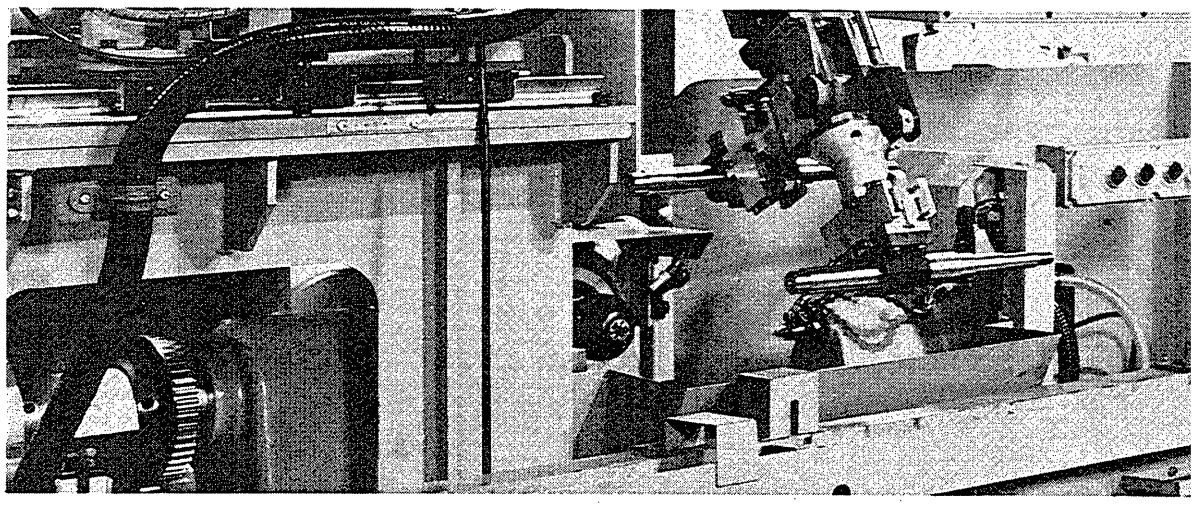
Motionpack FD Model 1
DESCRIPTIVE INFORMATION

YASKAWA MOTION CONTROLLER EXTENSION SYSTEM 1



YASKAWA

SIE-C883-1.2

INTRODUCTION

This Manual describes the functions and specifications of the Motionpack FD series model 1 (extension system 1).

The Motionpack FD series model 1 (extension system 1) is added with the built-in sequencer board (JAMP-S130) to the basic system model 0. The built-in sequencer and M-NET interface can be used in addition to the basic functions of the Motionpack FD series.

In this manual, M-NET interface and the built-in sequencer that is added to the model 1 is described.

For the funcions and specifications of the Motionpack FD series, refer to "Motionpack FD Series USER'S MANAUL" SIE-C883-1.1.

CONTENTS

CONTENTS (Cont'd)

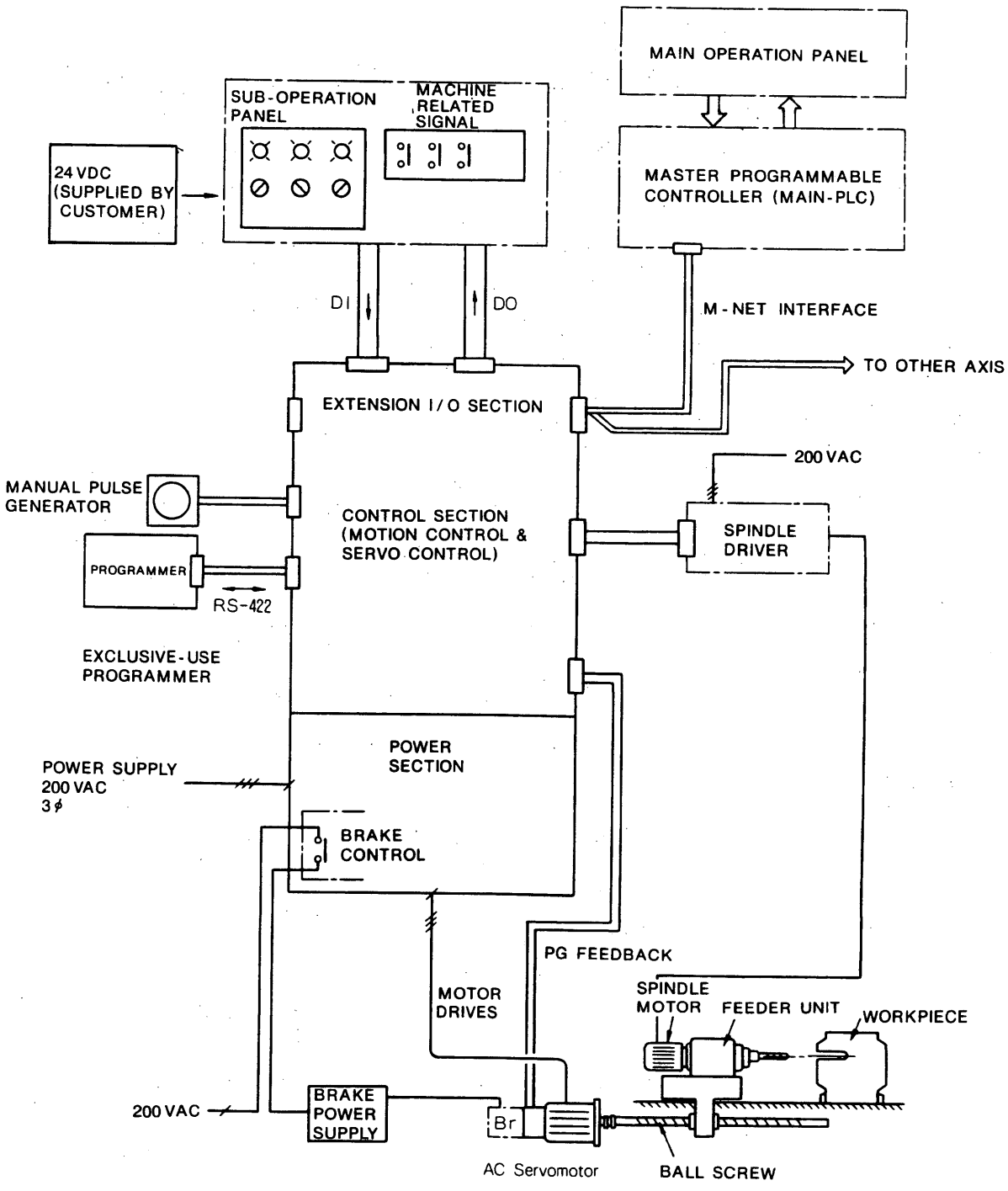
CONTENTS (Cont'd)

INDEX

INDEX (Cont'd)

1. Motionpack FD SERIES MODEL 1

1.1 SYSTEM CONFIGURATION



1.2 SPECIFICATIONS OF MODEL 1

The following table shows the functions that can be used for the model 1, compared with those of the basic system.

Table 1.2 Specifications of Model 1

Item	Basic System (Model 0)	Extension System (Model 1)
Type	CMPR-FD☐B0☐	CMPR-FD☐B1☐
Hardware	Basic section	Basic section + PLC board
Built-in PLC	Not available	Available
Solid Tap	Not available	Not available
No. of Programs	Up to 16	Up to 32
No. of Program Blocks	Up to 1000	Up to 1000
Standard I/O	I/O = 24/24	I/O = 24/24
Extended I/O	Not available	I/O = 24/24
M-NET Interface	Not available	Used (Y-mode, T-mode)
Spindle Reference	Analog reference ±10 V S-reference possible	Analog reference ±10 V S-reference possible
No. of Indirect Registers	R01 to R99	R01 to R99
Indirect Register Data Setting Method	Programmer	Programmer External Data Setting
External Data Setting	Not available	Available
Coordinate Compensation	Provided (T1 to T9)	Provided (T1 to T9)
External Compensation	Not available	Available

1.3 MODEL 1 PARAMETER SETTING

Set the model 1 the following parameters which are added to the basic system, if necessary.

Table 1.3 Additional Parameters for Model 1

<table>
<tr><th>Pr. No.</th><th>Name (Range/Unit)</th><th>Change</th><th>Description</th></tr>
<tr><td>Pr150</td><td>M-NET Interface Setting</td><td>P</td><td>
Pr150 = □□
└--- Baud Rate Setting
<table>
<tr><th>Set Value</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>4.8 kbps</td></tr>
<tr><td>1</td><td>9.6 kbps</td></tr>
<tr><td>2</td><td>19.2 kbps</td></tr>
<tr><td>3</td><td>38.4 kbps</td></tr>
</table>
└---- M-NET Interface Setting
<table>
<tr><th>Set Value</th><th>M-NET</th></tr>
<tr><td>0</td><td>Not provided</td></tr>
<tr><td>1</td><td>T-type</td></tr>
<tr><td>2</td><td>Y-type</td></tr>
</table>
EXAMPLE 1) Pr150 = 0 : M-NET not provided
EXAMPLE 2) Pr150 = 21 : Y-type provided
M-NET Baud rate 9.6 kbps
</td></tr>
</table>

Table 1.3 Additional Parameters for Model 1 (Cont'd)

Pr. No.	Name (Range/Unit)	Change	Description
Pr151	Transmission Points Setting	P	The following three conditions are set to Pr151. (1) No. of discrete input data transmission points (RSW1) (2) No. of discrete output data transmission points (RSW2) (3) No. of register data transmission points (RSW3) Pr151 is expressed in 5-decimal digit. Pr151 = □□□□□ ---- RSW1 (0 to 15) -------- RSW2 (0 to 15) ---------- RSW3 (0 to 7)

	RSW1	RSW2	
No.	DI (Point)	DO (Point)	No. of Connectable Slave Stations 1 2 3 4 5 6 7
0	0	0	
1	8	8	
2	16	16	
3	24	24	
4	32	32	
5	40	40	
6	48	48	
7	56	56	
8	64	64	
9	72	72	
10	80	80	
11	88	88	
12	96	96	
13	104	104	
14	120	120	
15	128	128	

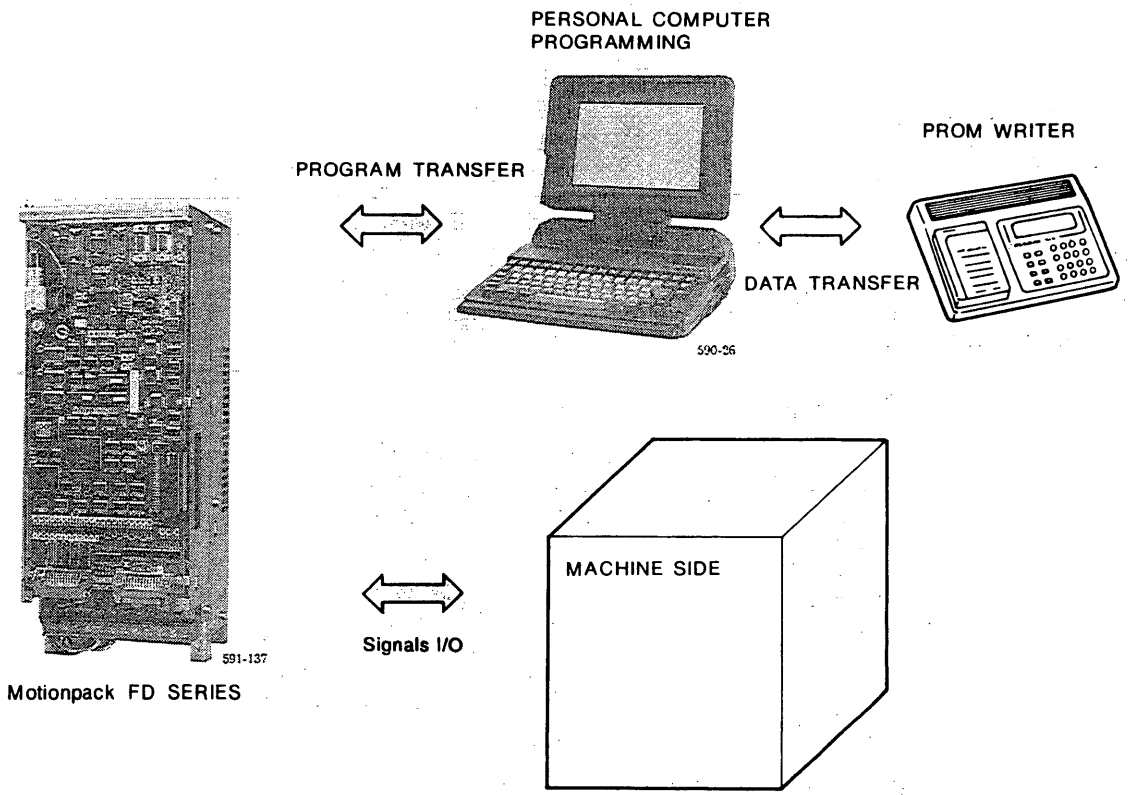
No.	RI (Point)	RO (Point)	No. of Connectable Slave Stations 1 2 3 4 5 6 7
0	0	0	
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	6	6	
7	7	7	
8	Setting Error		

Table 1.3 Additional Parameters for Model 1 (Cont'd)

Pr. No.	Name (Range/Unit)	Change	Description
Pr152	Keep-memory Heading Address (4800 to 5999/)	P	The heading address of keep-memory on the address map is set to Pr152. EXAMPLE 1) Pr152 = 5400 : #5400 to #5999 are the keep memory (keep registers).
Pr153	Built-in Sequencer Provided/Not Provided	P	Whether the built-in sequencer is provided or not is set to Pr153. Pr153 = □ ↑------ 0 : Not provided 1 : Provided

2. OUTLINE OF BUILT-IN SEQUENCER

2.1 PERIPHERAL DEVICES



2.2 SPECIFICATIONS

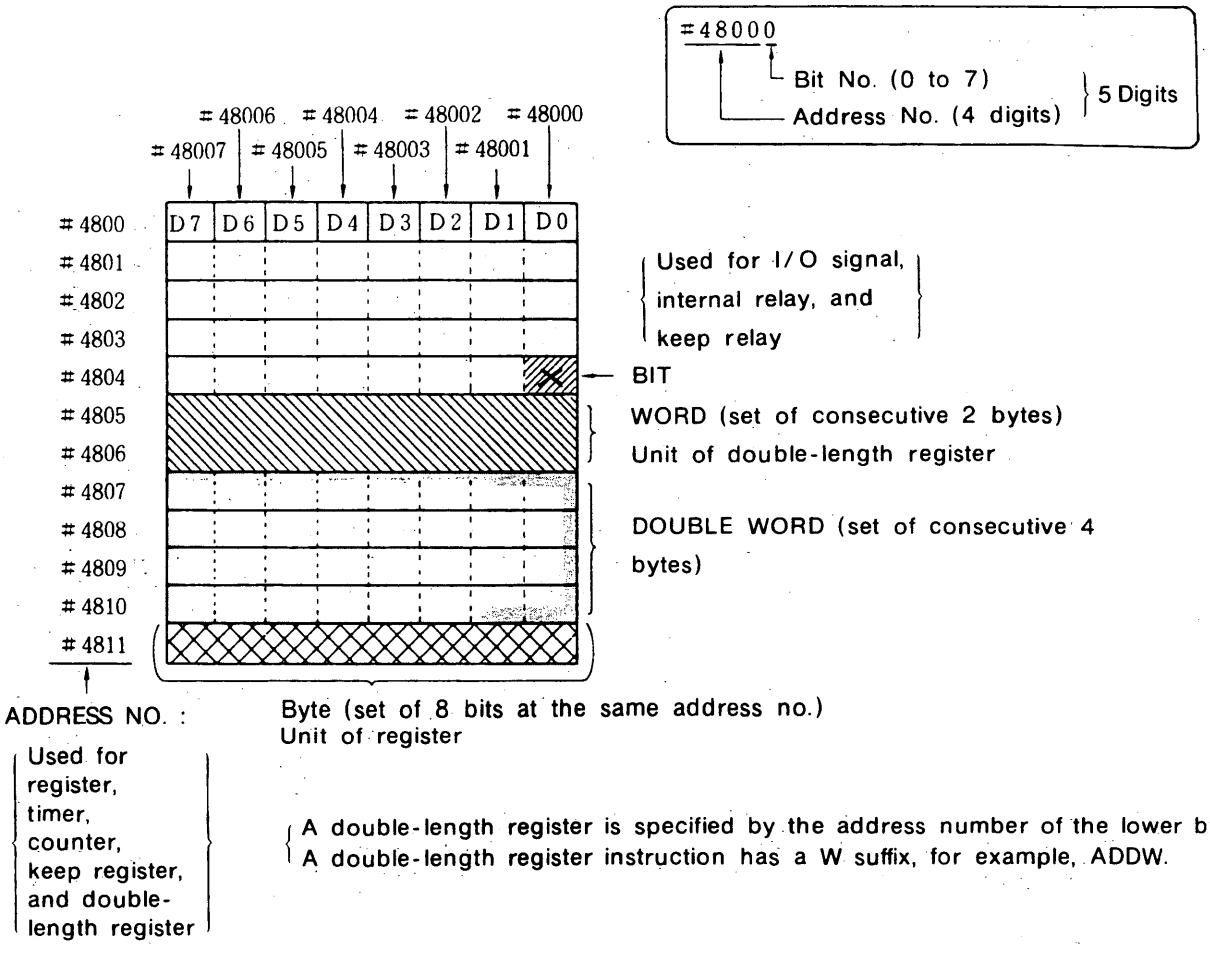
Table 2.1 Specifications of Built-In Sequencer

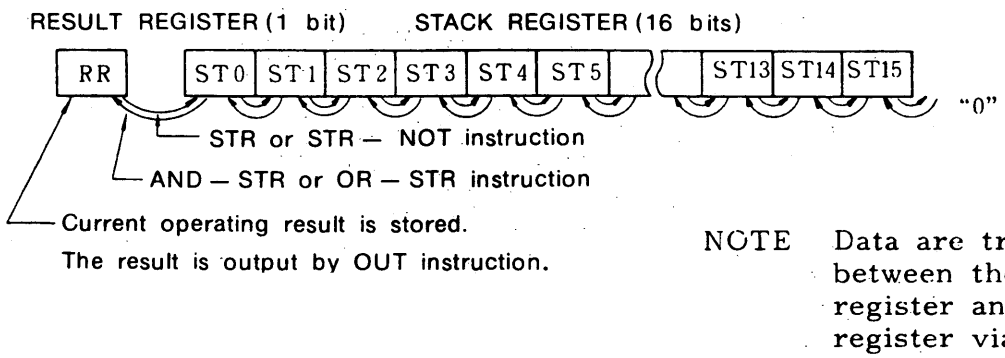
Function		Specifications
Control Method		Stored program scanning method
Scan Time		8 × N ms (N : 1 or above) automatic setting
Processing Time	Commands for Relays	1.5 μs/step
	Commands for Registers	3 to several hundred μs/step
Types of Commands	Commands for Relays	11 types
	Commands for Control	9 types
	Commands for Timer	2 types
	Commands for Counter	2 types
	Commands for Register	46 types
Sequence Capacity		30 kbytes, approx. 7000 steps
Number of I/O Points		Input : 48 points (standard 24 points, extended 24 points) Output : 48 points (standard 24 points, extended 24 points) 128 points each for I/O when M-NET interface is used.
Internal Memory	Timer	100 timers (#4600 to #4699)
	Counter	100 counters (#4700 to #4799)
	Internal Relay	Up to 9600 points (#4800 to #5999)*
	Internal Register	Up to 1200 registers (#4800 to #5999)*
	Keep-memory	Internal relays or internal register area can be set to holding hocked up memory.
Programming		Transferred to built-in sequencer after programming by personal computer.

* #4800 to #5999 can only be used as internal relays or internal registers.

3. BEFORE USING BUILT-IN SEQUENCER

3.1 TECHNICAL TERMS





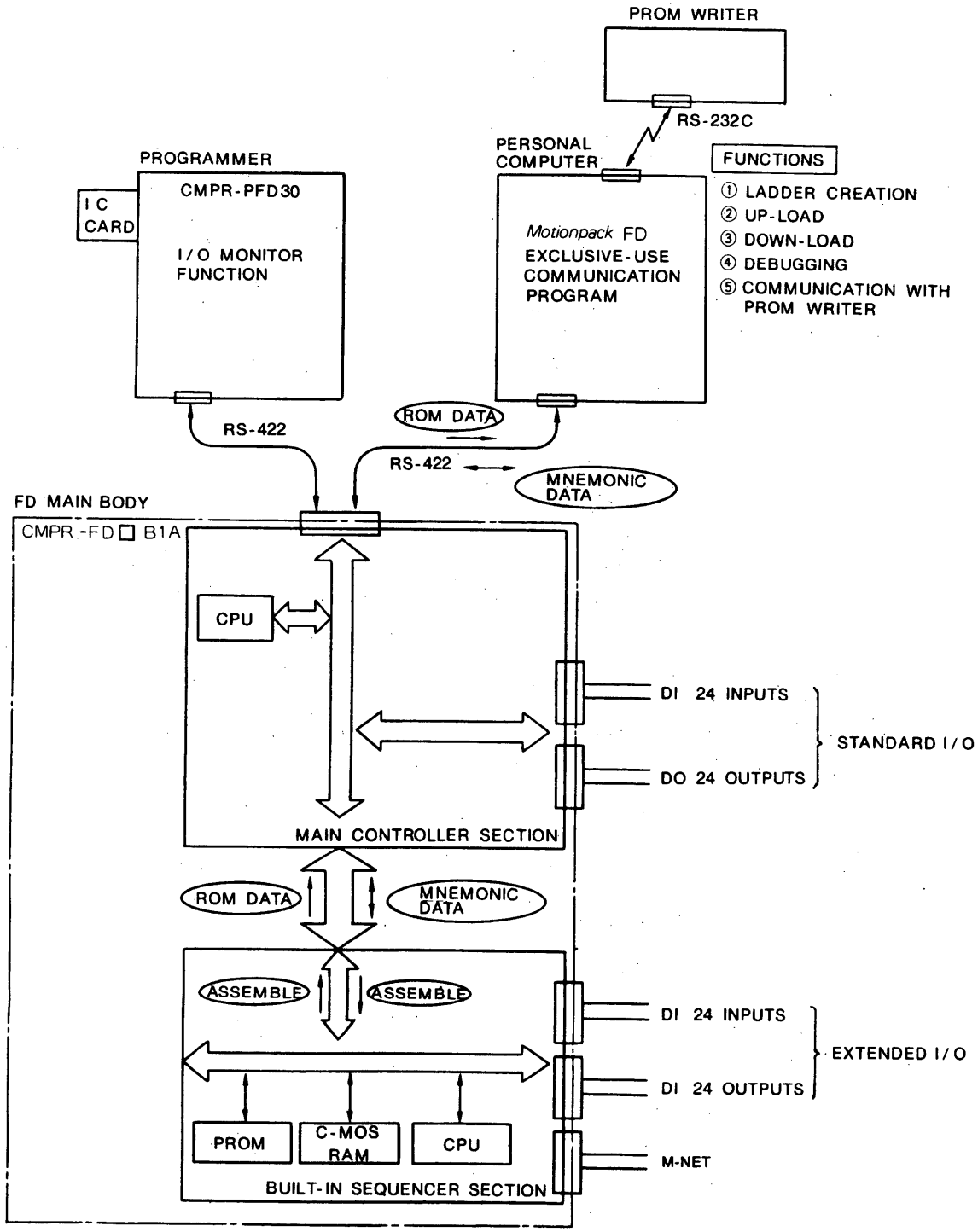
NOTE Data are transferred between the result register and the stack register via stack ST0.

CARRY Carry Register (1 bit)

1 is set in the carry register if a carry or borrow arises from addition or subtraction. The carry register cannot be incorporated in the stack register. Therefore, the status of the previous carry cannot be stored.

