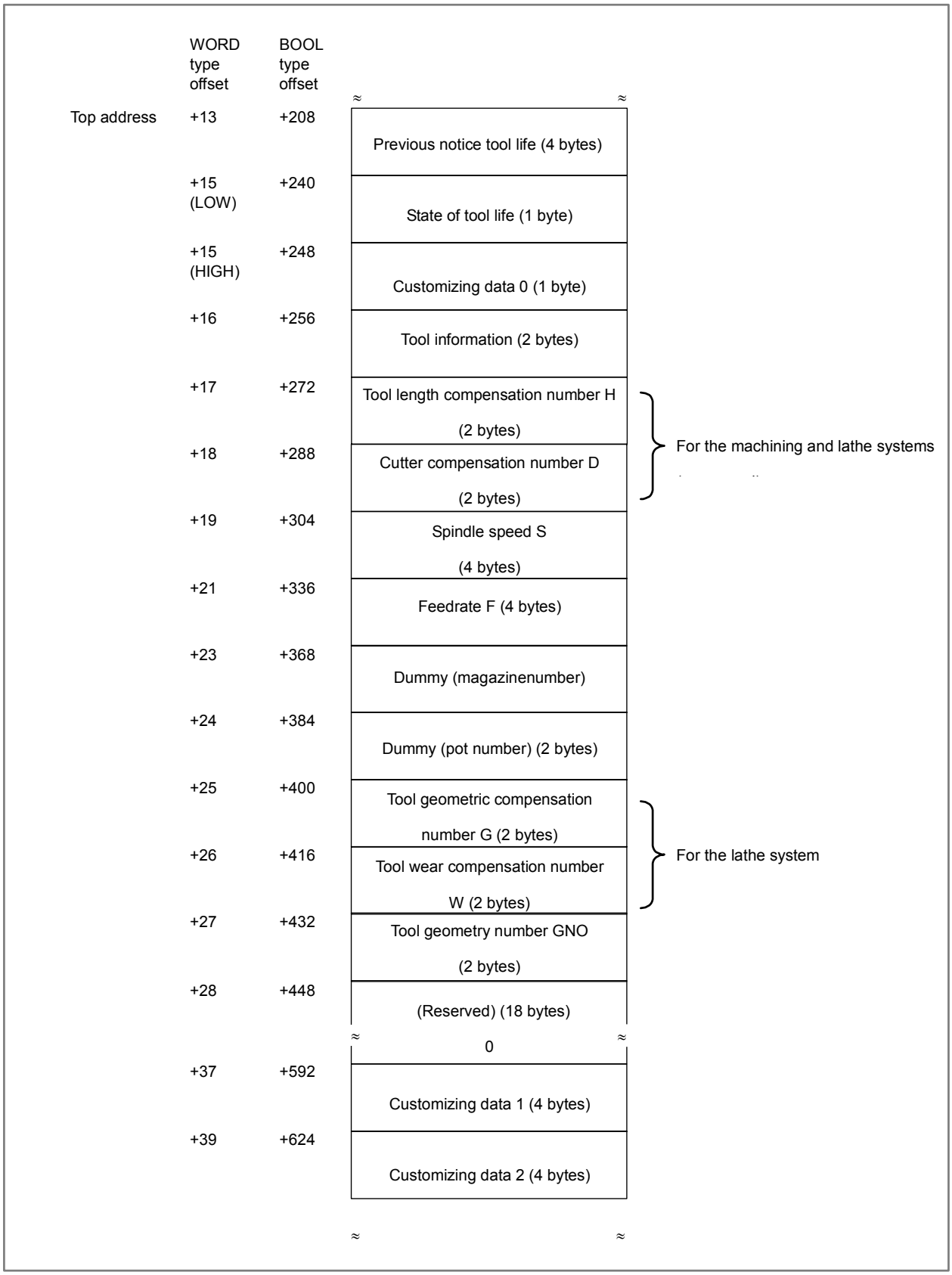
This function writes a tool data according to specified magazine and pot number. If the pot has no tool (if tool management data number is not allocated), error 9 is output.

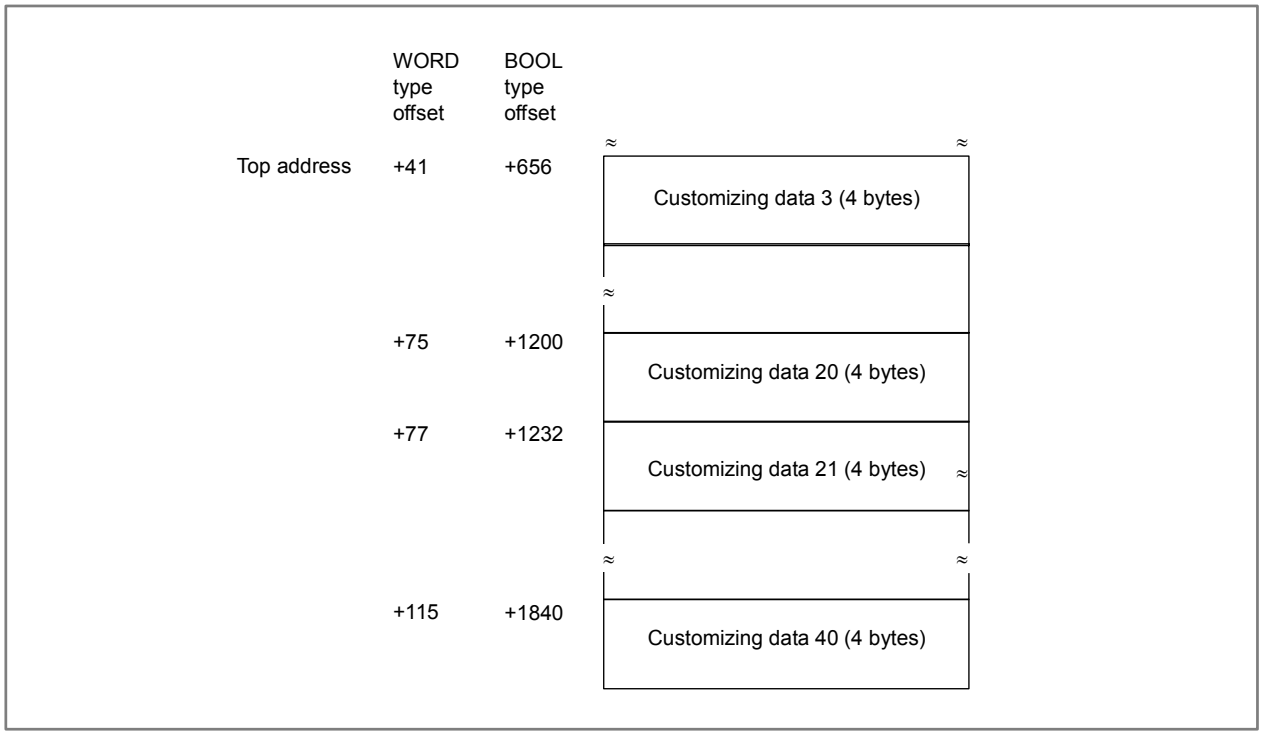
NOTE
The data length varies depending on whether the "tool management function customized data extension (5 to 20)" and "tool management function customized data extension (5 to 40)" options are present or not.