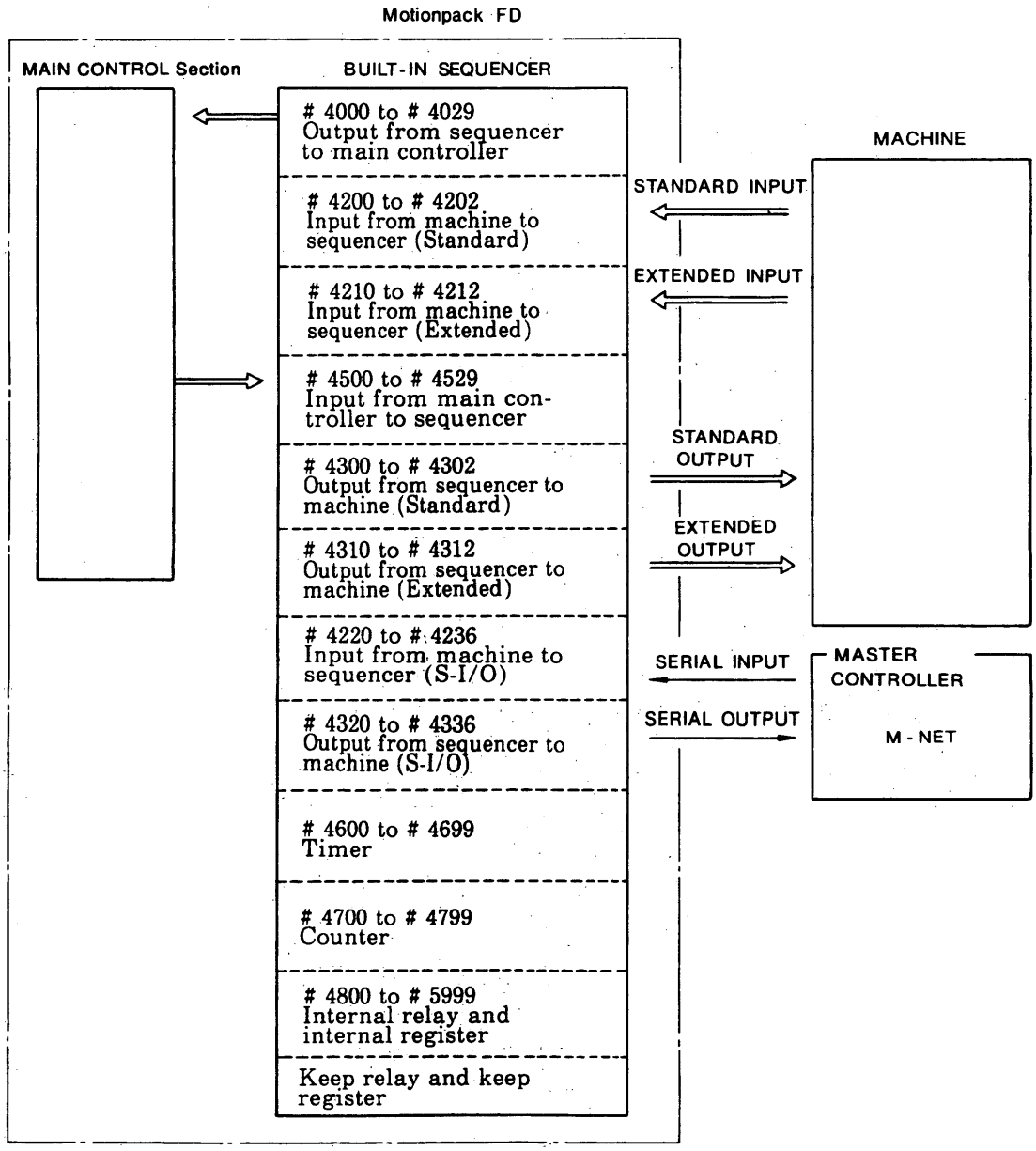
Use the carry register to prevent losing carry information.

3.2 BUILT-IN SEQUENCER SYSTEM CONFIGURATION



NOTE: Standard I/O data are taken in by the main controller section and transmitted to the built-in sequencer section.

3.3 ADDRESS MAP



Keep memory: This is a battery-backed up memory in which data are kept after power is turned off. Set timer and counter constants in this area. The range of keep memory is determined with a parameter. Set the beginning address of keep memory to parameter # 152.
(Example) When 5400 is set for parameter Pr152:
#5400 to #5999 are assigned to keep memory or keep register.

Relay, Register: The relay and register areas in the built-in sequencer are used for both relays and registers. Determine the relay and register areas upon use.

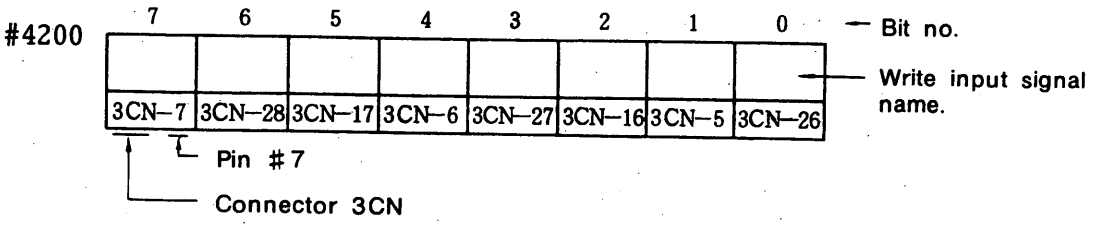
(1) Addresses for input signals from machine (#4200 to #4202: Standard input signal, #4210 to 4210: Extended input signal)

These addresses are assigned to input signals from push buttons and limit switches on the machine operation panel and distribution board. Each signal is specified by a set of an address number and a bit number (#□□□□□□). This assignment must be performed by the user.

(a) One bit of address #4200 corresponds to one input signal.

(b) The actual address number and bit number are determined by which pin of which I/O connector the input signal is connected to.

(Example)



For details, see Section 13 "I/O SIGNAL LIST".

(c) The input signal at address #4200 is represented by the following symbols:

(d) This address area can be used by a register type command using address numbers only.

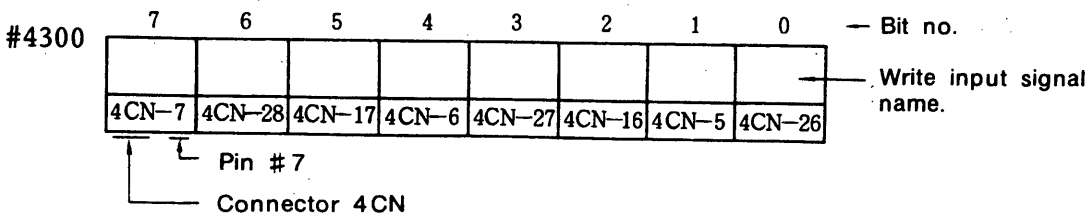
(2) Addresses for output signals to machine (#4300 to #4302: Standard output signal, #4310 to #4312: Extended output signal)

These addresses are assigned to output signals to lamps and solenoids on the machine operation panel and distribution board. Each signal is specified by a set of an address number and a bit number (# □□□□□□). This assignment must be performed by the user.

(a) One bit of address #4300 corresponds to one output signal.

(b) The actual address number and bit number are determined by which pin of which connector on the I/O board the output signal is connected to.

(Example)

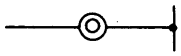


For details, see Section 13 "I/O SIGNAL LIST".

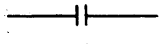
3.3 ADDRESS MAP (Cont'd)

(c) The output signal at address #4300 is represented by the following symbols:

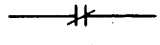
Contact



#43 □□□ Coil



#43 □□□ (NO Contact)



#43 □□□ (NC Contact)

(d) This address area can be used by a register instruction by using address numbers only.

(3) Addresses for input signals from main controller (#4500 to #4529)

Seen from the main controller, these addresses are for output signals from the main controller to the built-in sequencer.

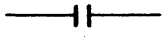
(Example) Address No. + bit No. assigned to M-BCD signal

(a) One bit of address #4500 corresponds to one input signal.
For details, see Section 13 "I/O SIGNAL LIST".

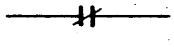
(Example)

	7	6	5	4	3	2	1	0	
#4500	EP ALM	G34	OFM	OFR	INCD	STL	SALM	MRDY	Bit no.

(b) The input signal at address #4500 is represented by the following symbols:



#45 □□□ (NO Contact)



#45 □□□ (NC Contact)

(c) This address area can be used by a register instruction if by using address numbers only.

(4) Addresses for output signals to main controller (#4000 to #4029)

Seen from the main controller, these addresses are for input signals from the built-in sequencer to the main controller.

(Example) Address No. + bit No. assigned to EDIT signal

(a) One bit of address #4000 corresponds to one output signal.

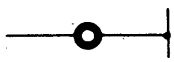
(Example)

	7	6	5	4	3	2	1	0	
#4000	ZRN	−JS	+JS	JSPD	HANDLE	JOG	PLAY	EDIT	Bit no.

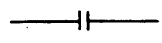
For details, see Section 13 "I/O SIGNAL LIST".

(b) The output signal at address #4000 is represented by the following symbols:

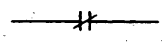
Contact



#40 □□□ Coil



#40 □□□ (NO Contact)



#40 □□□ (NC Contact)

(c) This address area can be used by a register instruction by using address numbers only.

(5) Addresses for timers (#4600 to #4699)

These addresses are assigned to timers. The addresses are used in timer instructions.

(a) One address number corresponds to one timer.

(b) Available timers are listed in the following table:

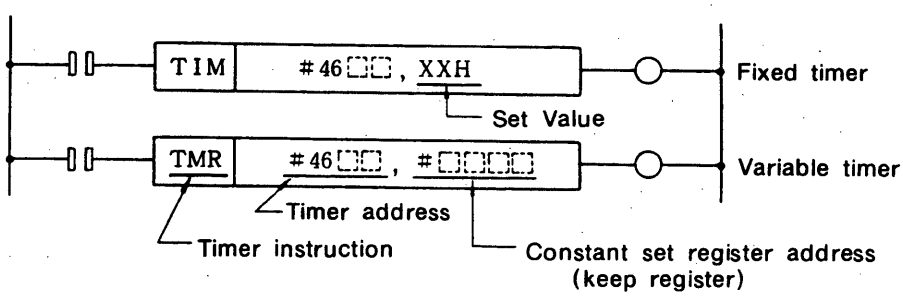
Table 3.1 Time Unit and Number of Timers

Address No.	Number of Timers	Time Unit
#4600 to #4619	20	8 ms
#4620 to #4649	30	50 ms
#4650 to #4679	30	100 ms
#4680 to #4689	10	1 s
#4690 to #4699	10	1 min

Value 0 to 255 can be set for each timer.

(c) An example of timer symbols is shown in the following:

(Example)



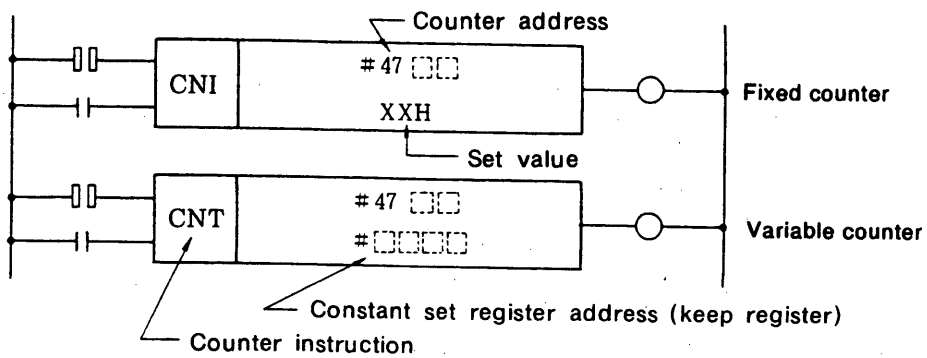
(6) Addresses for counters (#4700 to #4799)

These addresses are assigned to counters. The addresses are used in counter instructions.

(a) One address number corresponds to one counter.

(b) An example of counter symbols is shown in the following:

(Example)



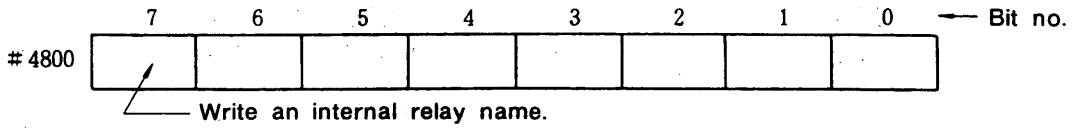
3.3 ADDRESS MAP (Cont'd)

(7) Addresses for internal relays (#4800 to #5999)

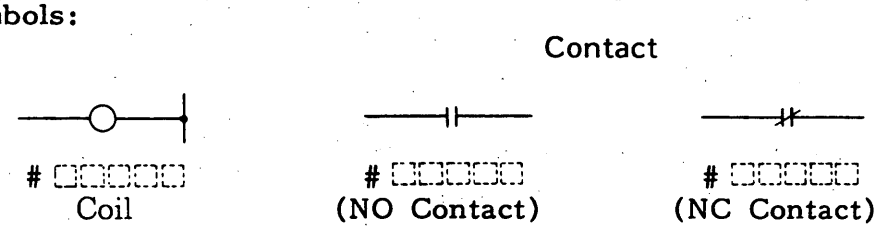
These addresses are assigned to internal relays for the built-in sequencer for generating logic programs. Each relay is assigned to a set of an address number and a bit number.

(a) One bit of address #4800 corresponds to one internal relay.

(Example)



(b) An internal relay and the contact are represented by the following symbols:



(c) This address area can be used by a register instruction by using address numbers only.

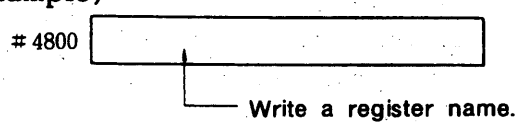
(d) An internal relay assigned in the keep memory area works as a keep relay.

(8) Addresses for internal registers (#4800 to #5999)

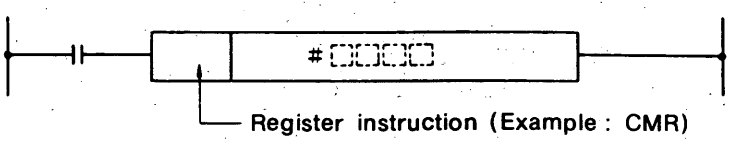
These address numbers are assigned to internal 1 byte (8 bit) register.

(a) One address corresponds to one 1 byte register.

(Example)



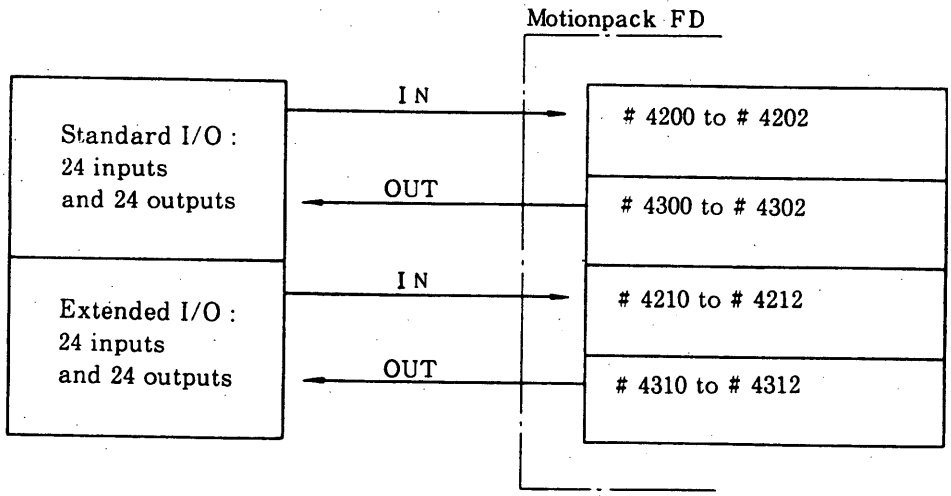
(b) The address number itself is the representation symbol of a register.



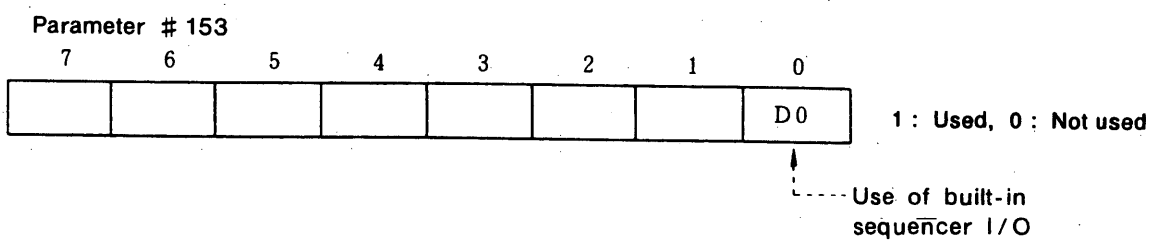
(c) A register assigned in the keep memory works as a keep register.

(d) A double-length register is specified by the address number of the lower byte.

3.4 RELATIONS BETWEEN I/O SIGNALS AND ADDRESS



NOTE Selection of built-in sequencer is set in parameter Pr153. Pr153 (D0) must be set to 1 to use built-in sequencer.
Otherwise, built-in sequencer is impossible to cannot be use.



4. BUILT-IN SEQUENCER INSTRUCTIONS

Table 4.1 Relay Instructions

	Relay Instructions		Function
①	LD	#××××××	Load
②	LD-NOT	#××××××	NOT
③	AND	#××××××	AND
④	AND-NOT	#××××××	AND-NOT
⑤	OR	#××××××	OR
⑥	OR-NOT	#××××××	OR-NOT
⑦	STR	#××××××	Store result and load
⑧	STR-NOT	#××××××	Store result and load NOT
⑨	AND-STR		AND with stored result
⑩	OR-STR		OR with stored result
⑪	OUT	#××××××	Output result to relay

Table 4.2 Control Instructions

	Control Instructions	Function
①	NOP	No operation
②	MCR	Master control relay
③	END	End of master control relay
④	RET	End of sequence
⑤	RTI	Conditional end of sequence
⑥	SET	Set result
⑦	RST	Clear result
⑧	STC	Set carry
⑨	CLC	Clear carry

Table 4.3 Timer and Counter Instructions

	Timer and Counter Instructions		Function
①	TIM	#××××, ××H	Fixed timer
②	TMR	#××××, #△△△△	Variable timer
③	CNI	#××××, ××H	Fixed counter
④	CNT	#××××, #△△△△	Variable counter

Table 4.4 Register Instructions

	Register Instructions		Function
①	CMR	#××××	Invert byte register
②	CMRW	#××××	Invert word register
③	ADI	#××××, ××H	Add binary constant
④	ADD	#××××, #△△△△	Add byte register
⑤	ADC	#××××, #△△△△	Add byte register with carry
⑥	ADDW	#××××, #△△△△	Add word register
⑦	ADCW	#××××, #△△△△	Add word register with carry
⑧	DAD	#××××, #△△△△	Add 2-digit BCD
⑨	DADS	#××××, #△△△△, #○○○○	Add any-even-number-of-digit BCD
⑩	SBI	#××××, ××H	Subtract binary constant
⑪	SUB	#××××, #△△△△	Subtract byte register
⑫	SBB	#××××, #△△△△	Subtract byte register with carry
⑬	SUBW	#××××, #△△△△	Subtract word register
⑭	SBBW	#××××, #△△△△	Subtract word register with carry

Table 4.4 Register Instructions (Cont'd)

	Register Instructions		Function
⑮	DSB	#××××, #△△△△	Subtract 2-digit BCD
⑯	DSBS	#××××, #△△△△, #○○○○	Subtract any-even-number-of-digits, BCD
⑰	MULW	#××××, #△△△△	Word * byte → word
⑱	MULD	#××××, #△△△△	Word * word → double word
⑲	DIVW	#××××, #△△△△	Word/byte → word
⑳	ANI	#××××, ××H	Binary constant AND byte register
㉑	ANR	#××××, #△△△△	Byte register AND byte register
㉒	ORI	#××××, ××H	Binary constant OR byte register
㉓	ORR	#××××, #△△△△	Byte register OR byte register
㉔	SAL	#××××, ××H	Byte register arithmetic left shift
㉕	SAR	#××××, ××H	Byte register arithmetic right shift
㉖	SALW	#××××, ××H	Word register arithmetic left shift
㉗	SARW	#××××, ××H	Word register arithmetic right shift
㉘	MVI	#××××, ××H	Byte register, constant transfer
㉙	MVIW	#××××, ××××H	Word register, constant transfer
㉚	MOV	#××××, #△△△△	Data transfer between byte registers
㉛	MOVS	#××××, #△△△△, #○○○○	Data transfer between two sets of registers
㉜	DST	#××××, #△△△△, #××H	Constant And byte register, and result transfer
㉝	DSTW	#××××, #△△△△, #××××H	Constant AND word register, and result transfer
㉞	DIN	#××××, #△△△△, #××H	Byte register data selection
㉟	DEC	#××××, ××H	Unconditional data match detection
㊱	COI	#××××, ××H	Data match detection
㊲	COR	#××××, #△△△△	Byte register contents match detection
㊳	CORW	#××××, #△△△△	Word register contents match detection
㊴	CMP	#××××, ××H	Unconditional data comparison
㊵	CPI	#××××, ××H	Data comparison
㊶	CPR	#××××, #△△△△	Byte register contents comparison
㊷	CPRW	#××××, #△△△△	Word register contents comparison
㊸	BCD4	#××××, #△△△△	Conversion from binary to 4-digit BCD
㊹	BCD8	#××××, #△△△△	Conversion from binary to 8-digit BCD
㊺	BIN4	#××××, #△△△△	Conversion from 4-digit BCD to binary
㊻	BIN8	#××××, #△△△△	Conversion from 8-digit BCD to binary

Table 4.5 Special Instruction

	Special Instruction	Function
①	POPR	Operation error result set
②	ERROR	Download error information

5. EXPLANATIONS OF BUILT-IN SEQUENCER INSTRUCTIONS

5.1 RELAY INSTRUCTIONS

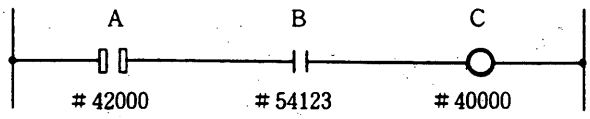
(1) LD (load) {RR¦}

(a) Format: LD #XXXXX

Relay Example #42000
#54123

(b) This instruction reads status (1 or 0) of a contact and sets it to RR.

(c) This instruction is used on normally open (NO) contact.




```
LD   #42000
AND  #54123
OUT  #40000
```

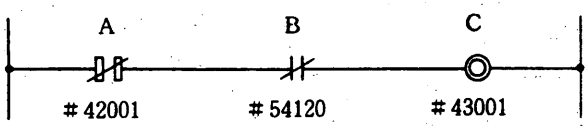
(2) LD-NOT (load not) {RR¦}

(a) Format: LD NOT #XXXXX

Relay Example #42001
#54120

(b) This instruction reads status (1 or 0) of an inverted contact and sets it to RR.

(c) This instruction is used on normally closed (NC) contact.



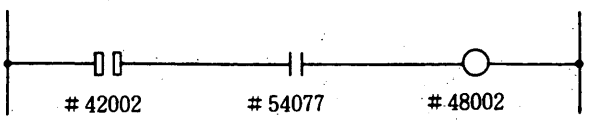

```
LD-NOT   #42001
AND-NOT  #54120
OUT      #43001
```

(3) AND {RR¦}

(a) Format: AND #XXXXX

Relay

(b) This instruction performs AND operation with the contents of RR and a contact (NO contact), then stores the result (1 or 0) to RR.



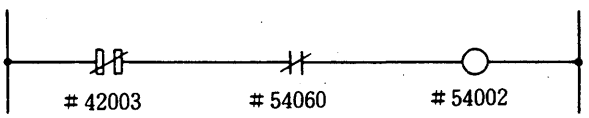

```
LD   #42002
AND  #54077
OUT  #48002
```

(4) AND-NOT {RR¦}

(a) Format: AND-NOT #XXXXX

Relay

(b) This instruction performs AND operation with the contents of RR and an inverted contact (NC contact), then stores the result to RR.

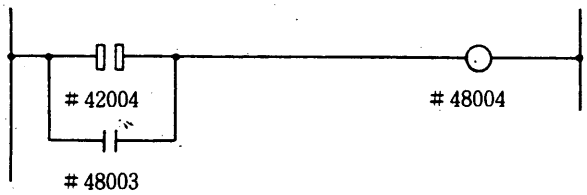



```
LD       #42003
AND-NOT  #54060
OUT      #54002
```

(5) OR {RR|}

(a) Format: OR #XXXXX
Relay

(b) This instruction performs OR operation with the contents of RR and a contact (NO contact), then stores the result to RR.

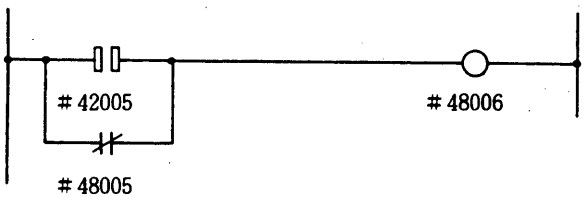



```
LD   #42004
OR   #48003
OUT  #48004
```

(6) OR-NOT {RR|}

(a) Format: OR-NOT #XXXXX
Relay

(b) This instruction performs OR operation with the contents of RR and an inverted contact (NC contact), then stores the result to RR.

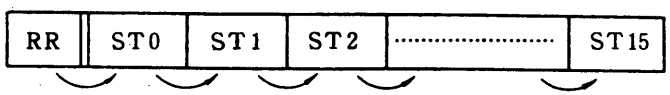



```
LD       #42005
OR-NOT   #48005
OUT      #48006
```

(7) STR (store) {RR|}

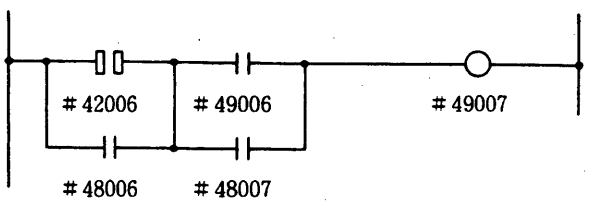
(a) Format: STR #XXXXX
Relay

(b) This instruction stores the contents of RR then executes the LD instruction.



Up to 16 stacks can be used.

(c) This instruction is used at NO contact.



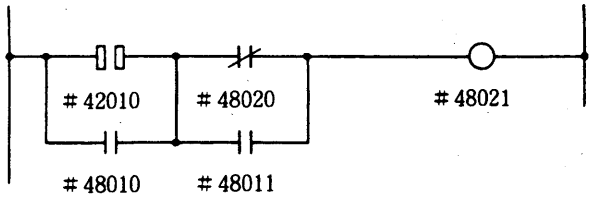

```
LD       #42006
OR       #48006
STR      #49006
OR       #48007
AND-STR
OUT      #49007
```

(8) STR-NOT {RR|}

(a) Format: STR-NOT #XXXXX
Relay

5.1 RELAY INSTRUCTIONS (Cont'd)

(b) This instruction stores the contents of RR then executes the LD-NOT instruction.

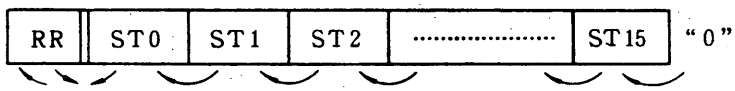



```
LD          #42010
OR-NOT      #48010
STR-NOT     #48020
OR          #48011
AND-STR
OUT         #48021
```

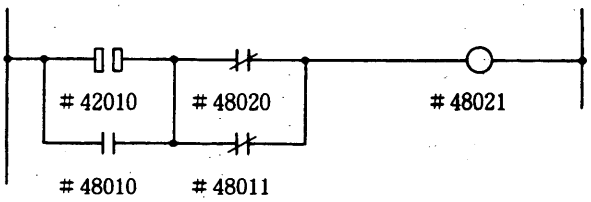
(9) AND-STR {RR↕}

(a) Format: AND-STR

(b) This instruction performs AND instruction with the contents of RR and stack STO, then stores the result to RR. Contents of the remaining stacks are shifted to the left as shown in the following:



AND

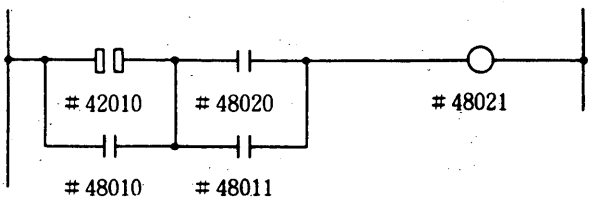



```
LD          #42010
OR          #48010
STR-NOT     #48020
OR-NOT      #48011
AND-STR
OUT         #48021
```

(10) OR-STR {RR↕}

(a) Format: OR-STR

(b) This instruction performs OR instruction with the contents of RR and stack STO, then stores the result to RR.

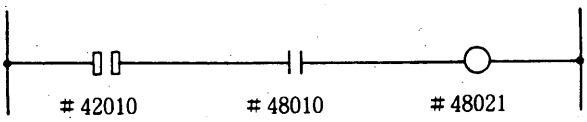



```
LD          #42010
AND         #48020
STR         #48010
AND         #48011
OR-STR
OUT         #48021
```

(11) OUT {RR↕}

(a) Format: OUT #XXXXX
Relay

(b) This instruction writes operation results to relays.




```
LD  #42010
AND #48010
OUT #48021
```

5.2 CONTROL INSTRUCTIONS

(1) NOP (No operation) {RR-}

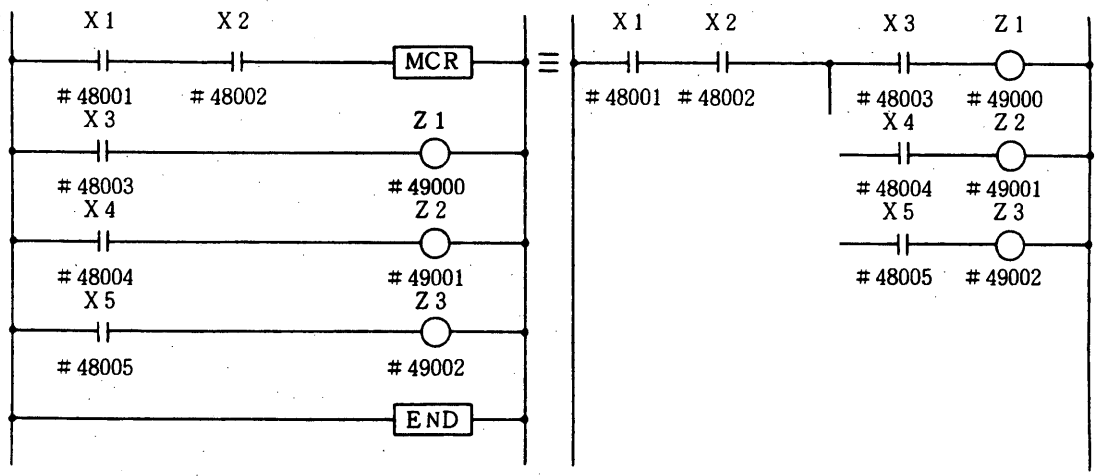
(a) Format: NOP

(b) This instruction directs the computer to only proceed to the next step. This instruction does not affect the contents of RR.

(2) MCR (Master control) {RR-}

(a) Format: MCR

(b) If RR before MCR is 1, this instruction releases the sequence ladder between MCR and EWD is executed.
If RR before MCR is 0, this instruction writes 0 to all relays before END.




```
LD    #48001
AND   #48002
MCR
LD    #48003
OUT   #49000
LD    #48004
OUT   #49001
LD    #48005
OUT   #49002
END
```

If contacts X1 and X2 are off, 0 is output to internal relays, Z1, Z2, and Z3.

(c) Between MCR and END, other MCR and END can be added. (The maximum nesting level is seven.)

(d) If a timer or counter instruction is included in MCR, the timer or counter is cleared when MCR is off.

(e) An auto retention circuit can be incorporated in MCR, but the output from the auto retention circuit automatically becomes off (0) when MCR is off.

(3) END (Master control end) {RR-}

(a) Format: END

(b) This instruction indicates the end of master control.

5.2 CONTROL INSTRUCTIONS (Cont'd)

(4) RET (Return) {RR-}

(a) Format: RET

(b) This instruction indicates the end of the sequence program.

(5) RTI (Return indirect) {RR-}

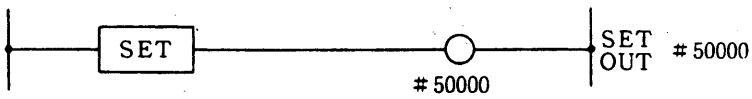
(a) Format: RTI

(b) If RR before the RTI instruction is 1, the computer executes the RET instruction. If RR before the RTI instruction is 0, the next step is executed.

(6) SET {RR-}

(a) Format: SET

(b) This instruction sets RR to 1.



(7) RST (Reset) {RR = 0}

(a) Format: RST

(b) This instruction resets RR to 0.

(8) STC (Set carry) {RR-}

(a) Format: STC

(b) This instruction sets the carry to 1.

(9) CLC (Clear carry) {RR-}

(a) Format: CLC

(b) This instruction clears carry to 0.

5.3 TIMER AND COUNTER INSTRUCTIONS

(1) TIM (Fixed timer) {RR time up = 1}

(a) Format: TIM #XXXX, XXH

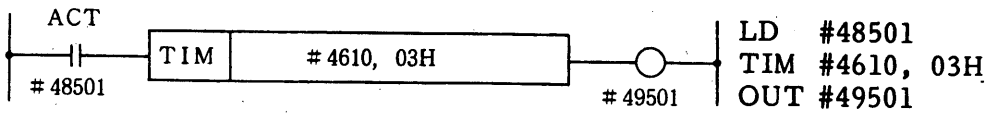
- #XXXX: Timer set address (#4600 to #4699)
- XXH: Timer set time

(b) If ACT before the TIM instruction is 1, the timer starts count-up and when the set time has elapsed, the result RR is set to 1. The RR is 0 until the time elapses. If ACT before the TIM instruction is 0, the timer is reset and the result RR is cleared to 0.

(c) Time can be set in the range of 0 to 255 in decimal number. The instruction must be written in hexadecimals.

(d) There are five types of timers as listed in the table:

Address	Type	Qty
#4600 to #4619	Set value 1 equals to 8 ms.	20
#4620 to #4659	Set value 1 equals to 50 ms.	30
#4660 to #4679	Set value 1 equals to 100 ms.	30
#4680 to #4689	Set value 1 equals to 1 s.	10
#4690 to #4699	Set value 1 equals to 1 min.	10



NOTE 1. Do not use duplicated address for fixed and variable timers.
2. Time error is the time set unit of the timer. Use a small-scale timer when precision is required.

(2) TMR (Variable timer) {RR time up = 1}

(a) Format: TMR #XXXX, #XXXX

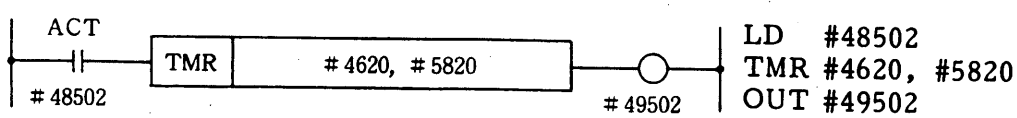
- #XXXX: Timer address (#4600 to #4699)
- #XXXX: Timer constant set register address

(b) If ACT before the TMR instruction is 1, the timer starts count-up and when the set time has elapsed, the result RR is set to 1. The RR is 0 until the time elapses. If ACT before the TMR instruction is 0, the timer is reset and the result RR is cleared to 0.

(c) Time can be set by a decimal number from 0 to 255.

(d) Time address must be set in the keep memory area.

(e) Similar to the TIM instruction, five types of timers are available. The duplicated timer address cannot be shared by the TIM and TMR instructions.



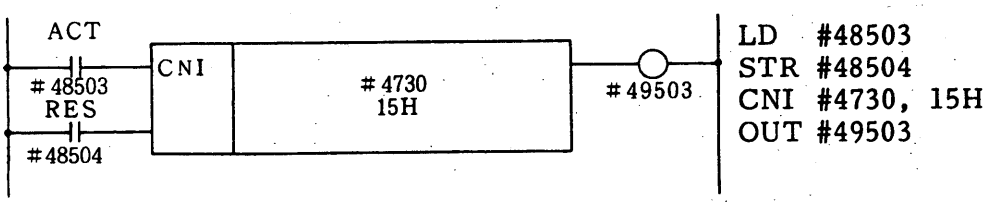
5.3 TIMER AND COUNTER INSTRUCTIONS (Cont'd)

(3) CNI (Fixed down counter) {RR count up = 1}

(a) Format:
```
LD  #XXXXX ← Counter input
STR #XXXXX ← Reset input
CNI #XXXX, XXH
          |      |
          |      └── Counter preset value
          └── Counter address (#4700 to #4799)
```

(b) If the reset input (RES) is 1, the counter is preset and RR is reset to 0. Count-down is started when the counter input (ACT) rises from 0 to 1. When the count becomes to 0, RR is set to 1. After that, if ACT changes from 1 to 0, RR is reset to 0.
After the count is reduced to 0, if ACT changes from 0 to 1 before RES goes to 1, the counter is preset from 0 to 255.

(c) Counter preset value range is from 0 to 255 (decimals). The instruction must be written in hexadecimals.




```
LD  #48503
STR #48504
CNI #4730, 15H
OUT #49503
```

(4) CNT (Variable down counter) {RR count up = 1}

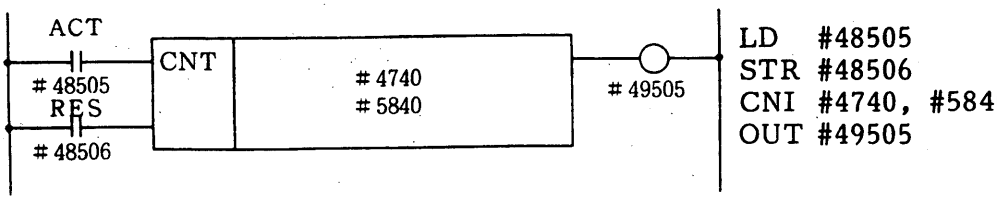
(a) Format:
```
LD  #XXXXX ← Counter input
STR #XXXXX ← Reset input
CNT #XXXX, XXXX
          |      |
          |      └── Counter constant set register address
          └── Counter address (#4700 to #4799)
```

(b) If the reset input (RES) is 1, the counter is preset and RR is reset to 0. Count-down is started when the counter input (ACT) rises from 0 to 1. When the count becomes to 0, RR is set to 1. After that, if ACT changes from 1 to 0, RR is reset to 0.
After the count is reduced to 0, if ACT changes from 0 to 1 before RES goes to 1, the counter is preset from 0 to 255.

(c) Counter preset value range is from 0 to 255 (decimals).

(d) Counter constant set register address must be selected in the keep memory area.

(e) The duplicated counter address cannot be shared by the CNI and CNT instructions.




```
LD  #48505
STR #48506
CNI #4740, #5840
OUT #49505
```

5.4 REGISTER INSTRUCTIONS

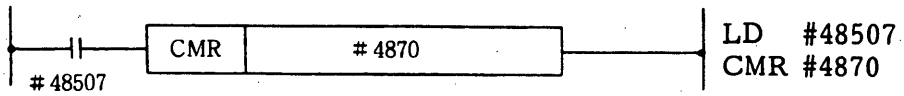
(1) CMR (Complement register) {RR-}

(a) Format: CMR #XXXX

Register address

(b) If RR immediately before CMR is 1, the contents of the register are inverted. If RR immediately before CMR is 0, the CMR instruction is not executed.

(c) A contact must be placed before CMR.



(2) CMRW (Complement word register) {RR-}

This instruction is the same as CMR except that the contents of a double-length register are to be inverted.

(3) ADI (Add immediate) {RR-}

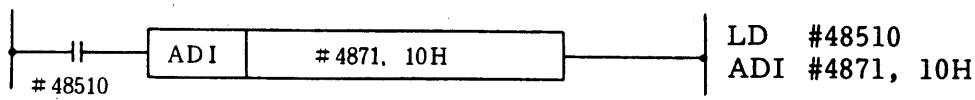
(a) Format: ADI #XXXX, XXH

Numeric value (hexadecimals)

Register address

(b) If RR immediately before ADI is 1, the numeric value is added to the contents of the register, then the result is stored in the register. This instruction does not affect the contents of RR or CARRY. If RR immediately before ADI is 0, the ADI instruction is not executed.

(c) A contact must be placed before ADI.



(4) ADD (Add register) {RR-}

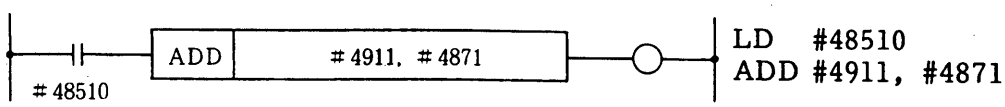
(a) Format: ADD #XXXX, #XXXX

Augend register (R2)

Addend register (R1)

(b) If RR immediately before ADD is 1, the contents of register R1 are added to that of register R2, then the result is stored in register R2. This instruction does not affect the contents of register R1, RR or CARRY. If RR immediately before ADD is 0, the ADD instruction is not executed.

(c) A contact must be placed before ADD.



5.4 REGISTER INSTRUCTIONS (Cont'd)

(5) ADC (Add register with carry) {RR↕, CARRY↕}

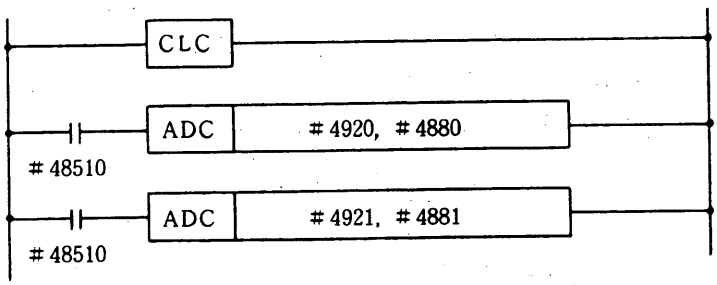
(a) Format: ADC #XXXX, #XXXX

Augend register (R2)

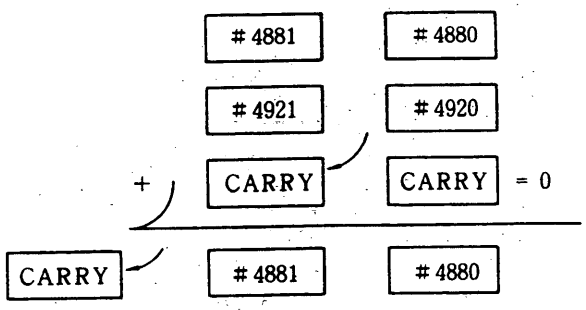
Addend register (R1)

(b) If RR immediately before ADC is 1, the contents of register R1 and CARRY are added to that of register R2, then the result is stored in register R2. If a carry arises from the addition, RR and CARRY are set to 1. This instruction does not affect the contents of register R1. If RR immediately before ADC is 0, the ADC instruction is not executed.

(c) A contact must be placed before ADC.




```
CLC
LD   #48510
ADC #4920, #4880
LD   #48510
ADC #4921, #4881
```



(6) ADDW (Add word register) {RR-}

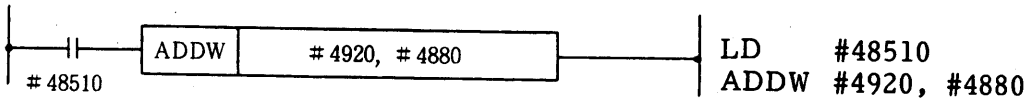
(a) Format: ADDW #XXXX, #XXXX

Lower byte of double-length register (WR2)

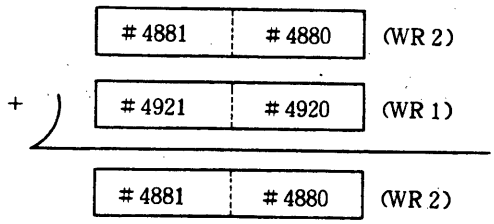
Lower byte of double-length register (WR1)

(b) If RR immediately before ADDW is 1, the contents of double-length register WR1 are added to that of double-length register WR2, then the result is stored in double-length register WR2.
This instruction does not affect the contents of double-length register WR1, RR or CARRY.
If RR immediately before ADDW is 0, the ADDW instruction is not executed.

(c) A contact must be placed before ADDW.




```
LD     #48510
ADDW   #4920, #4880
```



(7) ADCW (Add word register with carry) {RR↓, CARRY↓}

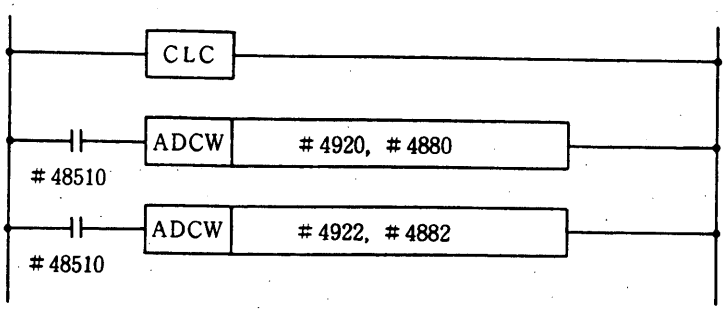
(a) Format: ADCW #XXXX, #XXXX

Lower byte of double-length register (WR2)

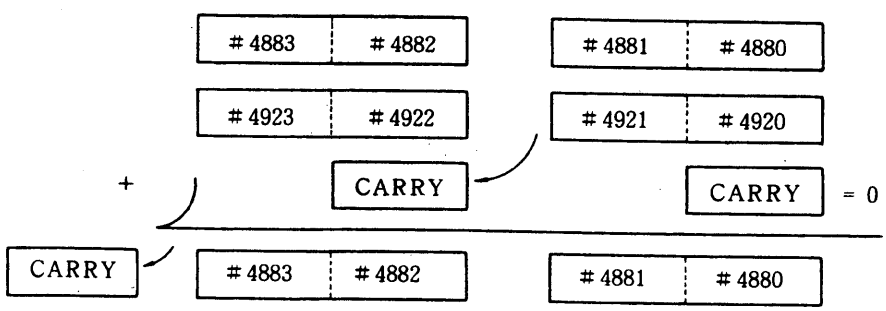
Lower byte of double-length register (WR1)

(b) If RR immediately before ADCW is 1, the contents of double-length register WR1 and CARRY are added to that of double-length register WR2, then the result is stored in double-length register WR2. If a carry arises from the addition, RR and CARRY are set to 1. This instruction does not affect the contents of double-length register WR1. If RR immediately before ADCW is 0, the ADCW instruction is not executed.

(c) A contact must be placed before ADCW.




```
CLC
LD     #48510
ADCW   #4920, #4880
LD     #48510
ADCW   #4922, #4882
```



5.4 REGISTER INSTRUCTIONS (Cont'd)

(8) DAD (Decimal add register) {RR↕, CARRY↕}

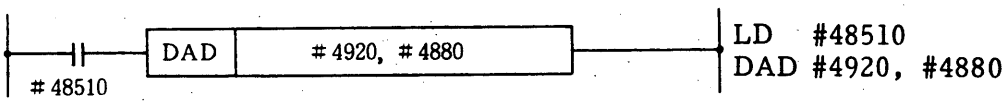
(a) Format: DAD #XXXX, #XXXX

Augend register (R2)

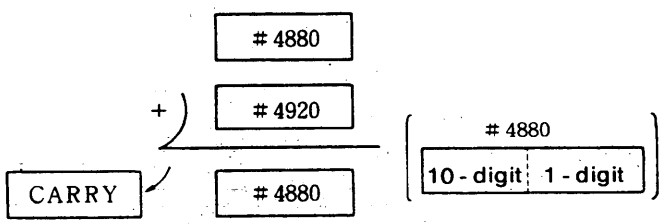
Addend register (R1)

(b) If RR immediately before DAD is 1, the contents of register R1 (decimals) are added to that of register R2 (decimals), then the result is stored to register R2.
If a carry arises from the addition, RR and CARRY are set to 1.
If RR immediately before DAD is 0, the DAD instruction is not executed.

(c) A contact must be placed before DAD.



Decimal 2-digit Operation



(9) DADS (Decimal add register string) {RR↕, CARRY↕}

(a) Format: DADS #XXXX, #XXXX #XXXX

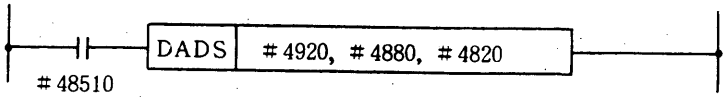
Operation repetition number register (R3)

Augend first register (R2)

Addend first register (R1)

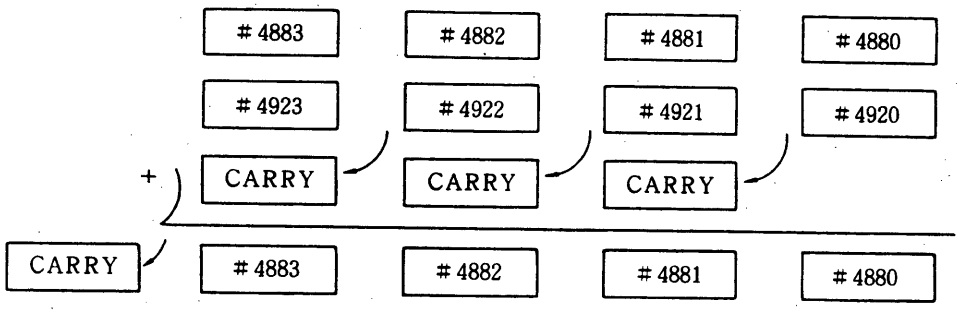
(b) If RR immediately before DADS is 1, the contents of a specified number of consecutive registers beginning with register R1 are added to the contents of the corresponding number of registers beginning with register R2. The number of additions is set in register R3.
The additions are performed on the decimal basis.
If a carry arises in the last register, RR and CARRY are set to 1.
If RR immediately before DADS is 0, the DADS instruction is not executed.

(c) A contact must be placed before DADS.




```
LD    #48510
DADS  #4920, #4880, #4820
```

If a content of #4820 is "4" (Decimal 8-digit Operation)



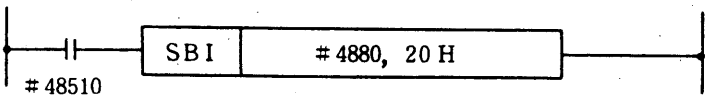
(10) SBI (Subtract immediate) {RR-}

(a) Format: SBI #XXXX, XXH

Numeric value (hexadecimals)
Register address

(b) If RR immediately before SBI is 1, the numeric value is subtracted from the contents of the register, then the result is stored in the register. This instruction does not affect the contents of RR or CARRY. If RR immediately before SBI is 0, the SBI instruction is not executed.