(a) Data length
- 76: When there is no option
- 140: When the "tool management function customized data extension (5 to 20)" option is present
- 220: When the "tool management function customized data extension (5 to 40)" option is present

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 332
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 76, 140, or 220
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	Tool type number (4 bytes)
	+9	+144	Tool life counter (4 bytes)
	+11	+176	Maximum of tool life (4 bytes)
	+13	+208	
	≈	≈	


WORD type offset
BOOL type offset
Top address
+13
+208
Previous notice tool life (4 bytes)
+15 (LOW)
+240
State of tool life (1 byte)
+15 (HIGH)
+248
Customizing data 0 (1 byte)
+16
+256
Tool information (2 bytes)
+17
+272
Tool length compensation number H (2 bytes)
+18
+288
Cutter compensation number D (2 bytes)
For the machining and lathe systems
+19
+304
Spindle speed S (4 bytes)
+21
+336
Feedrate F (4 bytes)
+23
+368
Dummy (magazinenumber)
+24
+384
Dummy (pot number) (2 bytes)
+25
+400
Tool geometric compensation number G (2 bytes)
+26
+416
Tool wear compensation number W (2 bytes)
For the lathe system
+27
+432
Tool geometry number GNO (2 bytes)
+28
+448
(Reserved) (18 bytes)
0
+37
+592
Customizing data 1 (4 bytes)
+39
+624
Customizing data 2 (4 bytes)



[Completion codes]

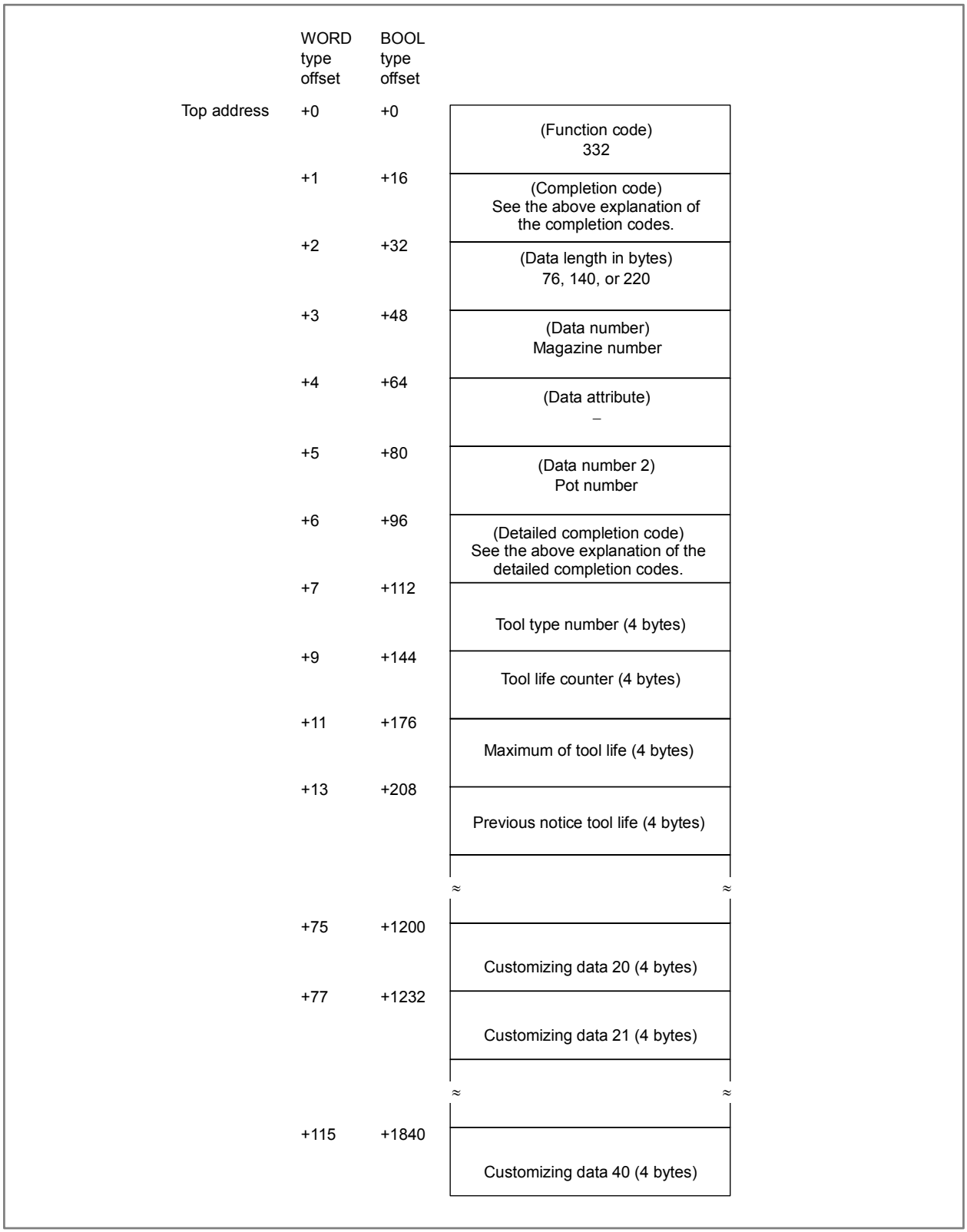
0: Normal end
2: Error of data length
3: Error of magazine or pot number
5: There is an error in data area / Tool interference
6: No required option
7: Protected area
9: The pot has no tool

[Detailed completion codes]

This code is 0 except completion code is 3 or 5.
Completion code = 3
- 21: Error of magazine number
- 22: Error of pot number
- 23: Error of tool management data number

Completion code = 5
- 1: Error of tool type number
- 2: Error of tool life counter
- 3: Error of maximum of tool life
- 4: Error of previous notices tool life
- 5: Error of state of tool life
- 7: Error of tool information
- 8: Error of tool length compensation number (H) (Machining and Lathe (Compound) system)
- 9: Error of cutter compensation number (D) (Machining and Lathe (Compound) system)
- 10: Error of spindle speed (S)
- 11: Error of feedrate (F)
- 12: Error of tool geometric compensation number (G) (Lathe system)
- 13: Error of tool wears compensation number (W) (Lathe system)
- 14: Error of tool geometry number (GNO)
- 27: Interfere with other tools
- 31 to 50: Error of customizing data 1 - 20
- 51 to 70: Error of customizing data 21 - 40

[Output data structure]



A.7.5 Deletion of a Tool Management Data (Low-speed Response)

[Description]

This function deletes a tool management data according to specified magazine and pot number.
If the pot has no tool (if tool management data number is not allocated), error 9 is output.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 333
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 0
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)

[Completion codes]

0: Normal end
3: Invalid magazine number or pot number
6: No required option
7: Protected area
9: The pot has no tool

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3

21: Error of magazine number
22: Error of pot number
23: Error of tool management data number

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 333
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 0
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) –
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.

A.7.6 Reading a Tool Management Data (Low-speed Response)

[Description]

This function reads a tool management data according to specified magazine and pot number. If the pot has no tool (if tool management data number is not allocated), error 9 is output.

NOTE

1 Customizing data 5~20 can be read when the option " Additional customized data on tool management function (5~20)" or " Additional customized data on tool management function (5~40)" exists.
Customizing data 21~40 can be read when the option "Additional customized data on tool management (5~40)" exists.

2 When tool management function oversize tools support is effective, the tool geometry number can be read.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 334
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	(Data area) 76, 140, 220

[Completion codes]