(c) A contact must be placed before SBI.




```
LD   #48510
SBI  #4880, 20H
```

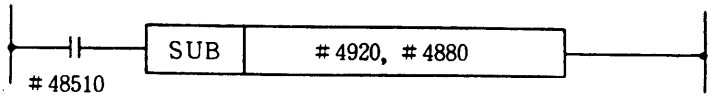
(11) SUB (Subtract register) {RR-}

(a) Format: SUB #XXXX, #XXXX

Minuend register (R2)
Subtrahend register (R1)

(b) If RR immediately before SUB is 1, the contents of register R1 are subtracted from that of register R2, then the result is stored in register R2.
This instruction does not affect the contents of RR or CARRY.
If RR immediately before SUB is 0, the SUB instruction is disregarded.

(c) A contact must be placed before SUB.




```
LD   #48510
SUB  #4920, #4880
```

5.4 REGISTER INSTRUCTIONS (Cont'd)

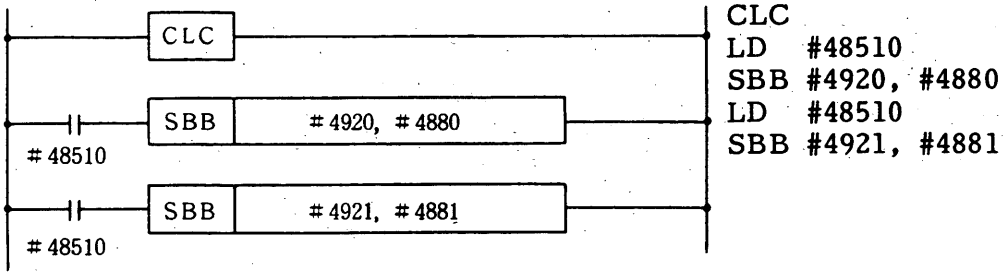
(12) SBB (Subtract register with borrow) {RR↓, CARRY↓}

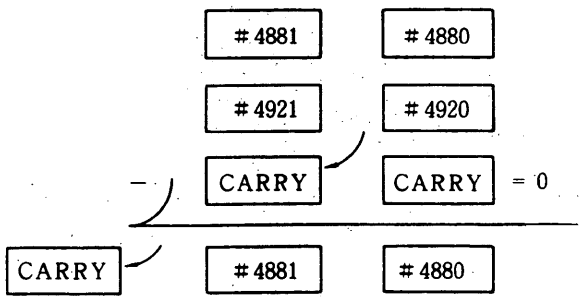
(a) Format: SBB #XXXX, #XXXX

#XXXX: Subtrahend register (R1)
#XXXX: Minuend register (R2)

(b) If RR immediately before SBB is 1, the contents of register R1 and CARRY are subtracted from those of register R2, then the result is stored in register R2. If a borrow arises from the subtraction, RR and CARRY are set to 1. This instruction does not affect the contents of register R1. If RR immediately before SBB is 0, the SBB instruction is not executed.

(c) A contact must be placed before SBB.





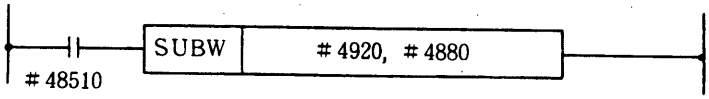
(13) SUBW (Subtract word register) {RR-}

(a) Format: SUBW #XXXX, #XXXX

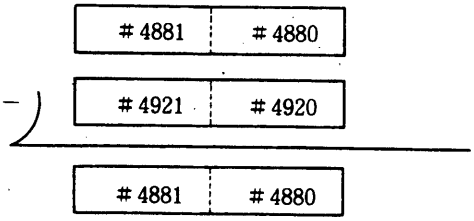
#XXXX: Lower byte of double-length register (WR2)
#XXXX: Lower byte of double-length register (WR1)

(b) If RR immediately before SUBW is 1, the contents of double-length register WR1 are subtracted from those of double-length register WR2, then the result is stored in double-length register WR2. This instruction does not affect the contents of RR or CARRY. If RR immediately before SUBW is 0, the SUBW instruction is not executed.

(c) A contact must be placed before SUBW.




```
LD   #48510
SUB #4920, #4880
```



(14) SBBW (Subtract word register with borrow) {RR↓, CARRY↓}

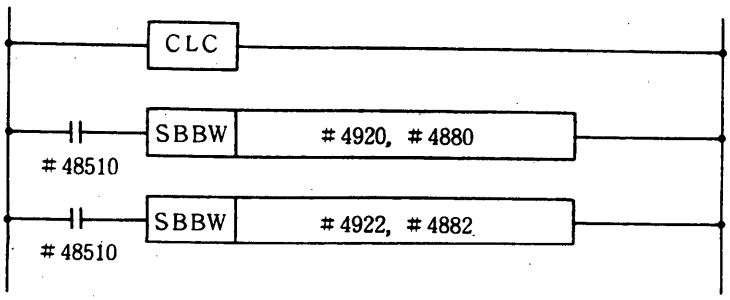
(a) Format: SBBW #XXXX, #XXXX

Lower byte of double-length register (WR2)

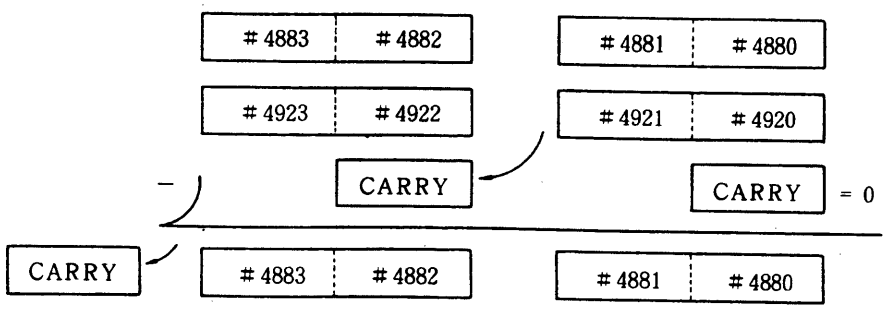
Lower byte of double-length register (WR1)

(b) If RR immediately before SBBW is 1, the contents of double-length register WR1 and CARRY are subtracted from those of double-length register WR2, then the result is stored in double-length register WR2.
If a borrow arises from the subtraction, RR and CARRY are set to 1.
If RR immediately before SBBW is 0, the SBBW instruction is not executed.

(c) A contact must be placed before SBBW.




```
CLC
LD    #48510
SBBW  #4920, #4880
LD    #48510
SBBW  #4922, #4882
```



5.4 REGISTER INSTRUCTIONS (Cont'd)

(15) DSB (Decimal subtract) {RR, CARRY}

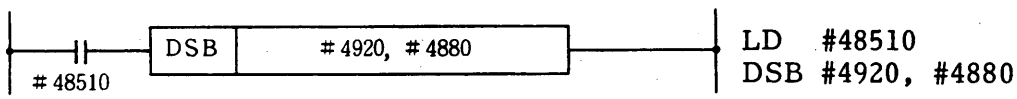
(a) Format: DSB #XXXX, #XXXX

Minuend register (R2)

Subtrahend register (R1)

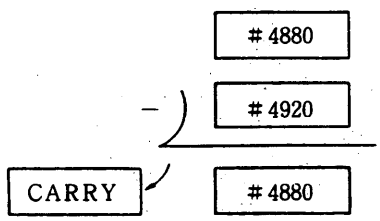
(b) If RR immediately before DSB is 1, the contents of register R1 (decimals) are subtracted from those of register R2 (decimals), then the result is stored in register R2.
If a borrow arises from the subtraction, RR and CARRY are set to 1.
If RR immediately before DSB is 0, the DSB instruction is not executed.

(c) A contact must be placed before DSB.



LD #48510
DSB #4920, #4880

Decimal 2-digit Operation



(16) DSBS (Decimal subtract string) {RR, CARRY}

(a) Format: DSBS #XXXX, #XXXX, #XXXX

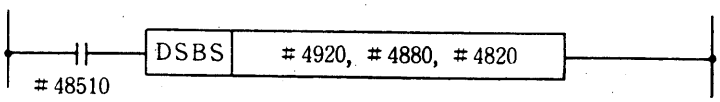
Operation repetition number register (R3)

Minuend register first address (R2)

Subtrahend register first address (R1)

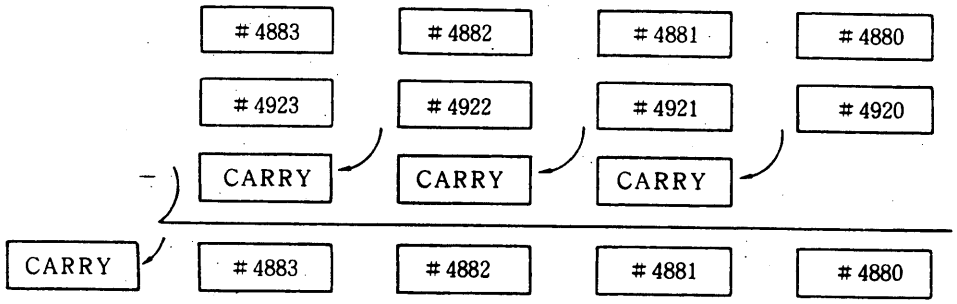
(b) If RR immediately before DSBS is 1, the contents of a specified number of consecutive registers beginning with register R1 are subtracted from the contents of the corresponding number of registers beginning with register R2. The number of subtractions is set in register R3.
If a borrow arises in the last register, RR and CARRY are set to 1.
If RR immediately before DSBS is 0, the DSBS instruction is not executed.

(c) A contact must be placed before DSBS.




```
LD    #48510
DSBS  #4920, #4880, #4820
```

If a content of #4820 is "4" (Decimal 8-digit Operation)



(17) MULW (Multiply word register) {RR}

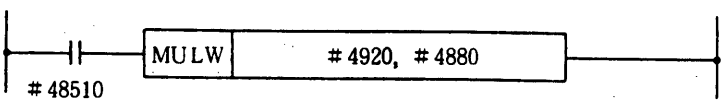
(a) Format: MULW #XXXX, #XXXX

Lower byte of double-length multiplicand register (WR2)

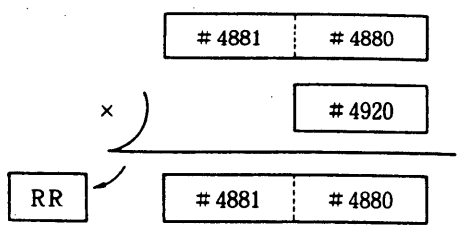
Multiplier register(R1)

(b) If RR immediately before MULW is 1, the contents of double-length register WR2 are multiplied by that of register R1, then the result is stored in double-length register WR2.
This instruction does not affect the contents of register R1 or CARRY.
If no overflow results from the multiplication, RR is set to 0. If an overflow results from the multiplication, RR is set to 1.

(c) A contact must be placed before MULW.




```
LD    #48510
MULW  #4920, #4880
```



Signs of the operands are ignored.

5.4 REGISTER INSTRUCTIONS (Cont'd)

(18) MULD (Multiply word register) {RR-}

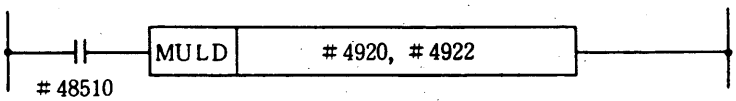
(a) Format: MULD #XXXX, #XXXX

Lower byte of double-length register (WR2)

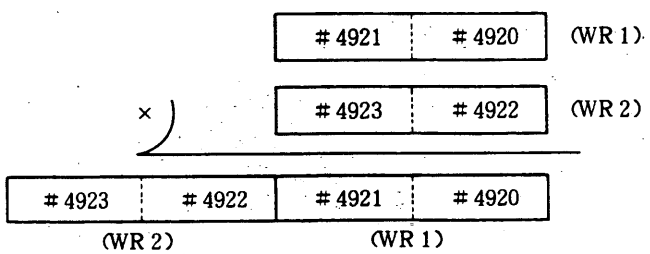
Lower byte of double-length register (WR1)

(b) If RR immediately before MULD is 1, the contents of double-length register WR1 are multiplied by that of WR2, then the result is stored using double-length registers WR1 and WR2.
This instruction does not affect the contents of RR or CARRY.
If RR immediately before MULD is 0, the MULD instruction is not executed.

(c) A contact must be placed before MULD.




```
LD    #48510
MULD  #4920, #4922
```



(19) DIVW (Divide word register) {RR 0}

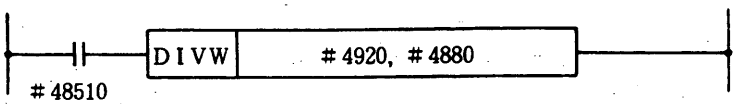
(a) Format: DIVW #XXXX, #XXXX

Lower byte of divident register (WR2)

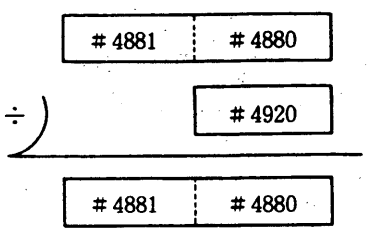
Divisor register (R1)

(b) If RR immediately before DIVW is 1, the contents of double-length register WR2 are divided by that of register R1, then the result is stored in double-length register WR2.
Signs of the operands are disregarded.
After the division, RR changes to 0. If the divisor is 0, RR remains 1.
This instruction does not affect the contents of register R1 or CARRY.
If RR immediately before DIVW is 0, the DIVW instruction is not executed.

(c) A contact must be placed before DIVW.




```
LD    #48510
MULW  #4920, #4880
```



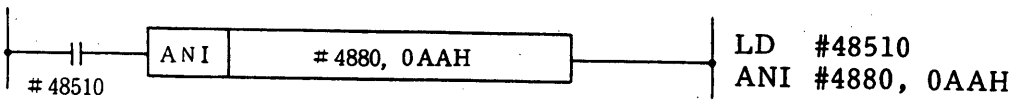
(20) ANI (And immediate) {RR-}

(a) Format: ANI #XXXX, XXH

#XXXX: Register; XXH: Numeric value (hexadecimals)

(b) If RR immediately before ANI is 1, the numeric value and the contents of the register are ANDed, then the result is stored in the register.
This instruction does not affect the contents of RR or CARRY.
If RR immediately before ANI is 0, the ANI instruction is not executed.

(c) A contact must be placed before ANI.




```
LD   #48510
ANI  #4880, 0AAH
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register	0	0	1	1	0	0	1	1
Numeric Value	1	0	1	0	1	0	1	0
Result	0	0	1	0	0	0	1	0

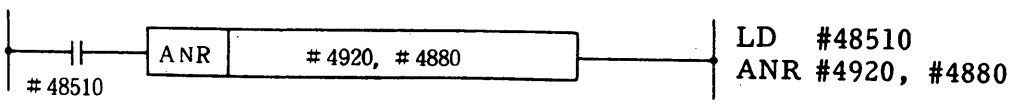
(21) ANR (Adn register) {RR-}

(a) Format: ANR #XXXX, #XXXX

#XXXX: Operand register (R1); #XXXX: Operand register (R2)

(b) If RR immediately before ANR is 1, the contents of registers R1 and R2 are ANDed, then the result is stored in register R2.
This instruction does not affect the contents of register R1, RR or CARRY.
If RR immediately before ANR is 0, the ANR instruction is not executed.

(c) A contact must be placed before ANR.




```
LD   #48510
ANR  #4920, #4880
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register (R2)	0	0	1	1	0	0	1	1
Register (R1)	0	1	0	1	0	1	0	1
Result (R2)	0	0	0	1	0	0	0	1

5.4 REGISTER INSTRUCTIONS (Cont'd)

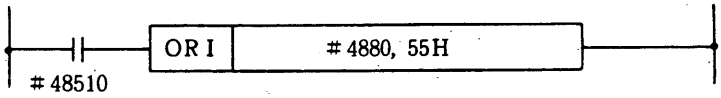
(22) ORI (Or immediate) {RR-}

(a) Format: ORI #XXXX, XXH

Register — Numeric value (hexadecimals)

(b) If RR immediately before ORI is 1, the numeric value and the contents of the register are ORed, then the result is stored in the register.
This instruction does not affect the contents of RR or CARRY.
If RR immediately before ORI is 0, the ORI instruction is not executed.

(c) A contact must be placed before ORI.




```
LD   #48510
ORI  #4880,55H
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register (R2)	0	0	1	1	0	0	1	1
Numeric Value	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	1	1	1

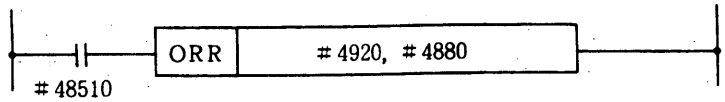
(23) ORR (Or register) {RR-}

(a) Format: ORR #XXXX, #XXXX

Operand register (R1) — Operand register (R2)

(b) If RR immediately before ORR is 1, the contents of the registers R1 and R2 are ORed, then the result is stored in the register R2.
This instruction does not affect the contents of register R1, RR or CARRY.
If RR immediately before ORR is 0, the ORR instruction is not executed.

(c) A contact must be placed before ORR.




```
LD   #48510
ORR  #4920, #4880
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register (R2)	0	0	0	1	0	0	0	1
Register (R1)	1	0	1	0	1	0	1	0
Result (R2)	1	0	1	1	1	0	1	1

(24) SAL (Shift left) {RR↕, CARRY↕}

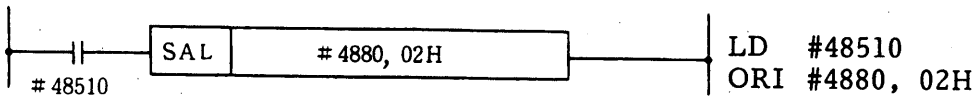
(a) Format: SAL #XXXX, XXH

Numeric value (hexadecimals)

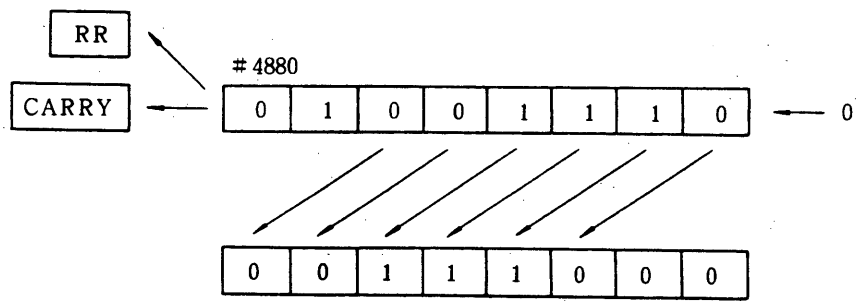
Register

(b) If RR immediately before SAL is 1, the contents of the register are shifted to the left for the numeric value. If data "1" overflows from the register after shifting, RR and CARRY are set to 1.
If RR immediately before SAL is 0, the SAL instruction is not executed.

(c) A contact must be placed before SAL.




```
LD   #48510
ORI  #4880, 02H
```



(25) SAR (Shift right) {RR↕, CARRY↕}

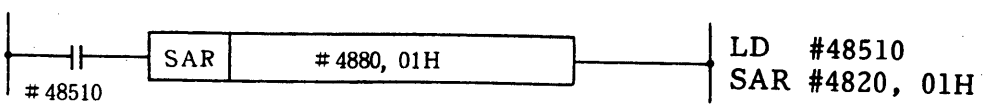
(a) Format: SAR #XXXX, XXH

Numeric value (hexadecimals)

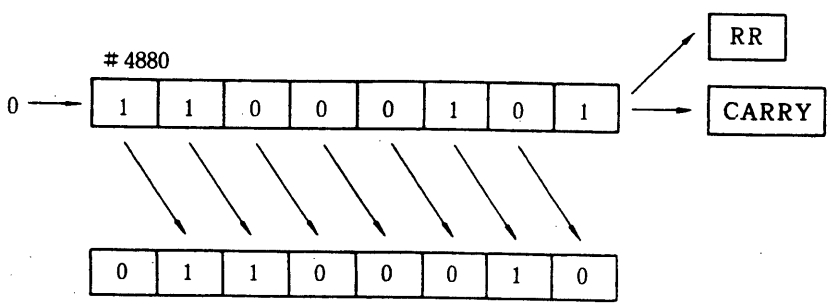
Register

(b) If RR immediately before SAR is 1, the contents of the register are shifted to the right for the numeric value. If data "1" overflows from the register after shifting, RR and CARRY are set to 1.
If RR immediately before SAR is 0, the SAR instruction is not executed.

(c) A contact must be placed before SAR.




```
LD   #48510
SAR  #4820, 01H
```



5.4 REGISTER INSTRUCTIONS (Cont'd)

(26) SALW (Shift left word register) {RR↕, CARRY↕}

This instruction is the same as SAL except a double-length register is to be shifted to the left instead of an 8-bit register.

(27) SARW (Shift right word register) {RR↕, CARRY↕}

This instruction is the same as SAR except a double-length register is to be shifted to the right instead of an 8-bit register.

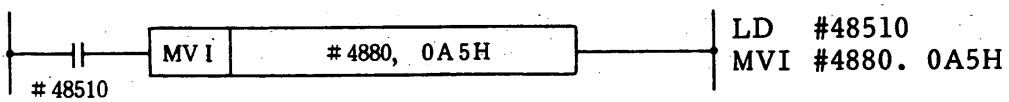
(28) MVI (Move immediate) {RR-}

(a) Format: MVI #XXXX, XXH

- #XXXX: Register
- XXH: Numeric value (hexadecimals)

(b) If RR immediately before MVI is 1, the numeric value is transferred to the register. This instruction does not affect the contents of RR or CARRY. If RR immediately before MVI is 0, the MVI instruction is not executed.

(c) A contact must be placed before MVI.




```
LD   #48510
MVI  #4880. 0A5H
```

(29) MVIW (Move immediated word register) {RR-}

This instruction is the same as the MVI instruction except that the constant is to be transferred to a double-length register instead of to an 8-bit register.

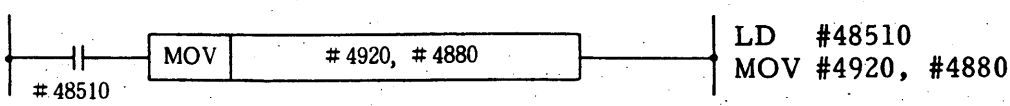
(30) MOV (Move register) {RR-}

(a) Format: MOV #XXXX, #XXXX

- #XXXX: Operand register (R1)
- #XXXX: Operand register (R2)

(b) If RR immediately before MOV is 1, the contents of register R1 are transferred to register R2.
This instruction does not affect the contents of register R1, RR or CARRY. The registers (R1 and R2) may be one byte consisting of a set of I/O relays. If RR immediately before MOV is 0, the MOV instruction is not executed.

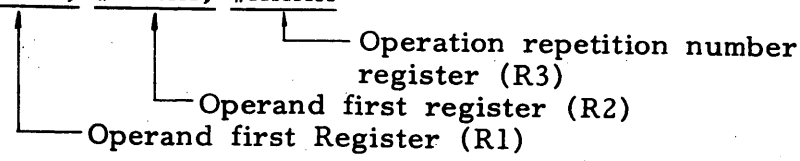
(c) A contact must be placed before MOV.




```
LD   #48510
MOV  #4920, #4880
```

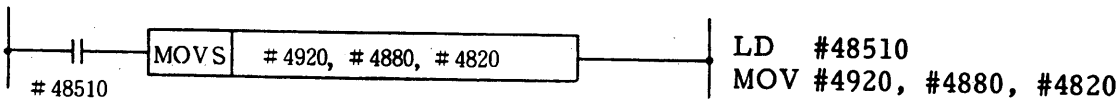
(31) MOVS (Move string) {RR-}

(a) Format: MOVS #XXXX, #XXXX, #XXXX

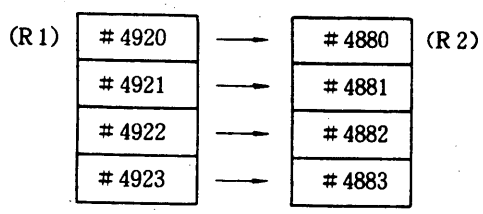


(b) If RR immediately before MOVS is 1, the contents of a specified number of consecutive registers beginning with register R1 are transferred to the corresponding number of registers beginning with register R2. The number of subtractions is set in register R3.
This instruction does not affect the contents of RR, CARRY, or consecutive registers beginning with R1.
If RR immediately before MOVS is 0, the MOVS instruction is not executed.

(c) A contact must be placed before MOVS.