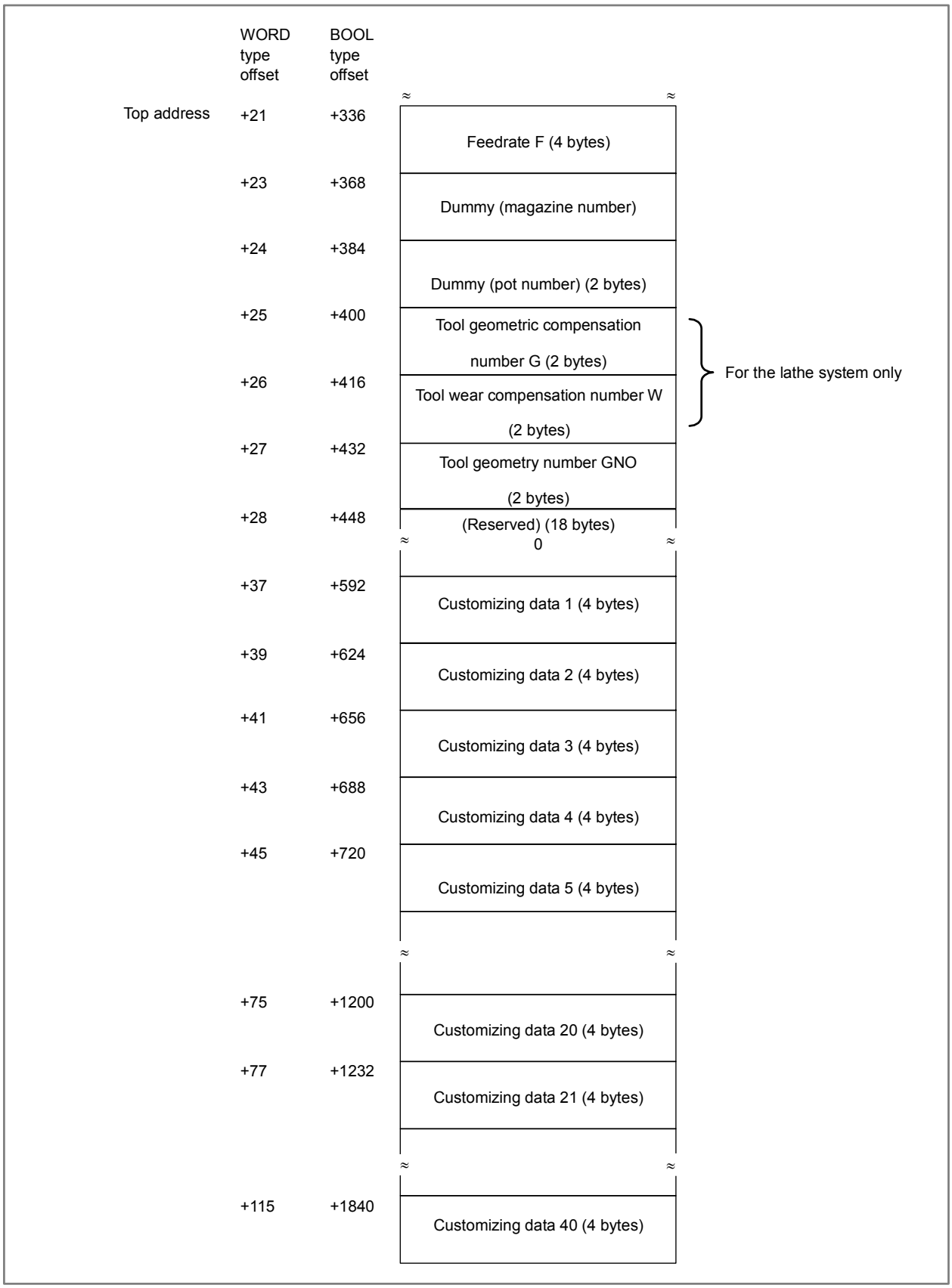
0: Normal end
3: Invalid magazine number or pot number
6: No required option
7: Protected area
9: The pot has no tool

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3
21: Error of magazine number
22: Error of pot number

[Output data structure]

	WORD type offset	BOOL type offset		
Top address	+0	+0	(Function code) 334	
	+1	+16	(Completion code) See the above explanation of the completion codes.	
	+2	+32	(Data length in bytes) 76, 140 or 230	
	+3	+48	(Data number) Magazine number	
	+4	+64	(Data attribute) –	
	+5	+80	(Data number 2) Pot number	
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.	
	+7	+112	Tool type number (4 bytes)	
	+9	+144	Tool life counter (4 bytes)	
	+11	+176	Maximum of tool life (4 bytes)	
	+13	+208	Previous notice tool life (4 bytes)	
	+15 (LOW)	+240	State of tool life (1 byte)	
	+15 (HIGH)	+248	Customizing data 0 (1 byte)	
	+16	+256	Tool information (2 bytes)	
	+17	+272	Tool length compensation number H (2 bytes)	For the machining and lathe (compound) systems
	+18	+288	Cutter compensation number D (2 bytes)	
	+19	+304	Spindle speed S (4 bytes)	
	≈		≈	


WORD type offset
BOOL type offset
Top address
+21
+336
Feedrate F (4 bytes)
+23
+368
Dummy (magazine number)
+24
+384
Dummy (pot number) (2 bytes)
+25
+400
Tool geometric compensation number G (2 bytes)
+26
+416
Tool wear compensation number W (2 bytes)
For the lathe system only
+27
+432
Tool geometry number GNO (2 bytes)
+28
+448
(Reserved) (18 bytes)
0
+37
+592
Customizing data 1 (4 bytes)
+39
+624
Customizing data 2 (4 bytes)
+41
+656
Customizing data 3 (4 bytes)
+43
+688
Customizing data 4 (4 bytes)
+45
+720
Customizing data 5 (4 bytes)
+75
+1200
Customizing data 20 (4 bytes)
+77
+1232
Customizing data 21 (4 bytes)
+115
+1840
Customizing data 40 (4 bytes)

A.7.7 Writing each Tool Data (Low-speed Response)

[Description]

This function writes specified data into tool management data.
The type of written data is input to the data attribute area.
The size of a necessary data area changes depending on the data type.
The relation among a data type, an input value and a necessary data area size is as follows.

Table A.7.7 Input value to the data attribute and the data area size

Input value	Data type	Data area size	Remarks
1	Tool type number	4 bytes	
2	Tool life counter	4 bytes	
3	Maximum of tool life	4 bytes	
4	Previous tool life	4 bytes	
5	State of tool life	4 bytes	
6	Customizing data 0	1 byte	Bit type
7	Tool information	2 bytes	
8	Tool length compensation number (H)	2 bytes	Machining and Lathe (Compound) system
9	Cutter compensation number (D)	2 bytes	Machining and Lathe (Compound) system
10	Spindle speed (S)	4 bytes	
11	Feedrate (F)	4 bytes	
12	Tool geometric compensation number (G)	2 bytes	Lathe system
13	Tool wear compensation number (W)	2 bytes	Lathe system
14	Tool geometry number (GNO)	2 bytes	
31	Customizing data 1	4 bytes	
32	Customizing data 2	4 bytes	
33	Customizing data 3	4 bytes	
34	Customizing data 4	4 bytes	
35	Customizing data 5	4 bytes	
36	Customizing data 6	4 bytes	
≈	≈	≈	≈
50	Customizing data 20	4 bytes	
51	Customizing data 21	4 bytes	
≈	≈	≈	≈
70	Customizing data 40	4 bytes	

NOTE
1 Customizing data 5~20 can be read when the option " Additional customized data on tool management function (5~20)" or " Additional customized data on tool management function (5~40)" exists.
Customizing data 21~40 can be read when the option "Additional customized data on tool management (5~40)" exists.
2 Tool geometry number (GNO) can be written when the option "Tool management function oversize tools support" exists.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 335
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 1, 2, 4
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Data type
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	(Data area) 1, 2, 4

[Completion codes]

0: Normal end
3: Invalid magazine number, pot number or tool management data number
4: Error of data type
5: Error of data area / Tool interference
6: No required option
7: Protected area

[Detailed completion codes]

This code is 0 except completion code is 3 or 5.
Completion code = 3
- 21: Error of magazine number
- 22: Error of pot number
- 23: Error of tool management data number

Completion code = 5
- 1: Error of tool type number
- 2: Error of tool life counter
- 3: Error of maximum of tool life
- 4: Error of previous notices tool life
- 5: Error of state of tool life
- 7: Error of tool information
- 8: Error of tool length compensation number (H) (Machining and Lathe (Compound) system)
- 9: Error of cutter compensation number (D) (Machining and Lathe (Compound) system)
- 10: Error of spindle speed (S)
- 11: Error of feedrate (F)
- 12: Error of tool geometric compensation number (G) (Lathe system)
- 13: Error of tool wears compensation number (W) (Lathe system)
- 14: Error of tool geometry number (GNO)
- 27: Interfere with other tools
- 31 to 50: Error of customizing data 1 - 20
- 51 to 70: Error of customizing data 21 - 40

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 335
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 1, 2, 4
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Data type
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	(Data area) 1, 2, 4

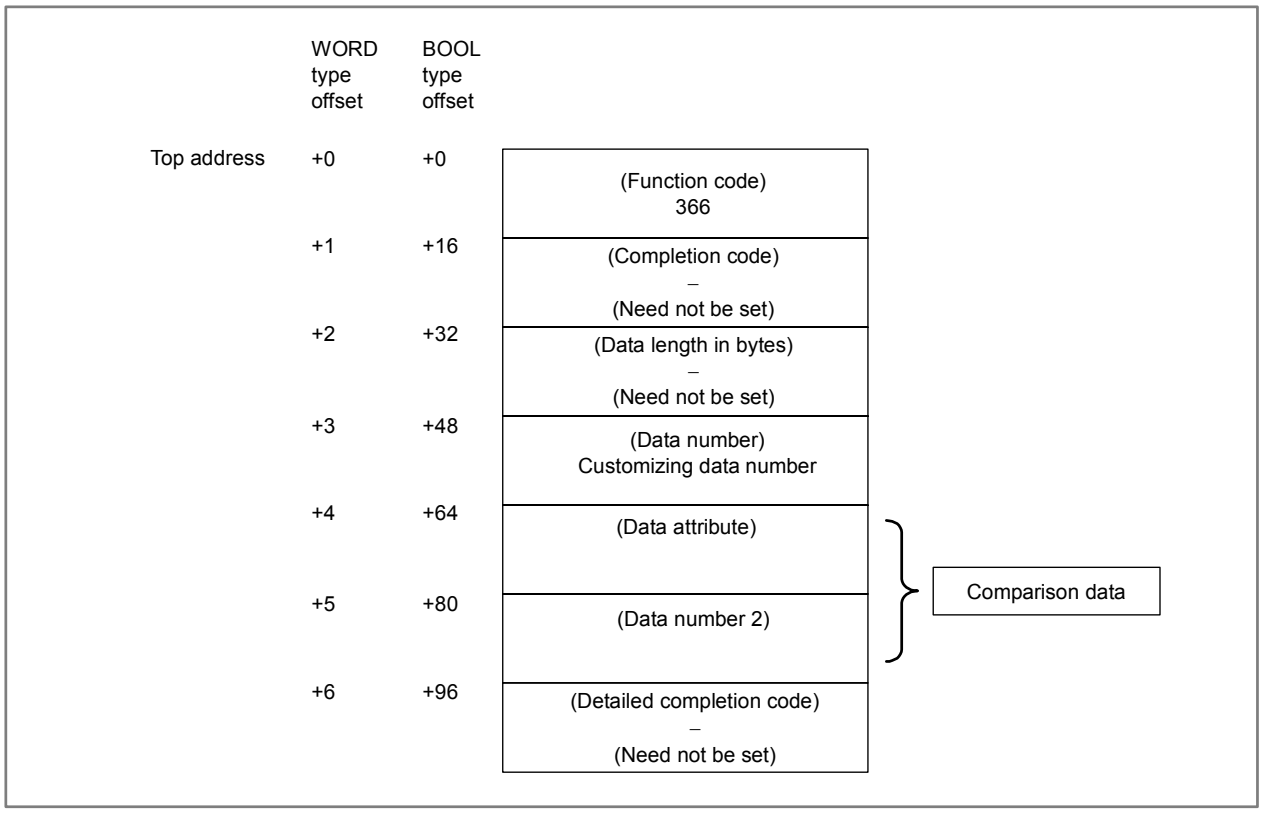
A.7.8 Search of Tool Management Data (Low-speed Response)

[Description]

This function searches the tool by customizing data.
This function searches whether there is corresponding data to the specified customizing data for the tool registered in the magazine management table, and returns the magazine and pot number of the tool found first.

Search order
1. Normal magazine
2. Spindle position
3. Wait position

[Input data structure]



[Completion codes]

0: Normal end
3: Invalid customizing data number
6: No required option

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3

1: Too small customizing data number (under 0)
2: Too large customizing data number
3: The tool is not found

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 366
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 4 or 0
	+3	+48	(Data number) –
	+4	+64	(Data attribute) –
	+5	+80	(Data number 2) –
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	Magazine number (2 bytes)
	+8	+128	Pot number (2 bytes)

A.7.9 Shifting Tool Management Data (Low-speed Response)

[Description]

This function shifts the magazine management table.
This function shifts tool management table for the fixed pot number type magazine.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 367
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 0
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Shift direction
	+5	+80	(Data number 2) Shift amount
	+6	+96	(Detailed completion code) – (Need not be set)

[Completion codes]

0: Normal end
3: Invalid magazine number or shift amount
4: Invalid shift direction
6: No required option

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3
1: Invalid magazine number
2: Invalid shift amount
(Specified under 0 or over magazine number)

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 367
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 0
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Shift direction
	+5	+80	(Data number 2) Shift amount
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.

The shift direction is defined follows.

Shift direction: 1
The tool data in pot 1 shifts pot 2.
The tool data in pot 2 shifts pot 3.
The tool data in pot 3 shifts pot 4.
The tool data in pot 4 shifts pot 5.
The tool data in pot 5 shifts pot 1.

Shift direction: -1
The tool data in pot 1 shifts pot 5.
The tool data in pot 2 shifts pot 1.
The tool data in pot 3 shifts pot 2.
The tool data in pot 4 shifts pot 3.
The tool data in pot 5 shifts pot 4.

Pot number	1	2	3	4	5
Direction: 1	7	3	4	5	6
Original condition	3	4	5	6	7
Direction: –1	4	5	6	7	3

A.7.10 Reading a Decimal Point of the Customizing Data (Low-speed Response)

[Description]

This function reads a decimal point of the customizing data.

NOTE
Customizing data 5~20 can be read when the option " Additional customized data on tool management function (5~20)" or " Additional customized data on tool management function (5~40)" exists.
Customizing data 21~40 can be read when the option "Additional customized data on tool management (5~40)" exists.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 392
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) – (Need not be set)
	+4	+64	(Data attribute) – (Need not be set)
	+5	+80	(Data area) – (Need not be set)

[Completion codes]

0: Normal end
6: No required option

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 392
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 4, 20, or 40
	+3	+48	(Data number) –
	+4	+64	(Data attribute) –
	+5 (LOW)	+80	Decimal point of the customizing data1 (1byte)
	+5 (HIGH)	+88	Decimal point of the customizing data2 (1byte)
	+6 (LOW)	+96	Decimal point of the customizing data3 (1byte)
	+6 (HIGH)	+104	Decimal point of the customizing data4 (1byte)
	+7 (LOW)	+112	Decimal point of the customizing data5 (1byte)
	≈		≈
	+14 (HIGH)	+232	Decimal point of the customizing data20 (1byte)
	≈		≈
	+24 (HIGH)	+392	Decimal point of the customizing data40 (1byte)

A.7.11 Search of Empty Pot for Oversize Tool Use (Low-speed Response)

[Description]

This function searches the nearest empty pot, which can store the specified oversize tool in the same magazine.
Spindle and waiting position are not searched as empty pot.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 397
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) 6
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Search direction
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) – (Need not be set)
	+7	+112	Tool geometry number (GNO) (2 bytes)

[Search direction]

–1: Descending order
0: No order (The nearest pot is searched)
1: Ascending order

[Completion codes]

0: Normal end
3: Error of magazine number / pot number / tool geometry number
4: Error of search direction
6: No required option
8: No empty pot

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3

21: Error of magazine number
22: Error of pot number
26: Error of tool geometry number

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 397
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 6
	+3	+48	(Data number) Magazine number
	+4	+64	(Data attribute) Search direction
	+5	+80	(Data number 2) Pot number
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	Tool geometry number (GNO) (2 bytes)
	+8	+128	Magazine number (2 bytes)
	+9	+144	Pot number (2 bytes)

When this function is executed without specifying search direction, the pot found in ascending direction is output if empty pots are found at the same distance away from the specified pot in both orders.