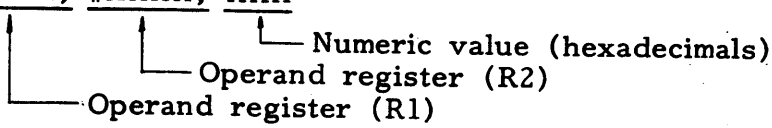


If #4820 is "04"



(32) DST (Data store) {RR-}

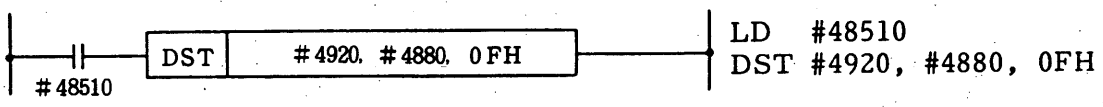
(a) Format: DST #XXXX, #XXXX, XXH



(b) If RR immediately before DST is 1, the numeric value and the contents of register R1 are ANDed, then the result is stored in register R2.
This instruction does not affect the contents of register R1, RR or CARRY.
If RR immediately before DST is 0, the DST instruction is not executed.

5.4 REGISTER INSTRUCTIONS (Cont'd)

(c) A contact must be placed before DST.




```
LD   #48510
DST  #4920, #4880, 0FH
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register (R1)	A	A	A	A	A	A	A	A
Numeric Value	0	0	0	0	1	1	1	1
Register (R2)	0	0	0	0	A	A	A	A

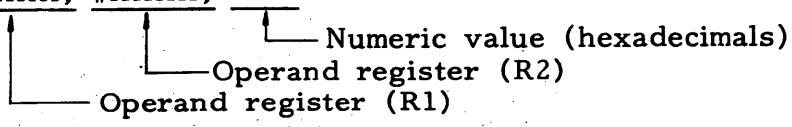
A: 1 or 0

(33) DSTW (Data store word register) {RR-}

This instruction is the same as the DST instruction except that this instruction is used for double-length registers (WR1 and WR2) instead of for registers (R1 and R2).

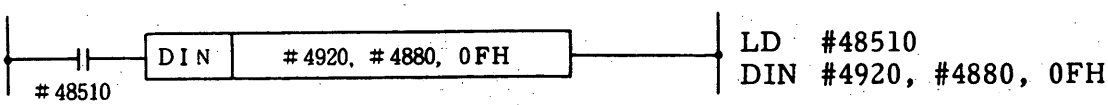
(34) DIN (Data insert) {RR-}

(a) Format: DIN #XXXX, #XXXX, XXH



(b) If RR immediately before DIN is 1, the inverted numeric data and the contents of register R1 are ANDed, then the result and the contents of register R1 are ORed.
The last result is stored in register R2.
This instruction does not affect the contents of register R1, RR or CARRY.
If RR immediately before DIN is 0, the DIN instruction is disregarded.

(c) A contact must be placed before DIN.




```
LD   #48510
DIN  #4920, #4880, 0FH
```

	D7	D6	D5	D4	D3	D2	D1	D0
Register (R1)	A	A	A	A	A	A	A	A
Register (R2)	B	B	B	B	B	B	B	B
Numeric Value	0	0	0	0	1	1	1	1
Result	B	B	B	B	A	A	A	A

A: 1 or 0. B: 1 or 0.

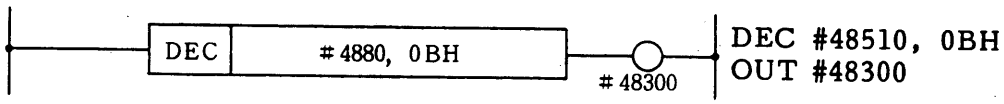
(35) DEC (Decode) {RR↓}

(a) Format: DEC #XXXX, XXH

#XXXX: Register
XXH: Numeric value (hexadecimals)

(b) The contents of the register are compared to the numeric value. If they match, RR is turned to 1.
The DEC instruction is executed regardless of the status of RR immediately before it.
This instruction does not affect the contents of registers or CARRY.

(c) Contacts must not be added before DEC. To add a contact, use the COI instruction.




```
DEC #48510, 0BH
OUT #48300
```

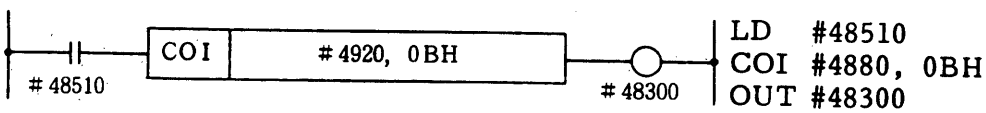
(36) COI (Coincide immediate) {RR↓}

(a) Format: COI #XXXX, XXH

#XXXX: Register
XXH: Numeric value (hexadecimals)

(b) If RR immediately before COI is 1, the contents of the register are compared to the numeric value. If they match, RR is turned to 1.
If not, RR is turned to 0.
This instruction does not affect the contents of registers or CARRY.
If RR immediately before COI is 0, the COI instruction is disregarded.

(c) A contact must be placed before COI.




```
LD   #48510
COI  #4880, 0BH
OUT  #48300
```

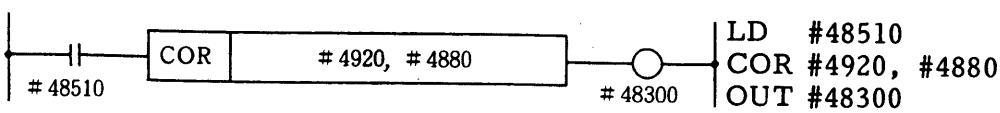
(37) COR (Coincide register) {RR↓}

(a) Format: COR #XXXX, #XXXX

#XXXX: Register (R1)
#XXXX: Register (R2)

(b) If RR immediately before COR is 1, the contents of registers R2 and R1 are compared to the numeric value. If they match, RR is turned to 1. If not, RR is turned to 0.
This instruction does not affect the contents of CARRY or registers R1 and R2.
If RR immediately before COR is 0, the COR instruction is disregarded.

(c) A contact must be placed before COR.




```
LD   #48510
COR  #4920, #4880
OUT  #48300
```

5.4 REGISTER INSTRUCTIONS (Cont'd)

(38) CORW (Coincide word register) {RR↓}

This instruction is the same as the COR instruction except that this instruction is used for matching double-length registers (WR1 and WR2) instead of registers (R1 and R2).

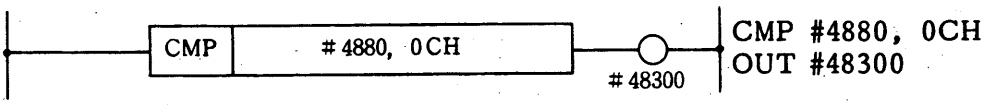
(39) CMP (Compare) {RR↓}

(a) Format: CMP #XXXX, XXH

- #XXXX: Register
- XXH: Numeric value (hexadecimals)

(b) The contents of the register are compared to the numeric value.
If register contents ≥ numeric value, RR is turned to 1.
If register contents < numeric value, RR is turned to 0.
The CMP instruction is executed regardless of the status of RR immediately before it.
This instruction does not affect the contents of registers or CARRY.

(c) Contacts must not be added before CMP. To add a contact, use the CPI instruction.



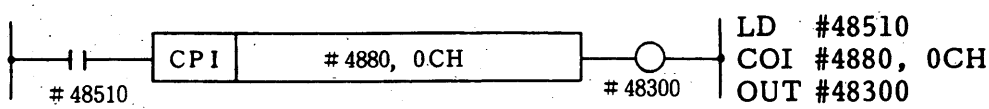
(40) CPI (Compare immediate) {RR↓}

(a) Format: CPI #XXXX, XXH

- #XXXX: Register
- XXH: Numeric value (hexadecimals)

(b) If RR immediately before CPI is 1, the contents of the register are compared to the numeric value.
If register contents ≥ numeric value, RR is turned to 1.
If register contents < numeric value, RR is turned to 0.
This instruction does not affect the contents of registers or CARRY.
If RR immediately before CPI is 0, the CPI instruction is disregarded.

(c) A contact must be placed before CPI.



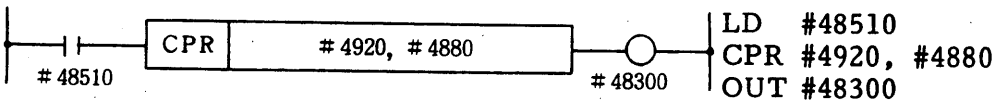
(41) CPR (Compare register) {RR↓}

(a) Format: CPR #XXXX, #XXXX

- #XXXX: Register (R1)
- #XXXX: Register (R2)

(b) If RR immediately before CPR is 1, the contents of registers R2 and R1 are compared to the numeric value.
If R1 ≥ R2, RR is turned to 1.
If R1 < R2, RR is turned to 0.
This instruction does not affect the contents of CARRY or registers R1 and R2.
If RR immediately before CPR is 0, the CPR instruction is disregarded.

(c) A contact must be placed before CPR.



(42) CPRW (Compare word register) {RR↕}

This instruction is the same as the CPR instruction except that this instruction is used for making comparison between double-length registers (WR1 and WR2) instead of between registers (R1 and R2).

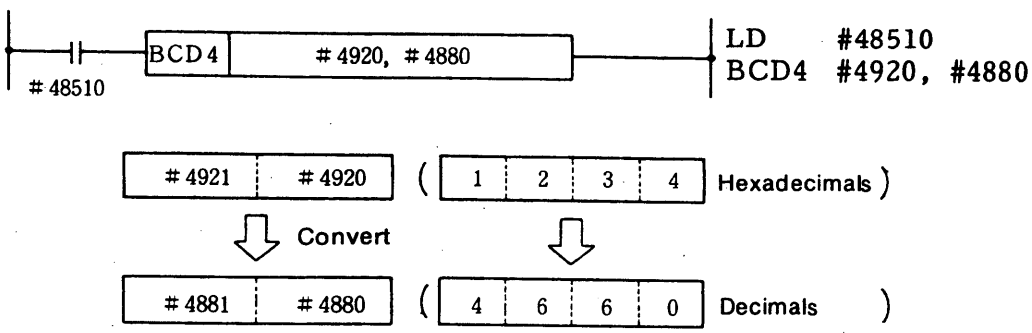
(43) BCD4 (Binary to 4 digits BCD) {RR↕}

(a) Format: BCD4 #XXXX, #XXXX

Operation result first register (R2)
Operation first register (R1)

(b) If RR immediately before BCD4 is 1, hexadecimal data are read from two consecutive bytes beginning with register R1, converted into a 4-digit BCD code, and stored to two consecutive bytes beginning with register R2.
If the converted BCD code consists of five or more digits (which occurs when register R1 contains 2710H or greater value), the data are not converted and RR is set to 1.
If the converted BCD code consists of four or less digits, RR is set to 0.
The sign of the data is disregarded.
This instruction does not affect the contents of register R1 or CARRY. If RR immediately before BCD4 is 0, the BCD4 instruction is disregarded.

(c) A contact must be placed before BCD4.



5.4 REGISTER INSTRUCTIONS (Cont'd)

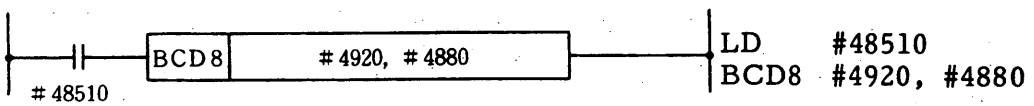
(44) BCD8 (Binary to 8 digits BCD) {RR‡}

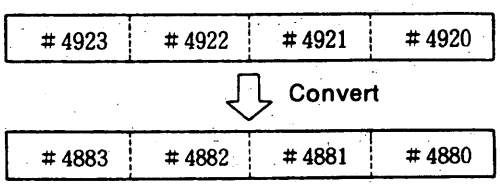
(a) Format: BCD8 #XXXX, #XXXX

Operation result first register (R2)
Operation first register (R1)

(b) If RR immediately before BCD8 is 1, hexadecimal data are read from four consecutive bytes beginning with register R1, converted into an 8-digit BCD code, and stored to four consecutive bytes beginning with register R2.
If the converted BCD code consists of nine or more digits (which occurs when register R1 contains 05F5E100H or greater value), the data are not converted and RR is set to 1.
If the converted BCD code consists of eight or less digits, RR is set to 0.
The signs of the data are disregarded.
This instruction does not affect contents of register R1 or CARRY.
If RR immediately before BCD8 is 0, the BCD8 instruction is disregarded.

(c) A contact must be placed before BCD8.





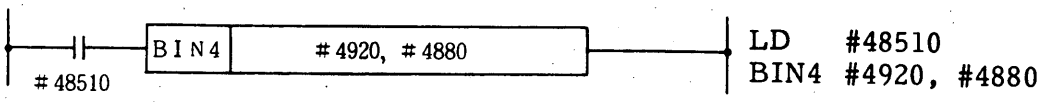
(45) BIN4 (4 digits BCD to binary) {RR‡}

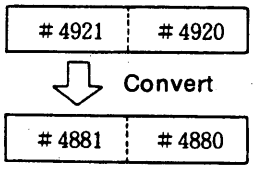
(a) Format: BIN4 #XXXX, #XXXX

Operation result first register (R2)
Operation first register (R1)

(b) If RR immediately before BIN4 is 1, the 4-digit BCD code stored in two consecutive bytes beginning with register R1 is read, converted into binary data, and stored to two consecutive bytes beginning with register R2.
The signs of the data are disregarded.
If conversion is impossible (because of an invalid BCD input), the data are not converted and RR is set to 1.
If the data are converted normally, RR is set to 0.
This instruction does not affect the contents of register R1 or CARRY.

(c) A contact must be placed before BIN4.





(46) BIN8 (8 digits BCD to binary) {RR}

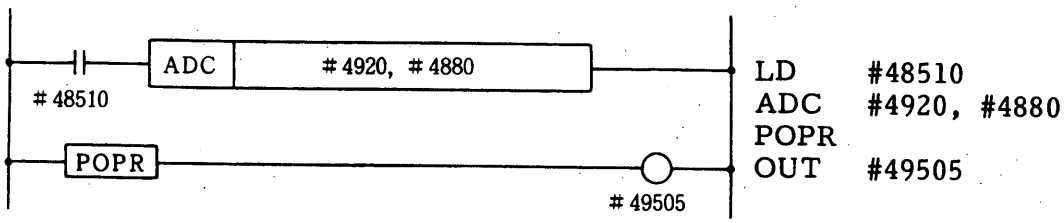
This instruction is the same as the BIN4 instruction except that this instruction converts an 8-digit BCD code into binary data instead of a 4-digit BCD code.

(47) POPR (POP RR) {RR}

(a) Format: POPR

(b) This instruction is used next to a register instruction by which the contents of RR are changed.
This instruction enables the use of the operation result of a line stored in RR on the next line.

(c)



As shown in the figure above, the POPR instruction can be used as the first instruction on a line. In this example, the POPR instruction enables the use of RR contents in the logic program when an overflow in ADC occurs and RR and CARRY are set to 1.

(48) ERROR (ERROR) {RR}

(a) If an error occurs when a ladder program is down-loaded to the built-in sequencer, the ERROR information is stored in the middle of the ladder.
When this file is up-loaded the message "ERROR" is inserted in place of the instruction that could not be analyzed at download.

6. LOGIC PROGRAM EDITING

6.1 LOGIC PROGRAM EDITING

The personal computer is used when a logic program is edited.

Personal computer programs are required for logic program transfer between the personal computer and the Motionpack, which must be supplied by the users. Refer to the attached material for the typical reference programs made by YASKAWA.

A file to be created for the first time by the personal computer is called "original ladder file". While, a file that is uploaded from the built-in sequencer after the "original ladder file" is downloaded to the built-in sequencer is called "list format file". Attention must be paid for editing the files since the "original ladder file" and "list format file" have different formats.

6.2 FORMAT OF ORIGINAL LADDER FILE

A file to be created for the first time by the personal computer is called "original ladder file" and its format is as described below:

(1) When the original ladder file is created, enter % ⏎ ORG ⏎ at the head of the ladder program.

(Example)