A.7.12 Reading a Total Life Data (Low-speed Response)

[Description]

This function reads total life data according to specified tool type number.

[Input data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 409
	+1	+16	(Completion code) – (Need not be set)
	+2	+32	(Data length in bytes) – (Need not be set)
	+3	+48	(Data number) Tool type number (4 bytes)
	+5	+80	(Data attribute) Display type (0: count / 1: time)
	+6	+96	(Detailed completion code) – (Need not be set)

[Completion codes]

0: Normal end
3: Error of tool type number
4: Error of display type
6: No required option

[Detailed completion codes]

This code is 0 except completion code is 3.
Completion code = 3

1: Error of tool type number (except 1 to 99,999,999)
28: Tool type number is not found

[Output data structure]

	WORD type offset	BOOL type offset	
Top address	+0	+0	(Function code) 409
	+1	+16	(Completion code) See the above explanation of the completion codes.
	+2	+32	(Data length in bytes) 26
	+3	+48	(Data number) Tool type number (4 bytes)
	+5	+80	(Data attribute) Display type (0: count / 1: time)
	+6	+96	(Detailed completion code) See the above explanation of the detailed completion codes.
	+7	+112	Tool type number (4 bytes)
	+9	+144	Tool life counter (4 bytes)
	+11	+176	Total remain life (4 bytes)
	+13	+208	Total maximum of life (4 bytes)
	+15	+240	Tool notice life (4 bytes)
	+17	+272	Number (2 bytes)
	+18 (LOW)	+288	Status (1 byte)
	+18 (HIGH)	+296	Display type (0: count / 1: time) (1byte)

B DIFFERENCES BETWEEN PMC-SB7

Those users who are already familiar with the specifications of CNC system with PMC-SB7 should note the differences between PMC-SB7 and PMC-SD7. This chapter explains such differences between the two types of PMC.

Availability of Additional Options

There are some differences in the additional options that can be used with the CNC system related to the type of PMC. When you are going to use the following options, please be careful to specify correct combinations of the options.

No.	Items	PMC-SD7	PMC-SB7
1	Loader control board	Not supported. (Refer to Appendix B.1)	Supported.
2	PMC C language board	Not supported. (Refer to Appendix B.1)	Supported.
3	Programming software on PC	FANUC LADDER-IIIC (Refer to Appendix B.2)	FANUC LADDER-III
4	PMC operation software on Open CNC	FANUC LADDER-IIIC for Open CNC (Refer to Appendix B.2)	Ladder Editing Package
5	Machine Remote Diagnosis Package	PMC diagnosis functions are not available. (Refer to Appendix B.3)	Full functions are available.

Differences Concerning Application Programming

There are following differences in the programming and setting of PMC sequence program and CNC customizing software.

No.	Items	PMC-SD7	PMC-SB7
1	Programming language and available instructions	Instructions are basically different. However, most of the instructions of PMC-SD7 correspond the ones of PMC-SB7. Sequence Program Conversion Tool (A08B-9210-J525) is provided for conversion of instructions from PMC-SB7. Refer to B-64213EN (FANUC PMC Program Conversion Tool OPERATOR'S MANUAL) and other chapters in this manual.	Refer to B-61863E (FANUC PMC-MODEL PA1/ PA3/ SA1/ SA2/ SA3/ SA5/ SB/ SB2/ SB3/ SB4/ SB5/ SB6/ SB7/ SC/ SC3/ SC4/ NB/ NB2/ NB6 Ladder Language Programming Manual)
2	PMC program monitoring and editing on CNC	Ladder display and edit screens are not built in CNC. Use an Open CNC type display unit running FANUC LADDER-IIIC software instead. (Refer to Appendix B.4)	Ladder display and edit screens are built in CNC.
3	Levels of PMC sequence program	Level-1 and level-2 are available. Level-2 can cover all the features of level-3. （Refer to Appendix B.5）	In addition to level-1 and level-2, level-3 is available.
4	PMC setting parameter for "Selectable I/O Link Assignment Function"	Not supported. As the assignment data can be set up for each machine, this function is not necessary. （Refer to Appendix B.6）	Supported.
5	PMC system parameter for "F0 Operator Panel"	Not supported. This function is seldom used now. （Refer to Appendix B.6）	Supported.
6	PMC setting parameter for "Multi-language Alarm/Operator Message Display Function"	Not supported. As PMC-SD7 is typically used with the HMI with alarm/operator messages on the screen implemented in PC-based display unit, the function for switching the languages by PMC program is not supported. （Refer to Appendix B.7）	Supported.
7	C language executor library function for "Reading PMC Message"	Not supported. Since it is rare to display PMC message on the application screen created by C language executor in CNC with PMC-SD7, this function is not supported. （Refer to Appendix B.7）	Supported.

Differently Implemented Equivalent Features

Even the equivalent functions used for the same purpose, different implementations may have been done regarding the screen display or operations related to the type of PMC. In case of using these functions, please be careful.

No.	Items	PMC-SD7	PMC-SB7
1	Setting for Timer/Counter	Neither specific screen nor system parameter setting is necessary because each timer/counter is programmed using freely assigned %R address in binary type only. （Refer to Appendix B.8）	Following PMC parameter screens for setting Timer/Counter specific address are to be used. - [PMCPRM] –[TIMER] for T address - [PMCPRM] –[COUNTR] for C address PMC system parameter for “Counter Data Type” is to be set.
2	Use of MDI keys on CNC for PMC address and symbol (variable name)	Some punctuation characters unavailable on MDI keys are omitted or substituted. For example, IEC61131-3 compliant PMC address with “%” like “%I1” is to be entered as “I1”. （Refer to Appendix B.9）	Most of punctuation characters are interpreted as they are entered. PMC address such as “X0.0” is entered as it is.
3	Display screens for FA network board	The same option boards are supported. Note that the PMC address in the setting or diagnosis screens for these boards is displayed in the PMC-SB7 address representation. （Refer to Appendix B.10）	Following FA network option boards are supported. - Profibus-DP - Device-Net - FL-Net - I/O Link-II
4	Editing program “On-the-Fly”	“Smart Download to PMC” of FANUC LADDER-IIIC supports the feature. （Refer to Appendix B.11）	On-line editing with FANUC LADDER-III is supported.
5	Protection of sub-program	Each sub-program (block) is protected individually with Ladder Key Lock Setting. （Refer to Appendix B.11）	All sub-programs are protected by one EDIT password and one READ password.

B.1 SUPPORT OF OPTION BOARDS

Following option boards and relevant software options are not supported with PMC-SD7.

- Loader control board
- PMC C language board

B.2 PROGRAMMING SOFTWARE ON PC

Different programming software, FANUC LADDER-III and FANUC LADDER-IIIC, are used for PMC-SB7 and PMC-SD7 respectively. FANUC LADDER-III for programming PMC-SB7 does not support PMC-SD7. Oppositely, FANUC LADDER-IIIC does not support PMC-SB7. FANUC LADDER-IIIC should be solely used for programming PMC-SD7.

When the CNC system is Open CNC type, Ladder Editing Package, which is the dedicated software for maintenance operation on CNC, is available for PMC-SB7. In case of PMC-SD7, such dedicated software is not provided. Instead, FANUC LADDER-IIIC for Open CNC, which is another type of software license intended for the use of FANUC LADDER-IIIC on an Open CNC type display unit connected with one CNC, is provided.

B.3 MACHINE REMOTE DIAGNOSIS PACKAGE

CNC with Ethernet interface features remote diagnosis functions. In case of PMC-SB7, functions equivalent to FANUC LADDER-III are integrated in the Machine Remote Diagnosis Package for the purpose of remote diagnosis on PMC. In case of PMC-SD7, such functions are not provided. If necessary, use any remote operation software in the market to operate FANUC LADDER-IIIC running on the Open CNC type CNC display unit.

B.4 MONITORING AND EDITING ON CNC

PMC-SD7 does not support the following functions for monitoring and editing PMC program which are available on PMC-SB7.

(1) Built-in Editing Function

The built-in editing function is not supported. Program editing is performed using FANNUC LADDER-IIIC software on the PC.
To delete PMC data (sequence programs, PMC parameters), hold down the following keys and turn on the system power.

Delete of sequence program: <O> + <X>
Delete of all PMC parameters: <O> + <Z>

(2) Debugging Function

Debugging functions other than the online function (ONLINE) are not supported.

B.5 LEVEL3

The 3rd level program was originally intended to execute low-priority program, which does not require critical response for the machine control, to reduce the load of the 2nd level program and to shorten the PMC execution cycle time by transferring some part of the program from the 2nd level to the 3rd level.

In case of PMC-SB7, with which the hardware performance has been highly improved, using the 2nd level is recommended and the 3rd level is simply meant for the compatibility with existing ladder programs created in the 3rd level for older models than PMC-SB7.

In case of PMC-SD7, with which hardware performance has been improved as well as PMC-SB7, the 3rd level is no longer needed for the reason of performance. When converting 3rd level programs from PMC-SB7 to PMC-SD7, use the 2nd level.

B.6 MISCELLANEOUS PMC PARAMETERS

The following PMC setting parameter and PMC system parameter are no longer supported by PMC-SD7 because they are rarely used even on the PMC-SB7.

(1) Selectable I/O Link Assignment Function (Setting Parameter)

This function enables the common use of a sequence program for several machines, which have different I/O device configuration with each other, by setting the parameter to enable/disable each group in I/O link assignment data. Typically, the configuration of I/O devices differs for each machine and assignment data is to be set up for each machine. Therefore, this function is not supported by PMC-SD7.

(2) F0 Operator Panel (System Parameter)

This setting is to support a very old type operator panel interface, which is rarely used now. As this function is meant for the compatibility with ladder programs created for the models before PMC-SB7, substitution of other types of operator panels are recommended.

B.7 MISCELLANEOUS FEATURES OF MESSAGE

The following features regarding PMC alarm/operator message display function is not supported by PMC-SD7.

(1) Multi-language Message Display (Setting Parameter)

PMC supports a message display function. In case of PMC-SB7, the following PMC setting parameters used for PMC program to determine the display language for alarm and operator message are available. This feature for language switching by PMC setting is not supported by PMC-SD7.

- MESSAGE SHIFT VALUE
- MESSAGE SHIFT START ADDRESS

(2) C language executor function to read PMC message

In case of PMC-SB7, user can read the text of message data from the C language executor program by calling a C-language function pmc_rdpcmsg(). This function is not supported by PMC-SD7.

B.8 FEATURES OF TIMER/COUNTER

The PMC-SB7 supports screens for displaying and setting the reference data for the timer and counter. These screens are not supported by the PMC-SD7, however.

There is a PMC system parameter for PMC-SB7 to select the counter data type among binary or BCD. However, this PMC system parameter does not exist for PMC-SD7 because only the binary counter is available.

B.9 USE OF MDI KEY FOR INPUT ON CNC

Programs for PMC-SD7 are mainly edited on PC with full keyboard. When operating on CNC with MDI keyboard on which some characters may not be unavailable, such characters are to be typed differently.

(1) IEC61131-3 address

The reference addresses defined in IEC61131-3 (ex. %I, %IF, %Q) may not be input by some type of MDI key. Input all the reference addresses without "%" (ex. I, IF, Q), and perform a search. (Ex. For trace address input, perform an address search using the PMCPRM screen)

(2) Symbol or variable name

The limitation to input SYMBOL name is as the following.

- The small letter may not be inputted. Input the capital.
- The '_' may not be input by MDI. Input the '-'.
- The '$' may not be input by MDI. Input the '&'.

B.10 FA NETWORK BOARD

Following FA network boards are supported by both PMC-SB7 and PMC-SD7. Most of the boards have specific setting and diagnosis screens on which some parameters in PMC memory address notation are shown. Even on PMC-SD7, such parameters are displayed in the PMC-SB7 address.

- Profibus-DP
- Device-Net
- FL-Net
- I/O Link-II

B.11 EDIT ON-THE-FLY

FANUC LADDER-IIIC supports the “Smart Download to PMC”. It works as “Editing On-the-Fly”. While the PMC is online and the PMC and the FANUC LADDER-IIIC project are equal, if some logic blocks are edited, the “Smart Download to PMC” transfers only the edited blocks into the PMC on which it is running.

B.12 PASSWORD FUNCTION

A password function for protecting the PMC program is not supported by PMC-SD7. Please use the Lock Settings for each ladder level in FANUC LADDER-IIIC side.

	PMC-SD7	PMC-SB7
Protection Unit	Logic Block (sub-program)	Whole program
Protection Type	Edit Lock: Block cannot be edited. View Lock: Block cannot be viewed. Permanent Edit Lock: Block can never be edited. Permanent View Lock: Block can never be viewed in an editor.	R - password: To protect against reading. RW - password: To protect against reading and writing.

Since PMC-SB7 only sets one or two passwords to the whole ladder program, it is easy to handle. On the other hand, PMC-SD7 can set various Lock Settings for each logic block. Therefore, more flexible protection can be allowed.

NOTE
Neither of the PMC-SD7 and PMC-SB7 can set a password / Lock Settings to the I/O link assignment data, message data, and symbol data.

C ALARM MESSAGE LIST

C.1 PMC ALARMS/SYSTEM ALARMS

Alarm number	Faulty location/corrective action	Contents
ER01 PROGRAM DATA ERROR	1) Re-input the sequence program. 2) Replace the master printed circuit board.	The sequence program is invalid.
ER02 PROGRAM SIZE OVER	1) Reduce the sequence program. 2) Contact FANUC to have a larger number-of-Ladder-steps option specified.	The sequence program is too large. The sequence program is invalid.
ER03 PROGRAM SIZE ERROR (OPTION)	1) Reduce the sequence program. 2) Contact FANUC to have a larger number-of-Ladder-steps option specified.	The sequence program exceeds the size specified by the number-of-Ladder-steps option.
ER04 PMC TYPE UNMATCH	Using an offline programmer, change the sequence program to that for the correct PMC type.	The setting of the type in the sequence program differs from the actual type.
ER06 PMC CONTROL SOFTWARE TYPE UNMATCH	Contact FANUC to specify certain PMC type.	The combination of CNC system configuration and PMC type is invalid. (Example: PMC-SB5 is used for a 3-path CNC system.)
ER07 NO OPTION(LADDER STEP)	1) Restore the backed up CNC parameter data. 2) Check the data sheet and re-input the CNC parameters. 3) Contact FANUC to specify a number-of-Ladder-steps option of the necessary size.	No number-of-Ladder-steps option is found.
ER08 OBJECT UNMATCH	1) Contact FANUC.	An unsupported function is used in the sequence program.
ER09 PMC LABEL CHECK ERROR PLEASE TURN ON POWER AGAIN WITH PUSH 'O'&'Z'. (CLEAR PMC SRAM)	1) Press and hold down the 'O' and 'Z' key combination, and turn the CNC back on. 2) Replace the backup battery. 3) Replace the master printed circuit board.	With a change in the PMC type, for example, the retention-type memory of the PMC must be initialized.
ER10 OPTION AREA NOTHING (xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC management software is not loaded correctly.
ER11 OPTION AREA NOTHING(xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC C board management software is not loaded correctly.
ER12 OPTION AREA ERROR(xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC management software is invalid. (The series of BASIC and OPTION do not match.)
ER13 OPTION AREA ERROR(xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC C board management software is invalid. (The series of BASIC and OPTION do not match.)
ER14 OPTION AREA VERSION ERROR (xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC management software is invalid. (The editions of BASIC and OPTION do not match.)
ER15 OPTION AREA VERSION ERROR (xxxx)	Contact FANUC to reconfigure the PMC management software.	The PMC C board management software is invalid. (The editions of BASIC and OPTION do not match.)