```
%⏎                              ····· Ladder file start
ORG⏎                            ····· Original file symbol
LD          #42080⏎  ┐
AND-NOT     #45011⏎  ├ ····· Ladder program
OUT         #43030⏎  ┤
LD          #42082⏎  ┘
⋮
%⏎                              ····· Ladder file end
```

(2) Enter a command to the left without any space.

Nothing can be entered before a command.

(3) When a space is provided between lines, use ⏎ return key.

(4) Depress the TAB key between a command and address.
The space key cannot be used.
When the TAB key is depressed, control it so that TAB code (09) can be output without fail.

(5) After an address is entered, depress ⏎ return key.

(6) Do not fail to enter % ⏎ at the end of the ladder program.

(7) ⏎ indicates (CR, LF). In downloading the LF code is disregarded.

6.3 FORMAT OF LIST FORMAT FILE

A ladder file that is uploaded from the built-in sequencer of the Motionpack FD is called "list format file" and its format is as described below:

(1) ORG is not provided for ladder files output from the Motionpack FD.

(2) The following commands are output with the net number and two spaces provided before the commands:

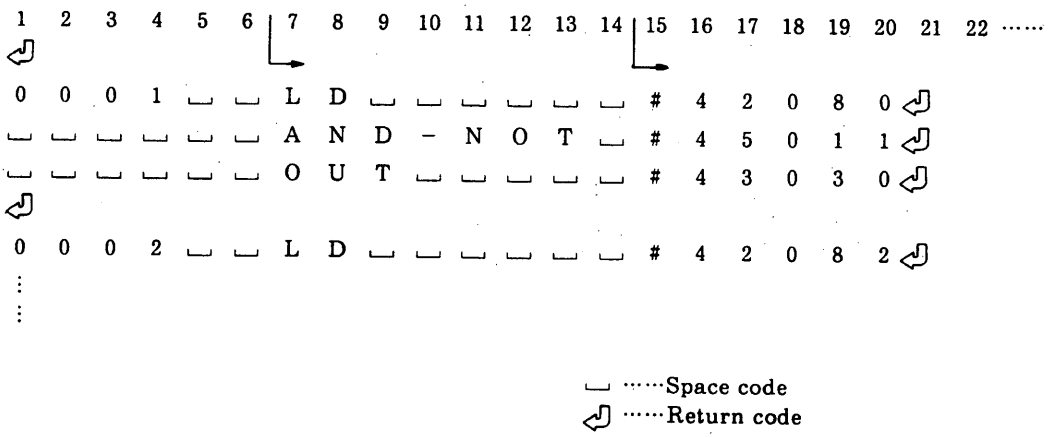
LD, LD-NOT, NOP, RET, SET, RST, STC,
CLC, DEC, CMP, POPR

The above commands can be the head of one net.

(3) A line without any net number is provided with six spaces before the command and the command is output from 6th column.

(4) Spaces are also provided between a command and address.
The number of spaces is determined so that the address can start from the 15th column from the head.

(5) When a net number is added, the return code is also added before the number.



(6) When this list format file is edited, the TAB key cannot be used. Provide a space by using the space key.
This list format file is a fixed format. (A command is output from the 7th column and an address from the 15th column. A space is provided in the space code.)
A wrong format cannot download the file.

(7) As described above, a list format file has a net number. When it is downloaded, this information is not input to the Motionpack FD built-in sequencer.

(8) When a file is transferred from the built-in sequencer to the personal computer, the LF code is added to the CR code.

6.4 CONNECTION WITH PERSONAL COMPUTER

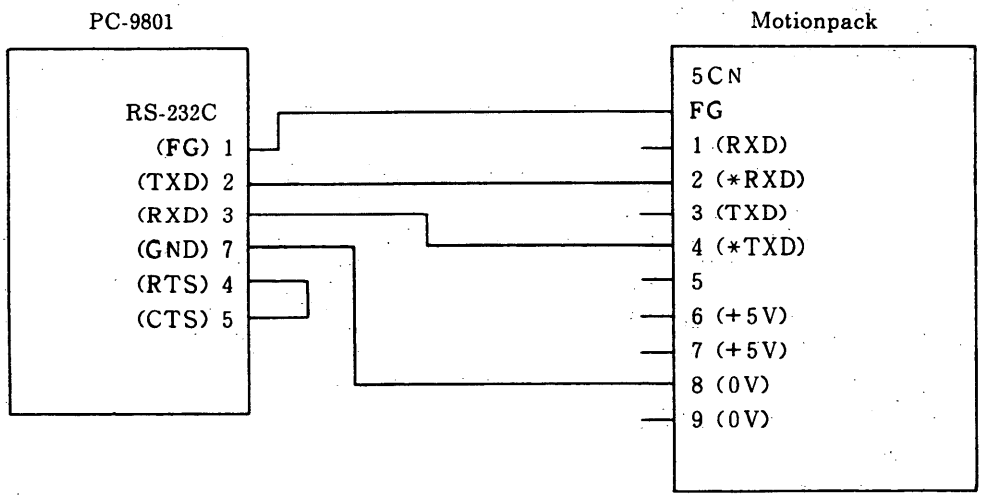
Either 5CN or 6CN of the Motionpack FD is used for connection with the personal computer.

After the power supply is turned on, either 5CN or 6CN (whichever has received a transmission request first) opens the port to enable the Motionpack FD to transmit with the personal computer.

If the power supply is turned on with the exclusive-use programmer connected to 5CN, 5CN is ready to be used. This is because the exclusive-use programmer sends a transmission request automatically after the power supply is turned on. Therefore, when the status where the programmer is used is used is switched to the personal computer, it is convenient to connect it using 5CN.

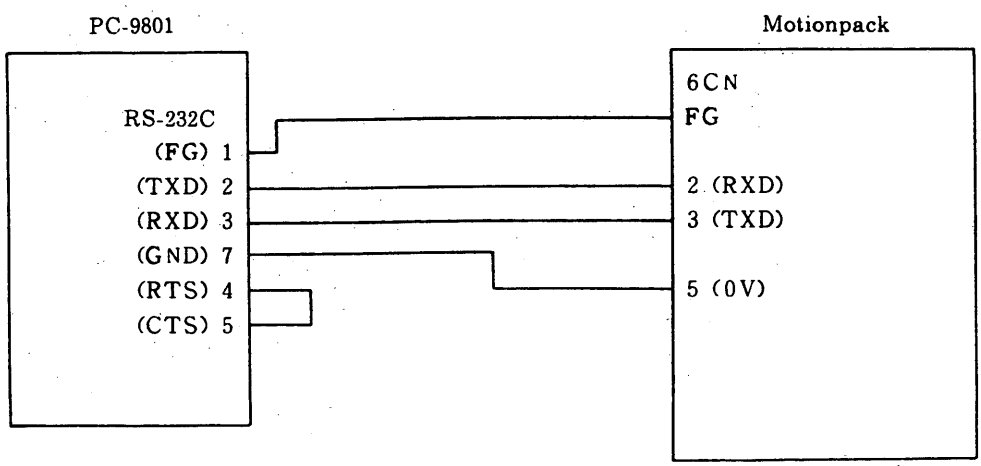
(1) Connection Diagrams

(a) When 5CN is used



NOTE: Cable length between the PC-9801 and Motionpack must be 1 m or less.

(b) When 6CN is used



(2) Transmission Conditions

Set the personal computer transmission conditions as shown below:

- Transmission speed: 9600 bps
- Bit length: 8 bits
- Stop bit: 1 bit
- Parity: Provided, even-number parity
- XON/OFF control: Provided
- Shift control: Not provided
- Transmission port: RS-232 port

7. CHECKING LOGIC

7.1 CHECKING LOGIC PROGRAM SYNTAX

Built-in sequencer checks the logic program syntax and displays the following error codes.

Table 7.1 Error Codes of Built-in Sequencer

Error Code	Error Type	Cause
01	Syntax error	—
02	I/O or register number error	An invalid number is used. (Available addresses are from #4000 to #5999.)
03	Upload error	The file could not be up-loaded because of lost memory.
04	Memory overflow	The ladder is too long to download.

Note The error code is issued when download fails. The built-in sequencer replaces the ladder instruction where the error occurred with an error instruction.
When the list file is up-loaded, the error code appears in the list.
Correct the error marked with the code and down-load the file again.

8. STORING LOGIC PROGRAM IN ROM

8.1 CONNECTING TO PROM WRITER

The ADVANTEST or Minato Electronics PROM writer is recommended.

Prepare a cable for connecting the PROM writer and the personal computer.

(1) Connection Diagram

Table 8.1 Connection Diagram

ADVANTEST PROM Writer				Minato PROM Writer			
PC-9801		PROM writer		PC-9801		PROM writer	
FG	1	1	FG	FG	1	1	FG
TXD	2	2	TXD	TXD	2	2	TXD
RXD	3	3	RXD	RXD	3	3	RXD
RTS	4	4	RTS	RTS	4	4	RTS
CTS	5	5	CTS	CTS	5	5	CTS
GND	7	7	GND	GND	7	7	GND

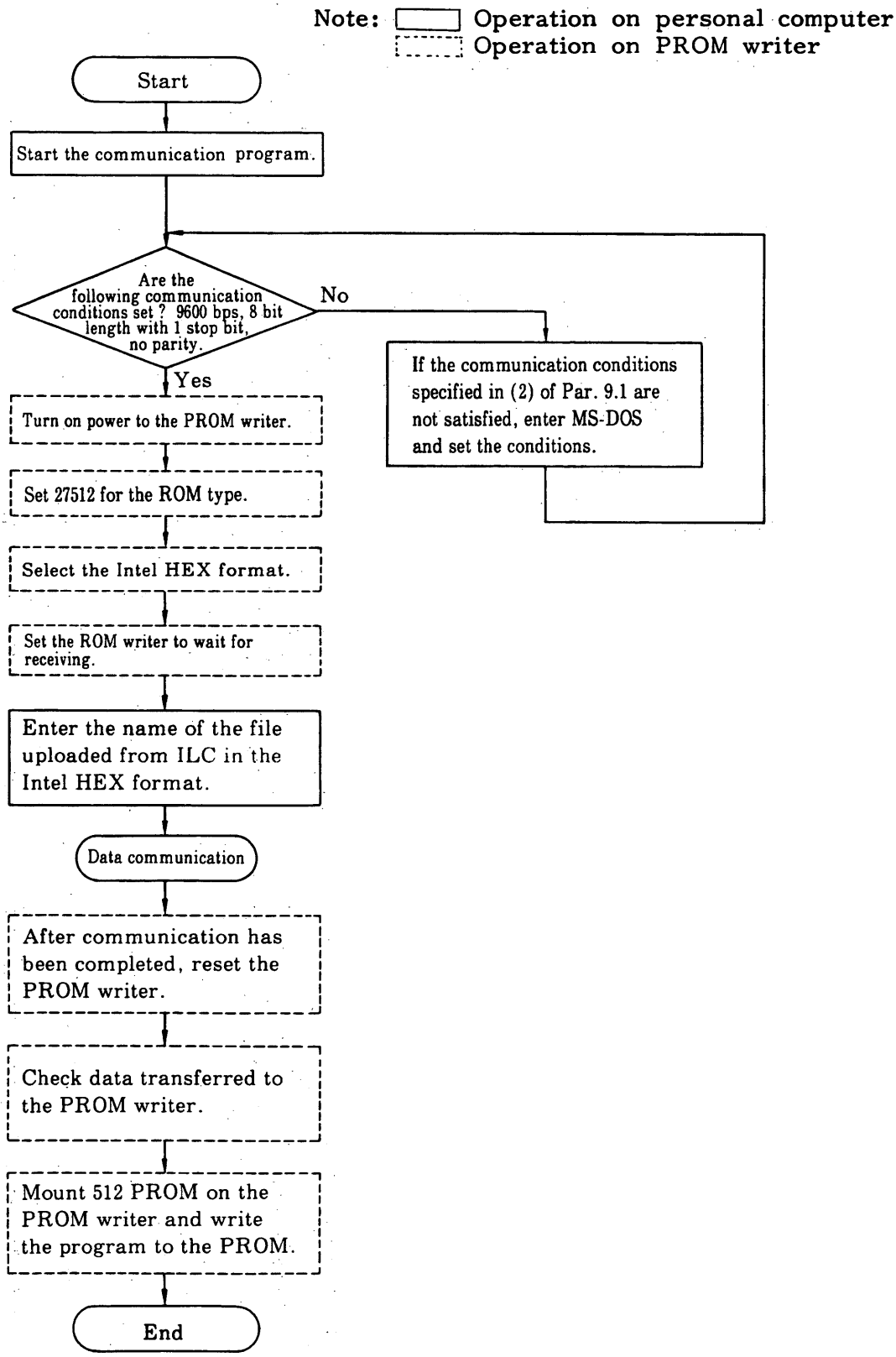
(2) Communication Conditions

Set the PROM writer as follows. To set otherwise, match the communication condition setting between the personal computer and the PROM writer.

- Communication speed: 9600 bps
- Bit length: 8 bits
- Stop bit: 1 bit
- Parity: Not used
- XON/XOFF control: Used
- Shift control: Not used

8.2 PROCEDURE FOR STORING LOGIC PROGRAM INTO ROM

Observe the following flowchart:



9. FIXED I/O SIGNALS

9.1 FIXED INPUT SIGNALS

Table 9.1 Fixed Input Signals

<table>
<tr><th>No.</th><th>Address</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>#4000
to
#4002</td><td></td><td>Same as standard input signals.
Signal names are shown below.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Re-marks</th></tr>
<tr><td>#4000</td><td>ZRN</td><td>-JS</td><td>+JS</td><td>JSPD</td><td>HANDLE</td><td>JOG</td><td>PLAY</td><td>EDIT</td><td></td></tr>
<tr><td>#4001</td><td>PGSL4</td><td>PGSL3</td><td>PGSL2</td><td>PGSL1</td><td>MFIN</td><td>G34F</td><td>SBLK</td><td>PGST</td><td></td></tr>
<tr><td>#4002</td><td></td><td>SVON</td><td>ESP6</td><td>ESP5</td><td>INC8/9</td><td>-INC</td><td>+INC</td><td>ERS</td><td></td></tr>
</table></td></tr>
<tr><td>2</td><td>#4010
to
#4013</td><td></td><td>Externally-set data.
Data length is fixed in 4 bytes.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Remarks</th></tr>
<tr><td>#4010</td><td>D07</td><td>D06</td><td>D05</td><td>D04</td><td>D03</td><td>D02</td><td>D01</td><td>D00</td><td>Exter-nally set-data</td></tr>
<tr><td>#4011</td><td>D17</td><td>D16</td><td>D15</td><td>D14</td><td>D13</td><td>D12</td><td>D11</td><td>D10</td><td>Exter-nally set-data</td></tr>
<tr><td>#4012</td><td>D27</td><td>D26</td><td>D25</td><td>D24</td><td>D23</td><td>D22</td><td>D21</td><td>D20</td><td>Exter-nally set-data</td></tr>
<tr><td>#4013</td><td>D37</td><td>D36</td><td>D35</td><td>D34</td><td>D33</td><td>D32</td><td>D31</td><td>D30</td><td>Exter-nally set-data</td></tr>
</table></td></tr>
</table>

9.1 FIXED INPUT SIGNALS (Cont'd)

Table 9.1 Fixed Input Signals (Cont'd)

<table>
<tr><th>No.</th><th>Address</th><th>Name</th><th>Description</th></tr>
<tr><td>3</td><td>#4014</td><td>Variable Number (N7 to N0)</td><td>Variable numbers such as register numbers offset numbers are indicated in binary data.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Remarks</th></tr>
<tr><td>#4014</td><td>N7</td><td>N6</td><td>N5</td><td>N4</td><td>N3</td><td>N2</td><td>N1</td><td>N0</td><td>Variable nos.</td></tr>
</table></td></tr>
<tr><td>4</td><td>#4015</td><td></td><td>
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Remarks</th></tr>
<tr><td>#4015</td><td>REQ</td><td>IN/ OUT</td><td>INC/ ABS</td><td>POS</td><td></td><td>RESV'D</td><td>OFS</td><td>REG</td><td></td></tr>
</table>
<table>
<tr><th>Address</th><th>Name</th><th>Description</th></tr>
<tr><td>#40150</td><td>REG</td><td>Indicates that data indicated by external setting data are register data.</td></tr>
<tr><td>#40151</td><td>OFS</td><td>Indicates that data indicated by external setting data are offset data.</td></tr>
<tr><td>#40152</td><td>RE-SERVED</td><td></td></tr>
<tr><td>#40154</td><td>POS</td><td>Indicates that data indicated by external setting data are position data.</td></tr>
<tr><td>#40155</td><td>INC/ ABS</td><td>Indicates the type of compensation data.
OFF: Absolute value compensation
ON: Incremental value compensation</td></tr>
<tr><td>#40156</td><td>IN/ OUT</td><td>Indicates external data input or output.
OFF: External data output request
ON: External data input request</td></tr>
<tr><td>#40157</td><td>REQ</td><td>Indicates external data I/O request.
Turn this ON when external data transfer is required.</td></tr>
</table></td></tr>
</table>

9.2 FIXED OUTPUT SIGNALS

Table 9.2 Fixed Output Signals

<table>
<tr><th>No.</th><th>Address</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>#4500
to
#4502</td><td></td><td>Same as standard output signals.
Signal names are shown below.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Re-marks</th></tr>
<tr><td>#4500</td><td>EPALM</td><td>G34</td><td>OFM</td><td>OFR</td><td>INCD</td><td>STL</td><td>SALM</td><td>MRDY</td><td></td></tr>
<tr><td>#4501</td><td>M57</td><td>M56</td><td>M55</td><td>M54</td><td>M53</td><td>M52</td><td>M51</td><td>M50</td><td></td></tr>
<tr><td>#4502</td><td>BALM</td><td>M30</td><td>PSW4</td><td>PSW3</td><td>PSW2</td><td>PSW1</td><td>CLD</td><td>M58</td><td></td></tr>
</table></td></tr>
<tr><td>2</td><td>#4510
to
#4513</td><td></td><td>Data set to external devices.
Data length is fixed in 4 bytes.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Remarks</th></tr>
<tr><td>#4510</td><td>E07</td><td>E06</td><td>E05</td><td>E04</td><td>E03</td><td>E02</td><td>E01</td><td>E00</td><td>Output data</td></tr>
<tr><td>#4511</td><td>E17</td><td>E16</td><td>E15</td><td>E14</td><td>E13</td><td>E12</td><td>E11</td><td>E10</td><td>Output data</td></tr>
<tr><td>#4512</td><td>E27</td><td>E26</td><td>E25</td><td>E24</td><td>E23</td><td>E22</td><td>E21</td><td>E20</td><td>Output data</td></tr>
<tr><td>#4513</td><td>E37</td><td>E36</td><td>E35</td><td>E34</td><td>E33</td><td>E32</td><td>E31</td><td>E30</td><td>Output data</td></tr>
</table></td></tr>
<tr><td>3</td><td>#45147</td><td>Read-out completed</td><td>Shows completion of external data setting and position data read-out.
<table>
<tr><th>Bit / Address</th><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th><th>Remarks</th></tr>
<tr><td>#4514</td><td>DACK</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Control</td></tr>
</table></td></tr>
</table>

10. EXTERNAL DATA SETTING AND INTERNAL DATA READ-OUT

The Motionpack FD model 1 can set the external data and read out the position data as shown below, by using the built-in PLC function.

① Register data setting (R01 to R99)

② Offset data setting, T_8 and T_9 coordinate system external compensation

③ Position data output

10.1 SIGNALS

Data exchange between Motionpack main controller and built-in PLC is performed in sequence by internal output signals.

(1) Controller Side Input Signals

① Variable nos. (N_0 to N_6): Specifies variable nos. such as register nos.

② Variable type
REG: Specifies register data.
OFS: Specifies offset data.
POS: Specifies position data.

③ I/O
IN/OUT: Outputs at OFF, inpus at ON.

④ Data I/O request
REQ: Requests data I/O execution.

⑤ Compensation data type
INC/ABS: Absolute value compensation at OFF, incremental value compensation at ON.

⑥ Input data (D00 to D37): Inputs external data setting value (fixed to 4 bytes).

(2) Controller Side Output Signals

① Completion signal
DACK: Completion signal of external data setting and position data read-out.

② Output data (E00 to E37): Internal data read-out value (fixed to 4 bytes).

10.2 I/O MAP (INTERNALLY-FIXED ADDRESS)

(1) Input

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4010	D07	D06	D05	D04	D03	D02	D01	D00	Externally-set data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4011	D17	D16	D15	D14	D13	D12	D11	D10	Externally-set data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4012	D27	D26	D25	D24	D23	D22	D21	D20	Externally-set data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4013	D37	D36	D35	D34	D33	D32	D31	D30	Externally-set data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4014	N7	N6	N5	N4	N3	N2	N1	N0	Variable nos.

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4015	REQ	IN/OUT	INC/ABS	POS		RESV'D	OFS	REG	Variable type, control

10.2 I/O MAP (INTERNALLY-FIXED ADDRESS) (Cont'd)

(2) Output

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4510	E07	E06	E05	E04	E03	E02	E01	E00	Output data

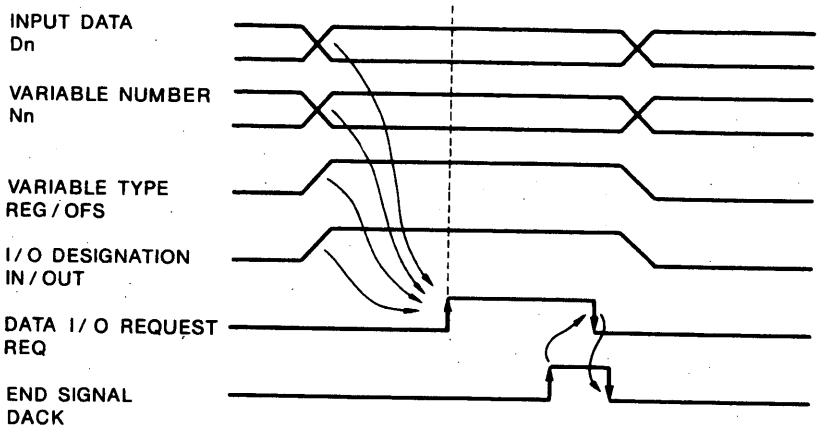
Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4511	E17	E16	E15	E14	E13	E12	E11	E10	Output data

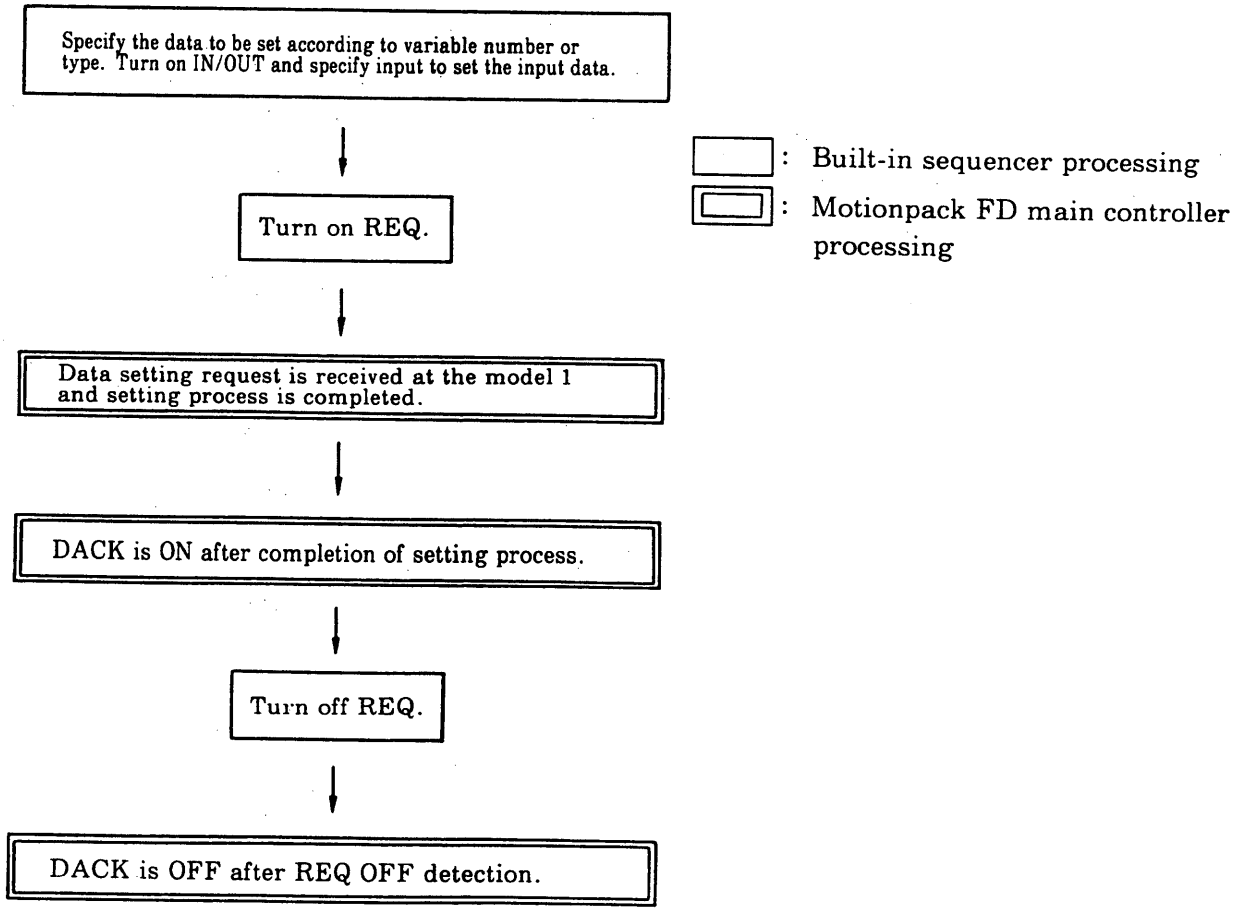
Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4512	E27	E26	E25	E24	E23	E22	E21	E20	Output data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4513	E37	E36	E35	E34	E33	E32	E31	E30	Output data

Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4514	DACK								Control

10.3 EXTERNAL DATA SETTING SEQUENCE





As shown in the above charts, external setting of data is performed by the built-in sequencer processing and Motionpack FD main controller processing.
It is necessary for users to create ladder program of built-in sequencer processing.
In this case, the following shows the external data specification.

10.3 EXTERNAL DATA SETTING SEQUENCE (Cont'd)

External Data Specifications

① The data set externally are stored in the specified register and become data for feeding reference position, speed, torque, etc.

② Data length

· Fixed to 4 bytes
32-bit binary (complement expression of two) input

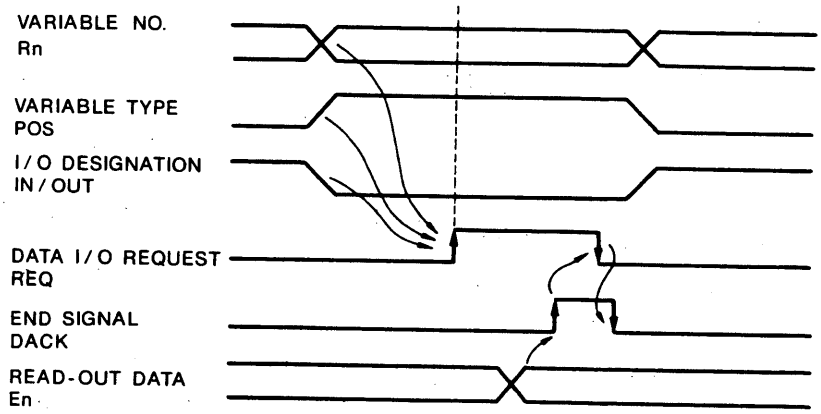
③ Data configuration

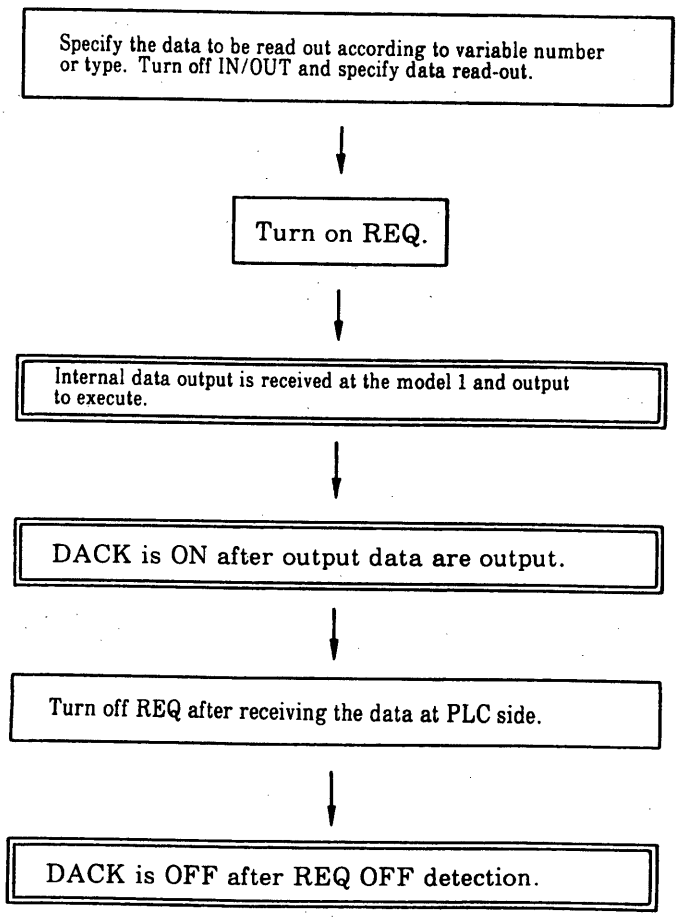
[Setting data]+[Variable No. (register No.)]+[Variable type REG designation)]

④ Data save

Read-in data stored in R01 to R99. At this time, pay attention not to misunderstand the data (eg. position data are used as torque data or speed data) since the register numbers are not individually defined for speed, position and torque data.

10.4 INTERNAL DATA READ-OUT SEQUENCE





: Built-in sequencer processing

: Motionpack FD main controller processing

10.4 INTERNAL DATA READ-OUT SEQUENCE (Cont'd)

As shown in the above charts, internal data read-out is performed by the built-in sequencer processing and Motionpack FD main controller processing.
It is necessary for users to create ladder program of built-in sequencer processing.
The following shows the read-out internal data specification.

Internal Data Specifications

① Can be read-out by specifying position data, offset data, and register data of variable types.

② Data type

32-bit binary (complement expression of two) output

11. EXTERNAL COMPENSATION

Data can be set externally to the offset registers for compensation by the Motionpack FD eternal data setting function.

11.1 SPECIFICATIONS OF COMPENSATION DATA

① Type

The contents of offset register O_8 or O_9 can be operated. The Motionpack FD compensating function operates the contents of the offset value registers shown below.

Coordinate System	Reference System Current Value	Shift Value Register	Offset Value Register	
T_0	A_0			
T_1	A_1	S_1		
T_2	A_2	S_2		
T_3	A_3	S_3		
T_4	A_4	S_4		
T_5	A_5	S_5		
T_6	A_6	S_6		
T_7	A_7	S_7		
T_8	A_8	S_8	O_8	O_8 or O_9 can be operated by external compensation.
T_9	A_9	S_9	O_9	

Compensation is possible in the external compensating function by using O_8 or O_9.

② External compensating data

External compensation data: 0 to ±99999999 (Unit: Position reference unit)

③ Data length

- Fixed to 4 bytes
- Binary code with signs

④ Type of compensation data

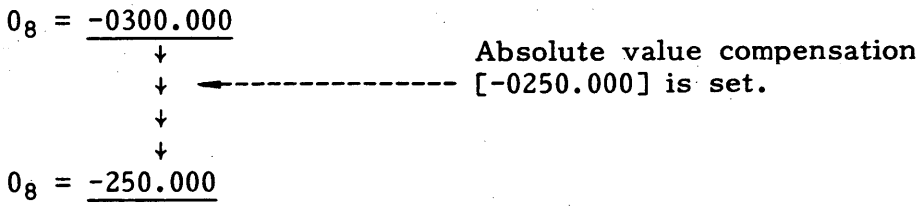
There are two types of compensation data: absolute value compensation and incremental value compensation, which are classified by types of variables (#4015-D5) INC/ABS bit.

INC/ABS = OFF: Absolute value compensation
INC/ABS = ON: Incremental value compensation

11.2 ABSOLUTE VALUE COMPENSATION

In the absolute value compensation, external compensation data become the contents of the specified offset value register (O_8 or O_9) without changing.

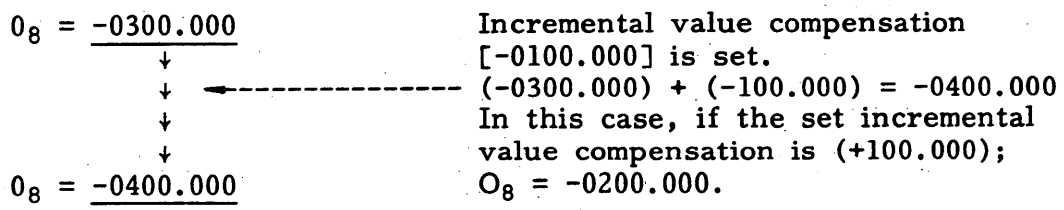
(Example)



11.3 INCREMENTAL VALUE COMPENSATION

In the incremental value compensation, external compensation data are added to or subtracted from the current value of the specified offset value register (O_8 or O_9).