Alarm number	Faulty location/corrective action	Contents
ER16 RAM CHECK ERROR (PROGRAM RAM)	Replace the master printed circuit board.	The initialization of the memory used to store the sequence program failed.
ER17 PROGRAM PARITY	1) Re-input the sequence program. 2) Replace the master printed circuit board.	The parity of the sequence program is invalid.
ER18 PROGRAM DATA ERROR BY I/O	Re-input the sequence program.	While the sequence program was being read, an interrupt command was generated.
ER22 PROGRAM NOTHING	1) Re-input the sequence program. 2) Replace the master printed circuit board.	The sequence program is empty.
ER23 PLEASE TURN OFF POWER	Turn the CNC off and then back on.	With a change in the PMC type, for example, the power must be turned off and then back on.
ER25 SOFTWARE VERSION ERROR (PMCAOPT)	Contact FANUC to reconfigure the PMC management software.	The PMC management software is invalid. (The edition of PMCAOPT does not match.)
ER26 PMC CONTROL MODULE ERROR(PMCAOPT)	1) Contact FANUC to reconfigure the PMC management software. 2) 2) Replace the master printed circuit board.	The initialization of the PMC management software failed.
ER32 NO I/O DEVICE	1) Check that the I/O device is on. 2) Check that the I/O device was turned on before the CNC was turned on. 3) Check the connection of the cable.	An I/O device such as the I/O Link, connection unit, and Power Mate is not connected.
ER33 I/O LINK ERROR	Replace the master printed circuit board.	The LSI of the I/O Link is defective.
ER34 I/O LINK ERROR (xx)	1) Check the connection of the cable leading to a device in group xx. 2) Check that the I/O device was turned on before the CNC. 3) Replace that device in group xx in which the PMC control module is installed.	In a slave in group xx, an error occurred in communication with an I/O device.
ER35 TOO MUCH OUTPUT DATA IN GROUP (xx)	Reduce the amount of output data in group xx.	The amount of output data in I/O Link group xx exceeds the limit (33 bytes). The excess data is nullified.
ER36 TOO MUCH INPUT DATA IN GROUP (xx)	Reduce the amount of input data in group xx.	The amount of input data in I/O Link group xx exceeds the limit (33 bytes). The excess data is nullified.
ER38 MAX SETTING OUTPUT DATA OVER (xx)	Modify the total amount of output data in each group to 128 bytes or less.	The I/O Link I/O area is insufficient. (The allocation of any group after group xx on the output side is nullified.)
ER39 MAX SETTING INPUT DATA OVER (xx)	Modify the total amount of input data in each group to 128 bytes or less.	The I/O Link I/O area is insufficient. (The allocation of any group after group xx on the input side is nullified.)
ER40 I/O LINK-II SETTING ERROR (CHx)	Reconfigure the I/O Link-II.	The I/O Link-II setting is invalid. (CH1: Primary board, CH2: Secondary board)
ER41 I/O LINK-II MODE ERROR(CHx)	Reconfigure the I/O Link-II.	The I/O Link-II mode setting is invalid. (CH1: Primary board, CH2: Secondary board)

Alarm number	Faulty location/corrective action	Contents
ER42 I/O LINK-II STATION NO.ERROR (CHx)	Reconfigure the I/O Link-II.	The I/O Link-II station number setting is invalid. (CH1: Primary board, CH2: Secondary board)
ER97 I/O LINK FAILURE (CHx yyGROUP)	1) Check whether the cables of I/O devices in group yy are connected properly. 2) Check the power to each I/O device. 3) Check the parameter setting of the I/O link assignment data selection function.	The number of assigned I/O modules in group yy differs from that of I/O devices actually connected.
WN02 OPERATE ADDRESS ERROR	Modify the setting of the PMC system parameter, address of the operator's panel for Series 0.	The setting of the PMC system parameter, address of the operator's panel for Series 0, is invalid.
WN03 ABORT NC-WINDOW/EXIN	1) Check that the Ladder program is free from problems and then restart the Ladder program (by pressing the RUN key). 2) Turn the CNC off and then back on.	The Ladder program was stopped during communication between the CNC and PMC. Function instructions such as WINDR, WINDW, EXIN, and DISPB may not be executed normally.
WN05 PMC TYPE NO CONVERSION	Using an offline programmer, change the sequence program to that for the correct PMC type.	The setting of the type in the sequence program differs from the actual type. (Example: For the PMC-SB5, the Ladder program of the PMC-SA3/SA5 was transferred.)
WN06 TASK STOPPED BY DEBUG FUNC	To restart a user task that has been stopped, stop the sequence program and then execute it again.	When a PMC C board is used, a user task has been stopped due to a break by a debug function.
WN07 LADDER SP ERROR (STACK)	Modify the sequence program so that the subprogram nesting level is eight or less.	For a subprogram call with the function instruction CALL or CALLU, the nesting level is too deep (exceeds 8).

C.2 PMC SYSTEM ALARM MESSAGES

Alarm number	Faulty location/corrective action	Contents
PC004 CPU ERR xxxxxxxx:yyyyyyyy PC006 CPU ERR xxxxxxxx:yyyyyyyy PC009 CPU ERR xxxxxxxx:yyyyyyyy PC010 CPU ERR xxxxxxxx:yyyyyyyy PC012 CPU ERR xxxxxxxx:yyyyyyyy	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	A CPU error occurred in the PMC. xxxxxxxx and yyyyyyyyy are internal error codes.
PC030 RAM PARITY aa:bb	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.) and the above internal error codes.	A RAM parity error occurred in the PMC. aa and bb are internal error codes.
PC050 NMI SLC aa:bb PC050 I/O LINK(CH1) aa:bb aa:bb PC050 IOLINK CH1 aaaa-bbbb:cccc	1) Check that the I/O allocation data matches the actual I/O device connection. 2) Check that the cable is connected properly. 3) Check the cable specifications. 4) Replace the I/O device interface module, cable, master printed circuit board, etc.	A communication error occurred in the I/O LINK. aa, bb and cc are internal error codes. If this alarm is generated, probable causes include the following: 1) Although the base expansion is assigned when the I/O Unit A is used, the base is not connected. 2) A cable is not connected securely. 3) Cabling is faulty. 4) I/O equipment (I/O unit, Power Mate, etc.) is faulty. 5) Power failure of Master or Slave device on I/O Link 6) Short circuit of DO terminal on I/O device 7) The motherboard is faulty.
PC060 FBUS xxxxxxxx:yyyyyyyy PC061 FL-R xxxxxxxx:yyyyyyyy	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	A bus error occurred in the PMC.
PC070 SUB65 CALL (STACK)	Check the correspondence between the CALL/CALLU and SPE instructions.	A stack error occurred in Ladder function instruction CALL/CALLU.
PC090 NMI(___) xxxxxxxx:yyyyyyyy	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	An NMI with an unknown cause occurred in the PMC management software.

Alarm number	Faulty location/corrective action	Contents
PC092 USER TRAP aa:xxxxxxxx	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	The TRAP instruction, which is not used in the PMC management software, was executed.
PC093 INT(SYS) xxxxxxxx:yyyyyyyy PC094 INT(TRAP) xxxxxxxx:yyyyyyyy PC095 INT(EX) xxxxxxxx:yyyyyyyy PC096 INT(IN) xxxxxxxx:yyyyyyyy	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	An interrupt with an unknown cause occurred in the PMC management software.
PC087 PARITY ERR (LADDER-2) PC097 PARITY ERR (LADDER) PC098 PARITY ERR (DRAM)	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	An error occurred in a RAM check.
PC501 NC/PMC INTERFACE ERR PATH_	1) Replace the master printed circuit board. 2) If the error recurs even after the replacement, contact FANUC to report the status (displayed message, system configuration, operation, when the error occurs, occurrence frequency, etc.).	The reading/writing of signals between the CNC and the PMC failed.
PC502 ILLEGAL FUNCTION (SUB xx)	Modify the sequence program so that instruction function xx is not used.	Unsupported function instruction xx is used.