(Example)



11.4 COMPENSATION CLEAR

When [00000.000] is set, the contents of the specified offset value register are cleared to 0. This indicates the same as setting absolute value compensation to 0.

11.5 DATA INPUT

The same as the external data setting function.

11.6 EXECUTION OF EXTERNAL COMPENSATION

External compensation is effective when the Motionpack FD satisfies the following conditions. If the external compensation is not possible, it is necessary to turn off then on the signal after setting becomes possible.

<Conditions>

In AUTO mode and when moving reference pulse is not discharged (at motor stop)

11.7 INTERRUPTION DURING COMPENSATION

When the power supply is turned off during external compensation (DATA SET INT), the data items which have not been compensated for are cleared. The compensation data must be input from the beginning after restart.

11.8 OFFSET VALUE ±MAX. REACH

As a result of the external compensation, when the contents of the relevant offset value register reach or exceed the maximum compensated value that is set to the parameter (Pm21 or Pm23) in absolute value, "offset value ±MAX. reach" signal is turned on without performing compensation.

12. M-NET INTERFACE

I/O signals can be input/output through the serial port by M-NET interface. Since it takes some time to transmit signals through the serial port, signals that require a shorter time can be input/output through the normal I/O channels.

12.1 SPECIFICATIONS OF M-NET INTERFACE

Table 12.1 Specifications of M-NET Interface

Item	Specifications	
Transmission Method	Semi-double method	
Synchronization	Asynchronous method	
Transmission Distance	Up to 100 m (total)	
Bit Configuration	JIS 7-unit system 10-bit (Start 1, data 7, even parity 1, stop 1)	
Parity Check	Vertical parity detection (even parity) Horizontal parity detection (even parity)	
Signal Level	In accordance with EIA standard RS-422	
Transmission Cable	JKEV-SB 0.75 sq.×2* (polyethylene insulation sequence cable with instrumentation paired copper braided shield)	
Internal Consumed Current (Vcc)	+ 5 V, 0.3 A/module	
Transmission Speed	4.8 kbps, 9.6 kbps, 19.2 kbps, 38.4 kbps	
Number of Connected Stations	Slave : 7 stations	
Transmission Mode	T mode	Y mode
Number of Discrete Transmission Points	256 points Input : 126 points Output : 126 points	256 points Input : 126 points Output : 126 points
Number of Register Transmission Points		14 sets Input : 7 sets Output : 7 sets

*: JKEV-SB (transmission cable specifications) is a standard of Japan Cable Industrial Association. The following shows the names for makers :
Sumitomo Denki Kogyo : DPEV-SB
Fujikura Densen : IPEV-SB
Furukawa Denki Kogyo : KPEV-SB

12.2 DATA SIGNAL CONNECTION

(1) Data signal terminals (TM4)

Table 12.2 Data Signal Terminals

Terminal No.	Signal Name	Contents
TM4-1	+D	Data line (active high)
TM4-2	−D	Data line (active low)
TM4-3	SG	Signal line
TM4-4	FG	Frame grounding

(2) Terminators (SW)

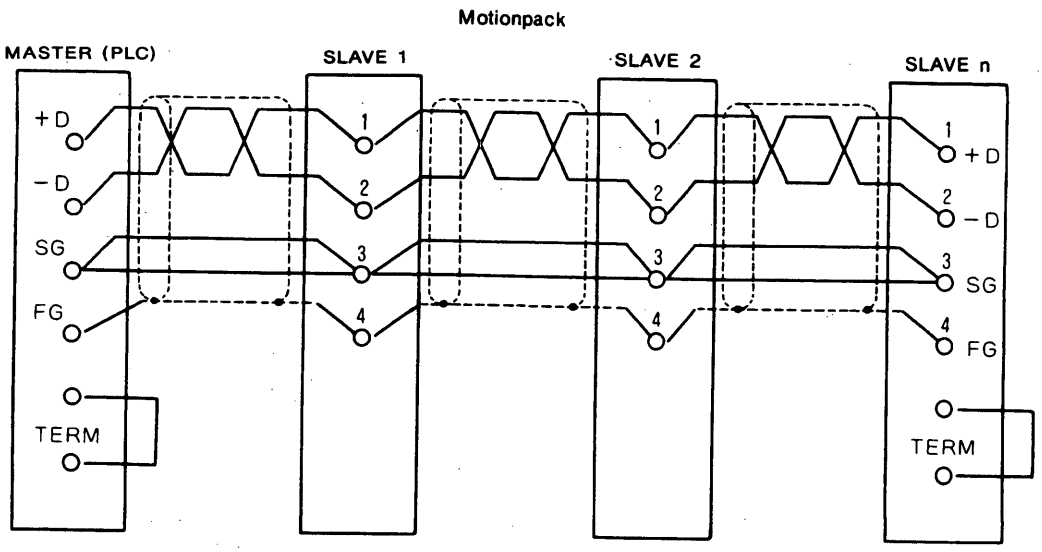
The final slave station must be provided with termination processing of transmission lines by connecting the TERM terminal and 0 terminal.

Table 12.3 Terminators

No.	Signal Name	Contents
1	TERM	Terminator
2	0	0 V
3	Dmy	Dummy

(3) Connection

Connect the stations in the crossover method as shown below and perform TERM terminal processing for the final station.



12.3 SETTING

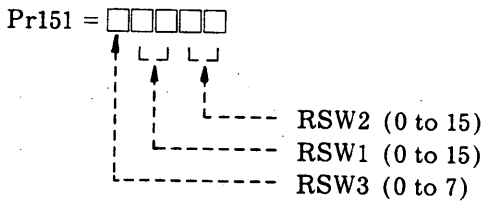
(1) Parameter setting

Set the Motionpack FD parameters.

Table 12.4 Parameter Setting

<table>
<tr><th>Pr. No.</th><th>Name (Range/Unit)</th><th>Change</th><th>Description</th></tr>
<tr><td>Pr150</td><td>M-NET Interface Setting</td><td>P</td><td>
Pr150 = □□
└--- Baud Rate Setting
<table>
<tr><th>Set Value</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>4.8 kbps</td></tr>
<tr><td>1</td><td>9.6 kbps</td></tr>
<tr><td>2</td><td>19.2 kbps</td></tr>
<tr><td>3</td><td>38.4 kbps</td></tr>
</table>
└---- M-NET Interface Setting
<table>
<tr><th>Set Value</th><th>M-NET</th></tr>
<tr><td>0</td><td>Not provided</td></tr>
<tr><td>1</td><td>T-type</td></tr>
<tr><td>2</td><td>Y-type</td></tr>
</table>
EXAMPLE 1) Pr150 = 0 : M-NET not provided
EXAMPLE 2) Pr150 = 21 : Y-type provided
M-NET Baud rate 9.6 kbps
</td></tr>
</table>

Table 12.4 Parameter Setting (Cont'd)

Pr. No.	Name (Range/Unit)	Change	Description
Pr151	Transmission Information Conditions	P	The following three conditions are set to Pr151. (1) No. of discrete input data transmission points (RSW1) (2) No. of discrete output data transmission points (RSW2) (3) No. of register data transmission points (RSW3) Pr151 is expressed in 5-decimal digit. 

	RSW1	RSW2	
No.	DI (Point)	DO (Point)	No. of Connectable Slave Stations 1 2 3 4 5 6 7
0	0	0	
1	8	8	
2	16	16	
3	24	24	
4	32	32	
5	40	40	
6	48	48	
7	56	56	
8	64	64	
9	72	72	
10	80	80	
11	88	88	
12	96	96	
13	104	104	
14	120	120	
15	128	128	

	RSW3		
No.	RI (Point)	RO (Point)	No. of Connectable Slave Stations 1 2 3 4 5 6 7
0	0	0	
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	6	6	
7	7	7	
8	Setting Error		

12.3 SETTING (Cont'd)

(2) Slave setting

Set the slave numbers by the rotary switch on the built-in sequencer board.

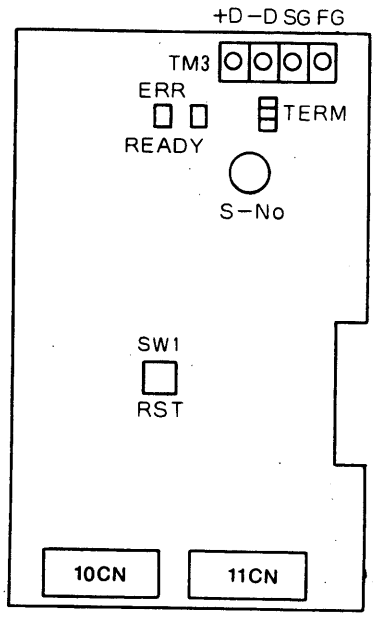
Table 12.5 Slave Setting

RSW No.	Slave No.	RSW No.	Slave No.
0	Default	8	Not used
1	Slave 1	9	Not used
2	Slave 2	A	Not used
3	Slave 3	B	Not used
4	Slave 4	C	Not used
5	Slave 5	D	Not used
6	Slave 6	E	Not used
7	Slave 7	F	Not used

Note : Do not any other numbers than those mentioned above.

12.4 PARTS ARRANGEMENT AND FUNCTIONS

(1) Arrangement



(2) Functions

Table 12.6 Parts Functions

	Type	Name	Contents
Display	LED	READY (green)	Indicates that M-NET interface is in the "READY" status.
		ERR (red)	Indicates that a transmission error occurs in M-NET interface.
Setting SW	Digital Rotary Switch	S-No	Sets a slave number of built-in sequencer [Refer to Par. 12.3 (2).]
RST	Pushbutton SW for reset	SW1	A transmission error of interface between modules can be reset.
SW	Short-circuit SW	TERM/0 V	Termination processing of transmission lines of M-NET interface. When TERM and 0 V are shortcircuited, a termination resistance is inserted.
Terminal/ connector	Terminal	TM3	For M-NET interface
	Connector	10CN 11CN	Connector for extended I/O 10CN : Input signal 11CN : Output signal

13. I/O SIGNAL LIST

13.1 MAIN CONTROLLER FIXED INPUT SIGNALS

Table 13.1 Main Controller Fixed Input Signals

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4000	ZRN	-JS	+JS	JSPD	HANDLE	JOG	PLAY	EDIT	
#4001	PGSL4	PGSL3	PGSL2	PGSL1	MFIN	G34F	SBLK	PGST	
#4002		SVON	ESP6	ESP5	INC8/9	-INC	+INC	ERS	
#4003									
#4004									
#4005									
#4006									
#4007									
#4008									
#4009									

Table 13.1 Main Controller Fixed Input Signals (Cont'd)

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4010	D07	D06	D05	D04	D03	D02	D01	D00	Externally-set data
#4011	D17	D16	D15	D14	D13	D12	D11	D10	Externally-set data
#4012	D27	D26	D25	D24	D23	D22	D21	D20	Externally-set data
#4013	D37	D36	D35	D34	D33	D32	D31	D30	Externally-set data
#4014	N7	N6	N5	N4	N3	N2	N1	N0	Variable numbers
#4015	REQ	IN/OUT	INC/ABS	POS		RESV'D	OFS	REG	Variable type, Control
#4016									
#4017									
#4018									
#4019									

13.1 MAIN CONTROLLER FIXED INPUT SIGNALS (Cont'd)

Table 13.1 Main Controller Fixed Input Signals (Cont'd)

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4020									
#4021									
#4022									
#4023									
#4024									
#4025									
#4026									
#4027									
#4028									
#4029									

13.2 MAIN CONTROLLER FIXED OUTPUT SIGNALS

Table 13.2 Main Controller Fixed Output Signals

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4500	EPALM	G34	OFM	OFR	INCD	STL	SALM	MRDY	
#4501	M57	M56	M55	M54	M53	M52	M51	M50	
#4502	BALM	M30	PSW4	PSW3	PSW2	PSW1	CLD	M58	
#4503									
#4504									
#4505									
#4506									
#4507									
#4508									
#4509									

13.2 MAIN CONTROLLER FIXED OUTPUT SIGNALS (Cont'd)

Table 13.2 Main Controller Fixed Output Signals (Cont'd)

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4510	E07	E06	E05	E04	E03	E02	E01	E00	Output Data
#4511	E17	E16	E15	E14	E13	E12	E11	E10	Output Data
#4512	E27	E26	E25	E24	E23	E22	E21	E20	Output Data
#4513	E37	E36	E35	E34	E33	E32	E31	E30	Output Data
#4514	DACK								Control
#4515									
#4516									
#4517									
#4518									
#4519									

Table 13.2 Main Controller Fixed Output Signals (Cont'd)

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4520									
#4521									
#4522									
#4523									
#4524									
#4525									
#4526									
#4527									
#4528									
#4529									

13.3 BUILT-IN SEQUENCER INPUT SIGNALS

(1) Standard input signals (standard assignment)

Table 13.3 Standard Input Signals

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4200	ZRN	-JS	+JS	JSPD	HANDLE	JOG	PLAY	EDIT	
	3CN-7	3CN-28	3CN-17	3CN-6	3CN-27	3CN-16	3CN-5	3CN-26	
#4201	PGSL4	PGSL3	PGSL2	PGSL1	MFIN	G34F	SBLK	PGST	
	3CN-31	3CN-20	3CN-9	3CN-30	3CN-19	3CN-8	3CN-29	3CN-18	
#4202		SVON	ESP6	ESP5	INC8/9	-INC	+INC	ERS	
	3CN-34	3CN-12	3CN-33	3CN-22	3CN-11	3CN-32	3CN-21	3CN-10	

(2) Extended input

Table 13.4 Extended Input

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4210									
	10CN-7	10CN-28	10CN-17	10CN-6	10CN-27	10CN-16	10CN-5	10CN-26	
#4211									
	10CN-31	10CN-20	10CN-9	10CN-30	10CN-19	10CN-8	10CN-29	10CN-18	
#4212									
	10CN-34	10CN-12	10CN-33	10CN-22	10CN-11	10CN-32	10CN-21	10CN-10	

13.4 BUILT-IN SEQUENCER OUTPUT SIGNALS

(1) Standard input signals (standard assignment)

Table 13.5 Standard Input Signals

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4300	EPALM	G34	OFM	OFR	INCD	STL	SALM	MRDY	
	4CN-7	4CN-28	4CN-17	4CN-6	4CN-27	4CN-16	4CN-5	4CN-26	
#4301	M57	M56	M55	M54	M53	M52	M51	M50	
	4CN-31	4CN-20	4CN-9	4CN-30	4CN-19	4CN-8	4CN-29	4CN-18	
#4302	BALM	M30	PSW4	PSW3	PSW2	PSW1	CLD	M58	
	4CN-34	4CN-12	4CN-33	4CN-22	4CN-11	4CN-32	4CN-21	4CN-10	

(2) Extended input

Table 13.6 Extended Input

Bit / Address	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
#4310									
	11CN-7	11CN-28	11CN-17	11CN-6	11CN-27	11CN-16	11CN-5	11CN-26	
#4311									
	11CN-31	11CN-20	11CN-9	11CN-30	11CN-19	11CN-8	11CN-29	11CN-18	
#4312									
	11CN-34	11CN-12	11CN-33	11CN-22	11CN-11	11CN-32	11CN-21	11CN-10	

13.5 TIMER AND COUNTER LIST

13.5.1 Timer Assignment Table

Table 13.7 Timer Assignment Table

Timer	Address	Application
8 ms timers	#5600	
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	#5610	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
50 ms timers	#5620	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	#5630	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	

Timer	Address	Application
50 ms timers	#5640	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	
100 ms timers	#5650	
	51	
	52	
	53	
	54	
	55	
	56	
	57	
	58	
	59	
	#5660	
	61	
	62	
	63	
	64	
	65	
	66	
	67	
	68	
	69	
	#5670	
	71	
	72	
	73	
	74	
	75	
	76	
	77	
	78	
	79	

Timer	Address	Application
1 s timers	#5680	
	81	
	82	
	83	
	84	
	85	
	86	
	87	
	88	
	89	
1 min timers	#5690	
	91	
	92	
	93	
	94	
	95	
	96	
	97	
	98	
	99	

13.5.2 Counter Assignment Table

Table 13.8 Counter Assignment Table

Address	Application
#5700	
1	
2	
3	
4	
5	
6	
7	
8	
9	
#5710	
11	
12	
13	
14	
15	
16	
17	
18	
19	
#5720	
21	
22	
23	
24	
25	
26	
27	
28	
29	
#5730	
31	
32	
33	
34	
35	
36	
37	
38	
39	

Address	Application
#5740	
41	
42	
43	
44	
45	
46	
47	
48	
49	
#5750	
51	
52	
53	
54	
55	
56	
57	
58	
59	
#5760	
61	
62	
63	
64	
65	
66	
67	
68	
69	
#5770	
71	
72	
73	
74	
75	
76	
77	
78	
79	

Address	Application
#5780	
81	
82	
83	
84	
85	
86	
87	
88	
89	
#5790	
91	
92	
93	
94	
95	
96	
97	
98	
99	

13.6 I/O SIGNAL SPECIFICATIONS

Table 13.9 I/O Signal Specifications

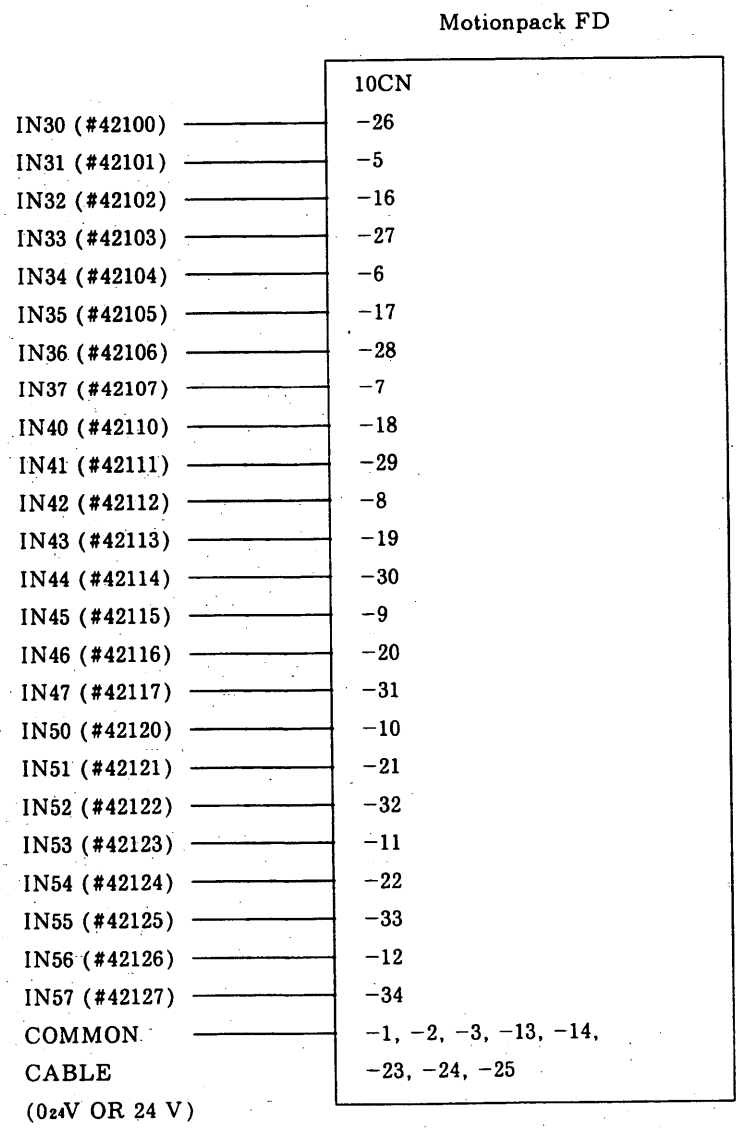
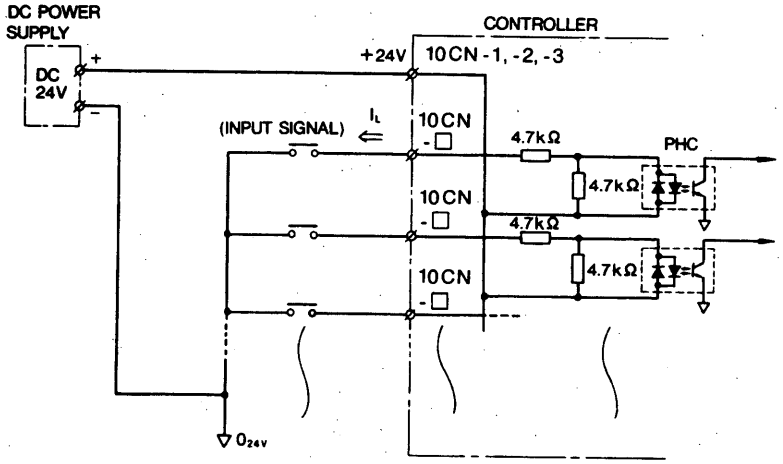
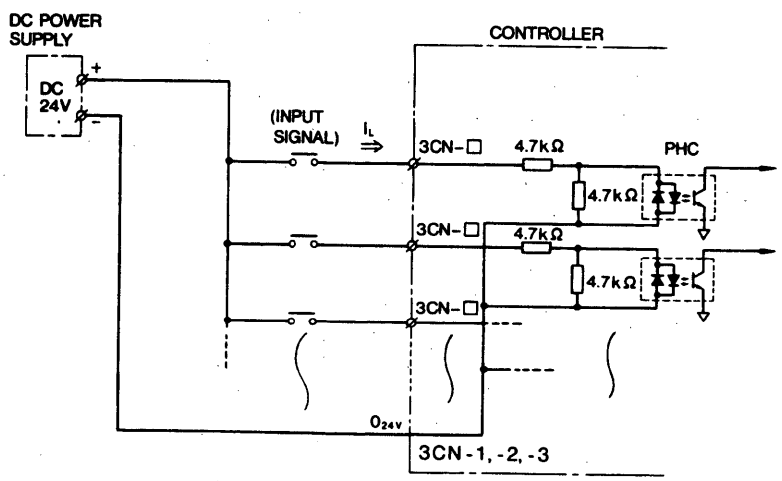
No.	Item	Contents
1	I/O Signal Specifications	Input signal circuit : 24 VDC, 5 mA at ON, 1 μA or less at OFF Output signal circuit : 0 V common, 24 V, common switchable Input signal minimum continuous time : 35 msec Recommended input signal contact : Ratings 30 V, 20 mA class Chattering 5 msec or less
2	Extensive I/O Signals	

Table 13.9 I/O Signal Specifications (Cont'd)

No.	Item	Contents
3	How to Build an Input Circuit 0_{24}V Common (Source Current)	 Same at 10 CN
4	How to Build an Input Circuit + 24 V Common (Sink Current)	 Same at 10 CN

13.6 I/O SIGNAL SPECIFICATIONS (Cont'd)

Table 13.9 I/O Signal Specifications (Cont'd)

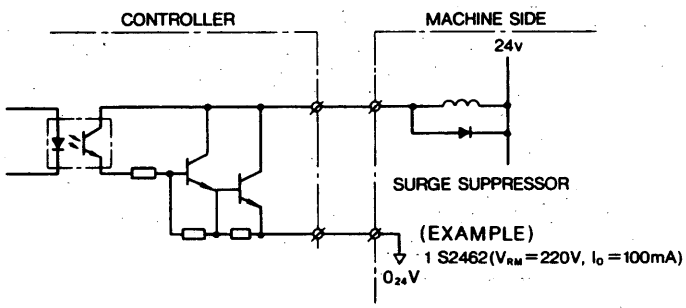
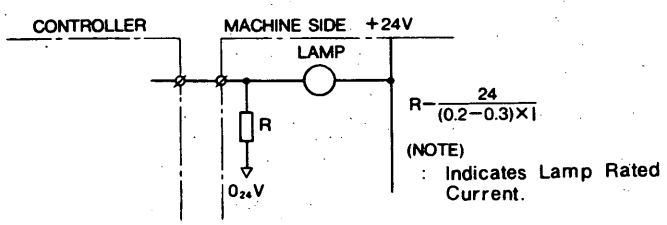
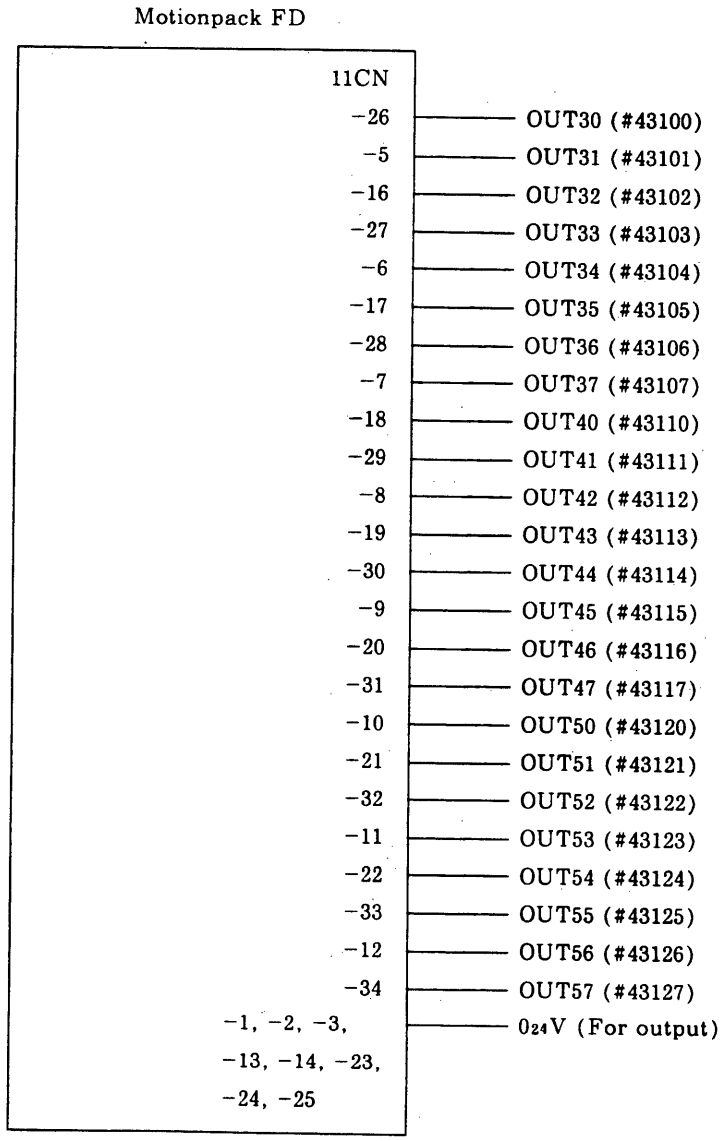
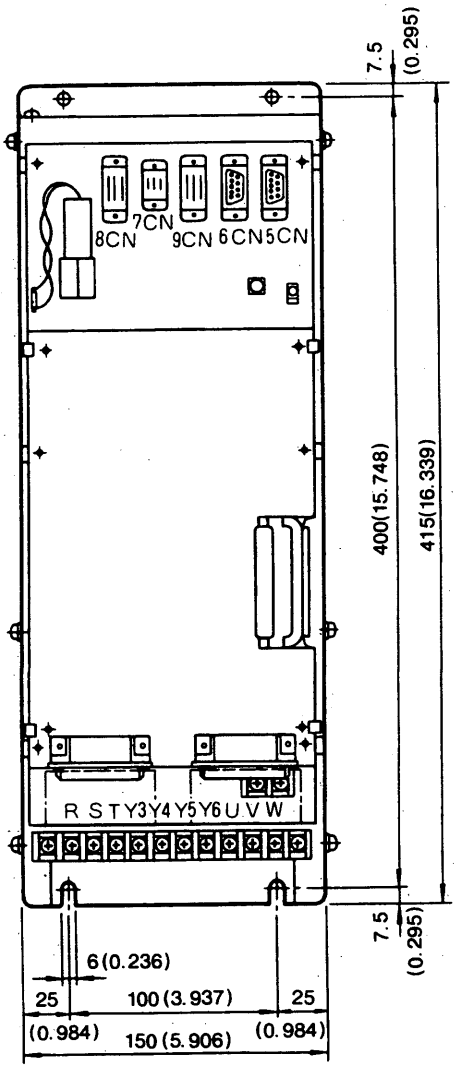
No.	Item	Contents
5	Output Signal Specifications	Output signal capacity : 24 VDC 50 mA or less Output circuit type : No-contact output
		① Insert a surge suppressor within 20 cm from the load when an inductive load (e.g. relay coils) is connected. If the polarities are connected reversely at surge suppressor insertion, the no-contact output circuit is damaged. 
6	Output Circuit Protections	② For lamp load, insert a preheating resistor and current including in-rush current which must be less than the rated current. Current flowing in the lamp by the preheating resistor must be 20 to 30% of the lamp rated current. 

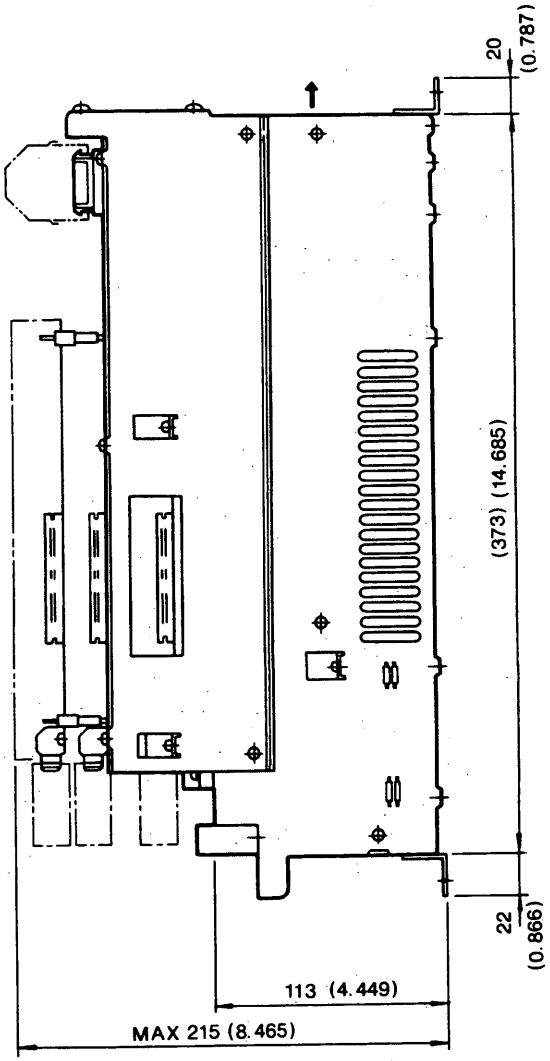
Table 13.9 I/O Signal Specifications (Cont'd)

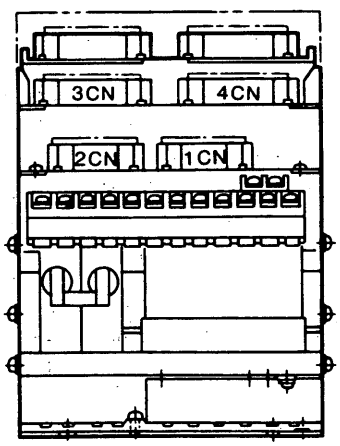
No.	Item	Contents
7	Extensive Output Signals	

14. DIMENSIONS in mm (in inches)

Extension System 1 (CMPR-FD05 to 30B1A)







APPENDIX

This section describes the programs to load/save the ladder programs of the Motionpack FD model 1 built-in sequencer, using the personal computer PC-9801 (NEC) RS-232C port.

A—1 ENVIRONMENT

(1) Personal computer

PC-9801 (NEC)
RS-232C port used

(2) Software (provided by users)

OS: MS-DOS
Utility: MASM. EXE
LINK. EXE
EXE2BIN. EXE

(3) Software (made by YASKAWA)

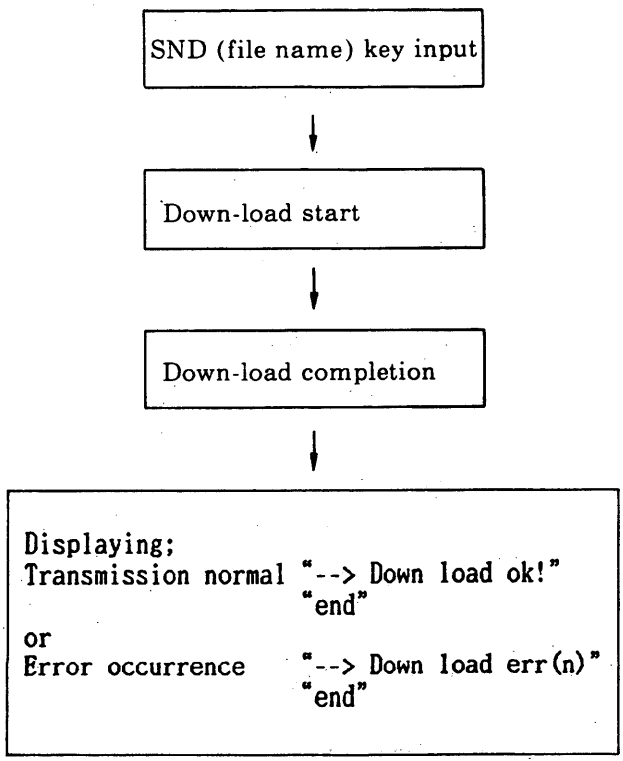
Source file: SND. ASM
RCV1. ASM
RCV2. ASM

Execution file: SND. COM
RCV1. COM
RCV2. COM

NOTE: It is necessary to specify the file (CONFIG. SYS) as "DEVICE = (PASS SPEC.) RSDRV. SYS" when version 3 or above of MS-DOS is used.

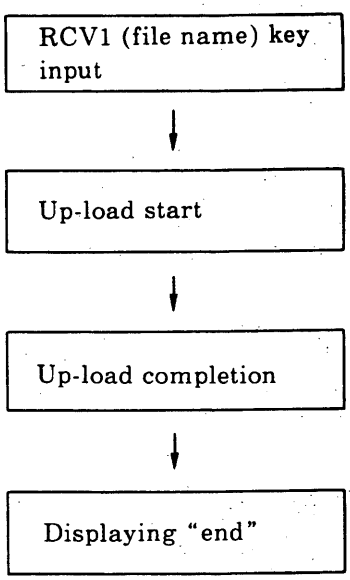
A–2 OPERATION PROCEDURES

(1) Original ladder file down-load (PC-9801 → Motionpack FD)



n: number of error occurrences

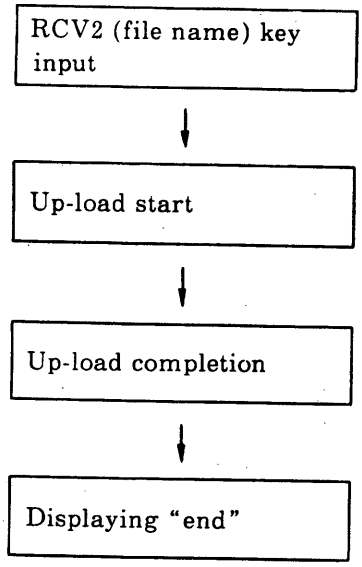
(2) List format file up-load (Motionpack FD → PC-9801)



: Receiving, file creation start

Note : Created files have a prefix "XPR" and a suffix "end". Therefore, when this file is written in to the Motionpack FD, delete the above "XPR" and "end" and add "ORG" to the line next to "%".

(3) Intel hexagon file up-load



: Receiving, file creation start

Note : Created files have a prefix "XPC" and a suffix "end". Therefore, when this file is written in to the Motionpack FD, delete the above "XPC" and "end" and add "ORG" to the line next to "%".

(4) Others

Depress [ESC] key to return to the MS-DOS.

A—3 CONNECTION WITH PERSONAL COMPUTER

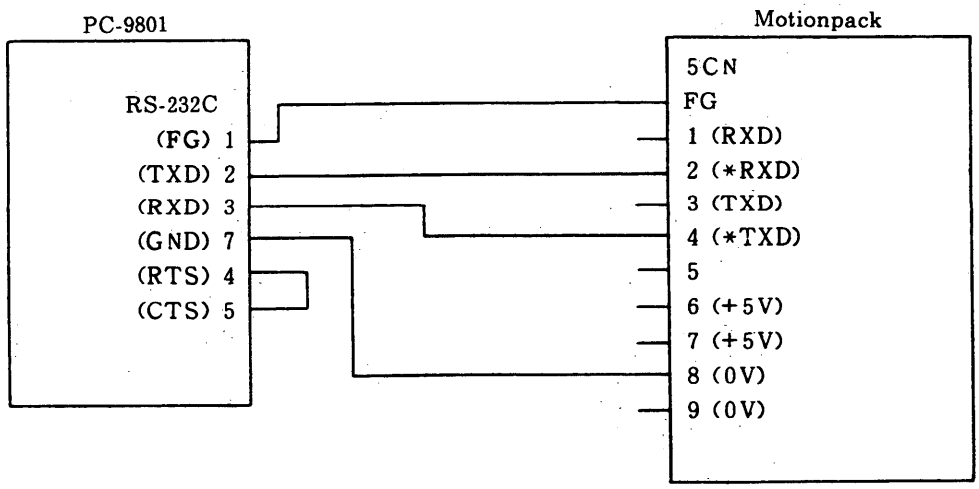
Either 5CN or 6CN of the Motionpack FD is used to connect with the personal computer.

After the power supply is turned on, either 5CN or 6CN that has received a transmission request first opens the port to enable the Motionpack to transmit with the personal computer.

When the power supply is turned on with the exclusive-use programmer connected to 5CN, 5CN is ready to be used. This is because the exclusive-use programmer sends a transmission request automatically after the power supply is turned on. Therefore, when the status where the programmer is used is switched to the personal computer, it is convenient to connect it using 5CN.

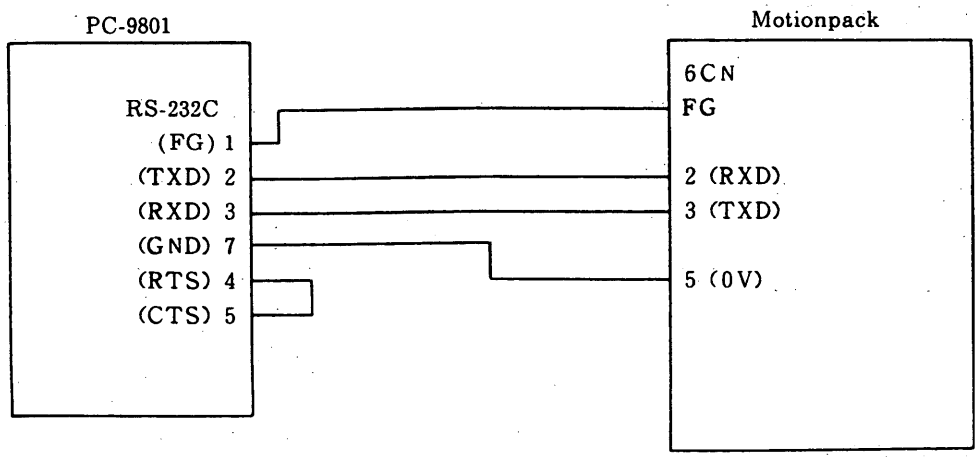
(1) Connection diagrams

(a) When 5CN is used



NOTE: Cable length between the PC-9801 and Motionpack must be 1 m or less.

(b) When 6CN is used



(2) Transmission conditions

Set the personal computer transmission conditions as shown below:

- Transmission speed: 9600 bps
- Bit length: 8 bits
- Stop bit: 1 bit
- Parity: Provided, even-number parity
- XON/OFF control: Provided
- Shift control: Not provided
- Transmission port: RS-232C port

A—4 PERSONAL PROGRAM LIST

The program list is shown below:

A-4 PERSONAL PROGRAM LIST (Cont'd)