D CHINESE CHARACTER CODE, HIRAGANA CODE, AND SPECIAL CODE LIST

When coding kanji character in message data, please used 4 digits of hexadecimal in ‘JIS” column in the following table.

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
alpha numeric ※	2330	824F	0	1	2	3	4	5	6	7	8	9						
	2340	825F		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	826F	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360	8280		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	8290	p	q	r	s	t	u	v	w	x	y	z					
hiragana	2420	829E		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	82AE	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	82BE	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	82CE	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	2460	82DE	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	82EE			を	ん												
katakana ※	2520	833F		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	834F	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	835F	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	836F	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	2560	8380	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	8390			ヲ	ン	ヴ	ヵ	ヶ									
Greek ※	2620	839E					Δ											
	2630	83AE			Σ						Ω							
	2640	83BE		α	β	γ		ε			θ				μ			
	2650	83CE	π					φ			ω							

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ア	3020	889E		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	88AE	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	88BE	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
イ	3040	88BE											以	伊	位	依	偉	囲
	3050	88CE	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	88DE	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	88EE	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120	893F		院	陰	隠	韻	吋										
ウ	3120	893F							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	894F	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	895F	雲															
エ	3140	895F		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	896F	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	8980	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	8990	艶	苑	薗	遠	鉛	鴛	塩									
オ	3170	8990								於	汚	甥	凹	央	奥	往	応	
	3220	899E		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	89AE	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
カ	3230	89AE													下	化	仮	何
	3240	89BE	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	89CE	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	89DE	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	89EE	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320	8A3F		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	8A4F	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	8A5F	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	3350	8A6F	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	8A80	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	3370	8A90	叶	椛	樺	鞄	株	兜	竃	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420	8A9E		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	8AAE	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	8ABE	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	8ACE	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	8ADE	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
キ	3460	8ADE												企	伎	危	喜	器
	3470	8AEE	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520	8B3F		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	8B4F	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	8B5F	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	8B6F	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	3560	8B80	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	8B90	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	3620	8B9E		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	8BAE	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	8BBE	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	8BCE	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	8BDE	謹	近	金	吟	銀											
ク	3660	8BDE						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	3670	8BEE	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720	8C3F		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君

shows the character which is impossible to display.

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ク	3730	8C4F	薫	訓	群	軍	郡											
ケ	3730	8C4F						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	8C5F	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	8C6F	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	3760	8C80	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	8C90	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820	8C9E		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	8CAE	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	8CBE	言	諺	限													
コ	3840	8CBE				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	8CCE	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	8CDE	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	8CEE	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920	8D3F		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	8D4F	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	3940	8D5F	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	8D6F	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降
	3960	8D80	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	8D90	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A20	8D9E		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	8DAE	紺	艮	魂													
サ	3A30	8DAE				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	8DBE	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	8DCE	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	8DDE	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	8DEE	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20	8E3F		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	8E4F	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	8E5F	酸	餐	斬	暫	残											
シ	3B40	8E5F						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	8E6F	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	8E80	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	8E90	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20	8E9E		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	8EAE	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	8EBE	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	8ECE	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	灼	爵	
	3C60	8EDE	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	8EEE	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20	8F3F		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D30	8F4F	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	8F5F	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D50	8F6F	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	8F80	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	8F90	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20	8F9E		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E30	8FAE	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	8FBE	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	8FCE	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	8FDE	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常

shows the character which is impossible to display.

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
シ	3E70	8FEE	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20	903F		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	904F	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	905F	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	906F	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						
ス	3F50	906F											筍	諏	須	酢	図	厨
	3F60	9080	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	9090	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020	909E		澄	摺	寸												
セ	4020	909E					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	90AE	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	90BE	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	90CE	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	90DE	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	90EE	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	4120	913F		繊	羨	腺	舛	船	薦	詮	賤	践	選	遷	銭	銑	閃	鮮
	4130	914F	前	善	漸	然	全	禅	繕	膳	糎							
ソ	4130	914F										噌	塑	岨	措	曾	曽	楚
	4140	915F	狙	疏	踈	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	916F	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	9180	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	9190	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	4220	919E		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	91AE	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
タ	4230	91AE															他	多
	4240	91BE	太	汰	詫	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	91CE	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	91DE	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	91EE	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320	923F		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	924F	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	925F	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
チ	4340	915F														値	知	地
	4350	926F	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	9280	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	9290	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	4420	929E		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	92AE	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	92BE	沈	珍	賃	鎮	陳											
ツ	4440	92BE						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	92CE	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壺	嬬	紬	爪	吊
	4460	92DE	釣	鶴														
テ	4460	92DE			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	4470	92EE	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520	933F		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	934F	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	4540	935F	点	伝	殿	澱	田	電										
ト	4540	935F							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	936F	登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬
	4560	9380	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟

shows the character which is impossible to display.

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ト	4570	9390	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	
	4620	939E		董	荡	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	93AE	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	93BE	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	93CE	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
ナ	4660	93DE	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	4670	93EE	軟	難	汝													
ニ	4670	93EE				二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	
	4720	943F		如	尿	韮	任	妊	忍	認								
ヌ	4720	943F									濡							
ネ	4720	943F										禰	祢	寧	葱	猫	熱	年
	4730	944F	念	捻	撚	燃	粘											
ノ	4730	944F						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	945F	農	覗	蚤													
ハ	4740	945F				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	946F	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	4760	9480	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
	4770	9490	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820	949E		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	94AE	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	4840	94BE	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	94CE	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ヒ	4850	94CE												匪	卑	否	妃	庇
	4860	94DE	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	94EE	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920	953F		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	954F	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	955F	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	956F	賓	頻	敏	瓶												
フ	4950	956F					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	9580	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	9590	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20	959E		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A30	95AE	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
ヘ	4A30	95AE											丙	併	兵	塀	幣	平
	4A40	95BE	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	95CE	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
ホ	4A50	95CE														保	舗	鋪
	4A60	95DE	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	95EE	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20	963F		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B30	964F	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	965F	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B50	966F	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
マ	4B60	9680	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B70	9690	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C20	969E		漫	蔓													
ミ	4C20	969E				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	96AE	粍	民	眠													
ム	4C30	96AE				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			

shows the character which is impossible to display.

Pronunciation	JIS	Shift JIS	Segment and point															
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
メ	4C30	96AE														冥	名	命
	4C40	96BE	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
モ	4C40	96BE															摸	模
	4C50	96CE	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	96DE	尤	戻	籾	貰	問	悶	紋	門	匁							
ヤ	4C60	96DE										也	冶	夜	爺	耶	野	弥
	4C70	96EE	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ユ	4C70	96EE												愉	愈	油	癒	
	4D20	973F		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D30	974F	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
ヨ	4D30	974F														予	余	与
	4D40	975F	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	976F	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
	4D60	977F	沃	浴	翌	翼	淀											
ラ	4D60	977F						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
	4D70	978F	乱	卵	嵐	欄	濫	藍	蘭	覧								
リ	4D70	978F									利	吏	履	李	梨	理	璃	
	4E20	979E		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	97AE	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	97BE	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	97CE	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
ル	4E50	97CE													塁	涙	累	
	4E60	97DE	類															
レ	4E60	97DE		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	97EE	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20	983F		蓮	連	錬												
ロ	4F20	983F					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	984F	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録
	4F40	985F	論															
ワ	4F40	985F		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	986F	椀	湾	碗	腕												

shows the character which is impossible to display.

INDEX

<G>

<I>

<J>

<L>

<M>

<N>

<O>

<P>

<R>

<S>

<T>

<U>

<W>

<X>

Revision Record

FANUC PMC-MODEL SD7 PROGRAMMING MANUAL (B-63823EN)

Edition	Date	Contents	Edition	Date	Contents
02	Nov., 2004	Addition of following items • I/O LINK • NOTE ON LD PROGRAMMING • NOTE ON IL PROGRAMMING • WINDOW FUNCTIONS • CHINESE CHARACTER CODE, HIRAGANA CODE, AND SPECIAL CODE LIST Correction of errors			
01	Oct., 2002	———————————————			