```
;
;-------------------------------------------------------------------;
;       S N D _ F I L E                                             ;
;                                   Copyright (c) 1991 by YASKAWA   ;
;-------------------------------------------------------------------;
;
CR                                  EQU     0DH
LF                                  EQU     0AH
FILE_END                            EQU     1AH
RS232C_BIOS_SEGMENT_BASE            EQU     060H
RS232C_BIOS_INT_NUM                 EQU     019H

;Macros-------------------------------------------------------------;
DISPLAY_CHAR MACRO CHARACTOR
        MOV     DL,CHARACTOR
        MOV     AH,02H
        INT     21H
        ENDM

CHECK_KBD_STATUS MACRO
        MOV     DL,0FFH
        MOV     AH,06H
        INT     21H
        ENDM

CHECK_RECEIVE_NUM MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,04H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

RECEIVE_RS232C MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,02H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

TRANSMIT_RS232C MACRO CHAR
        PUSH    DS
        PUSH    AX
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        POP     AX
        MOV     CL,CHAR
        MOV     AH,01H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

;Main Program-------------------------------------------------------;
;-------------------------------------------------------------------;
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE

        ORG     80H
CMDLN   DB      128     DUP (?)

        ORG     100H
MAIN:
```

```
        MOV     BL,BYTE PTR CMDLN+0
        TEST    BL,BL
        JNZ     FILE_SND

        JMP     ERROR
FILE_SND:
        XOR     BH,BH
        ADD     BX,OFFSET CMDLN
        MOV     BYTE PTR [BX+1],00

        MOV     AH,3DH
        MOV     DX,OFFSET CMDLN+2
        MOV     AL,00
        INT     21H
        JNC     HDL_GET

        JMP     ERROR2
HDL_GET:
        MOV     WORD PTR FHDL,AX

        MOV     AX,OFFSET CMD_CHAR
        MOV     WORD PTR RP,AX
        CALL    INIT_RS232C_P0

COM_START:
        check_receive_num
        CMP     CX,0
        JNE     RECEIVE_MOTION
        JMP     TRANSMIT_MOTION

RECEIVE_MOTION:
        LP01:
        PUSH    CX
        receive_rs232c
        display_char    AL
        POP     CX
        LOOP    LP01

        JMP     COM_START

TRANSMIT_MOTION:
        check_kbd_status
        CMP     AL,0H
        JNE     EXIST_KEY_INPUT

SEND_FILE_CHAR:
        TEST    BYTE PTR F_SND_FLG,0FFH
        JNZ     COM_START

        MOV     BX,WORD PTR RP
        MOV     AL,BYTE PTR [BX]
        CMP     AL,00
        JE      SEND_FILE_CHAR2

        INC     WORD PTR RP
        JMP     SNDOUT

SEND_FILE_CHAR2:
        MOV     BX,WORD PTR FHDL
        MOV     AH,3FH
        MOV     DX,OFFSET BUF
        MOV     CX,01
        INT     21H
        CMP     AX,00
        JE      READEND
```

A-4 PERSONAL PROGRAM LIST (Cont'd)

```
        MOV     AL,BYTE PTR BUF

        CMP     AL,FILE_END
        JE      READEND

        JMP     SNDOUT

EXIST_KEY_INPUT:
        CMP     AL,1BH
        JE      EXIT

        CMP     AL,'Q'
        JE      EXIT

        CMP     AL,'q'
        JE      EXIT

        CMP     AL,'¥'
        JNE     COM_START

        CALL    FILE_CLOSE
        MOV     BYTE PTR F_SND_FLG,0FFH
        MOV     AL,03H
SNDOUT: transmit_rs232c AL
        JMP     COM_START

READEND:
        CALL    FILE_CLOSE
        MOV     BYTE PTR F_SND_FLG,0FFH
        JMP     COM_START

ERROR:  MOV     DX,OFFSET ERRORMSG
        JMP     ERROUT

ERROR2: MOV     DX,OFFSET ERRORMSG2
ERROUT: MOV     AH,09
        INT     21H
EXIT:   CALL    FILE_CLOSE
        MOV     AH,4CH
        INT     21H

;Procedures-----------------------------------------------------------
INIT_RS232C_P0 PROC NEAR
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     BX,0068H
        MOV     BYTE PTR DS:[BX],01111101B
        MOV     BYTE PTR DS:[BX+1],00001000B

        MOV     AH,0
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        RET
INIT_RS232C_P0 ENDP

FILE_CLOSE      PROC    NEAR
        TEST    BYTE PTR F_SND_FLG,0FFH
        JNZ     FC_EXIT

        MOV     AH,3EH
        MOV     BX,WORD PTR FHDL
        INT     21H
        MOV     BYTE PTR F_SND_FLG,0FFH
```

```
FC_EXIT:
        RET
FILE_CLOSE      ENDP
;-------------------------------------------------------------------
BUF             DB      0
FHDL            DW      ?
F_SND_FLG       DB      0
ERRORMSG        DB      CR,LF," File not found.",CR,LF,'$'
ERRORMSG2       DB      CR,LF," File open err.",CR,LF,'$'
RP              DW      ?
CMD_CHAR        DB      'X','P','W',CR,00

CODE    ENDS
        END     MAIN
```

A-4 PERSONAL PROGRAM LIST (Cont'd)

```
;-------------------------------------------------------------;
;       R C V I _ F I L E                                     ;
;                                  Copyright (c) 1991 by YASKAWA   ;
;-------------------------------------------------------------;
;Constants----------------------------------------------------;
CR                                  EQU     0DH
LF                                  EQU     0AH
RS232C_BIOS_SEGMENT_BASE            EQU     060H
RS232C_BIOS_INT_NUM                 EQU     019H
;-------------------------------------------------------------
FILE_WRITE      MACRO
        MOV     AH,40H
        MOV     BX,WORD PTR FHDL
        MOV     DX,OFFSET BUF
        MOV     CX,01
        INT     21H
        ENDM

DISPLAY_CHAR MACRO CHARACTOR
        MOV     DL,CHARACTOR
        MOV     AH,02H
        INT     21H
        ENDM

CHECK_KBD_STATUS MACRO
        MOV     DL,0FFH
        MOV     AH,06H
        INT     21H
        ENDM

CHECK_RECEIVE_NUM MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,04H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

RECEIVE_RS232C MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,02H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

TRANSMIT_RS232C MACRO CHAR
        PUSH    DS
        PUSH    AX
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        POP     AX
        MOV     CL,CHAR
        MOV     AH,01H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM
;-------------------------------------------------------------
;-------------------------------------------------------------
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE
```

```
        ORG     80H
CMDLN   DB      128     DUP (?)

        ORG     100H
MAIN:
        MOV     BL,BYTE PTR CMDLN+0
        TEST    BL,BL
        JNZ     FILE_RCV

        JMP     ERROR
FILE_RCV:
        XOR     BH,BH
        ADD     BX,OFFSET CMDLN
        MOV     BYTE PTR [BX+1],00

        MOV     AH,3CH
        MOV     DX,OFFSET CMDLN+2
        MOV     CX,00
        INT     21H
        JNC     HDL_GET

        JMP     ERROR2
HDL_GET:
        MOV     WORD PTR FHDL,AX

        MOV     AX,OFFSET CMD_CHAR
        MOV     WORD PTR RP,AX
        CALL    INIT_RS232C_P0

COM_START:
        check_receive_num
        CMP     CX,0
        JNE     RECEIVE_MOTION
        JMP     TRANSMIT_MOTION

RECEIVE_MOTION:
        LP01:
        PUSH    CX
        receive_rs232c
        MOV     BYTE PTR BUF,AL
        file_write
        display_char    BUF
        POP     CX
        LOOP    LP01
        JMP     COM_START

TRANSMIT_MOTION:
        check_kbd_status
        CMP     AL,00
        JNE     EXIST_KEY_INPUT

        MOV     BX,WORD PTR RP
        MOV     AL,BYTE PTR [BX]
        CMP     AL,00
        JE      COM_START

        INC     WORD PTR RP
        JMP     SNDDAT

EXIST_KEY_INPUT:
        CMP     AL,1BH
        JE      EXIT

        CMP     AL,'Q'
        JE      EXIT
```

A-4 PERSONAL PROGRAM LIST (Cont'd)

```
        CMP     AL,'q'
        JE      EXIT

        CMP     AL,'Y'
        JNE     CON_START

        CALL    FILE_CLOSE
        MOV     AL,03H
SNDDAT: transmit_rs232c AL
        JMP     CON_START

ERROR:  MOV     DX,OFFSET ERRORMSG
        JMP     ERROUT

ERROR2: MOV     DX,OFFSET ERRORMSG2
ERROUT: MOV     AH,09
        INT     21H
        JMP     EX1

EXIT:   CALL    FILE_CLOSE
        MOV     AL,03
        transmit_rs232c AL
EX1:    MOV     AH,4CH
        INT     21H

;Procedures-----------------------------------------------------------
INIT_RS232C_P0 PROC NEAR
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     BX,0068H
        MOV     BYTE PTR DS:[BX],01111101B
        MOV     BYTE PTR DS:[BX+1],00001000B

        MOV     AH,0
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        RET
INIT_RS232C_P0 ENDP

FILE_CLOSE      PROC    NEAR
        TEST    BYTE PTR F_MAK_FLG,0FFH
        JNZ     FC_EXIT

        MOV     BYTE PTR BUF,1AH
        file_write
        MOV     AH,3EH
        MOV     BX,WORD PTR FHDL
        INT     21H
        MOV     BYTE PTR F_MAK_FLG,0FFH
FC_EXIT:
        RET
FILE_CLOSE      ENDP


BUF             DB      0
FHDL            DW      ?
F_MAK_FLG       DB      0
ERRORMSG        DB      CR,LF," File not found.",CR,LF,'$'
ERRORMSG2       DB      CR,LF," File open err.",CR,LF,'$'
RP              DW      ?
CMD_CHAR        DB      'X','P','R',CR,00
CODE    ENDS
        END     MAIN
```

```
;-------------------------------------------------------------;
;       R C V 2 _ F I L E                                     ;
;                                  Copyright (c) 1991 by YASKAWA   ;
;-------------------------------------------------------------;
:Constants-----------------------------------------------------;
CR                                 EQU     0DH
LF                                 EQU     0AH
RS232C_BIOS_SEGMENT_BASE           EQU     060H
RS232C_BIOS_INT_NUM                EQU     019H
;-------------------------------------------------------------
FILE_WRITE      MACRO
        MOV     AH,40H
        MOV     BX,WORD PTR FHDL
        MOV     DX,OFFSET BUF
        MOV     CX,01
        INT     21H
        ENDM

DISPLAY_CHAR MACRO CHARACTOR
        MOV     DL,CHARACTOR
        MOV     AH,02H
        INT     21H
        ENDM

CHECK_KBD_STATUS MACRO
        MOV     DL,0FFH
        MOV     AH,06H
        INT     21H
        ENDM

CHECK_RECEIVE_NUM MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,04H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

RECEIVE_RS232C MACRO
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     AH,02H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM

TRANSMIT_RS232C MACRO CHAR
        PUSH    DS
        PUSH    AX
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        POP     AX
        MOV     CL,CHAR
        MOV     AH,01H
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        ENDM
;-------------------------------------------------------------
;-------------------------------------------------------------
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE
```

A-4 PERSONAL PROGRAM LIST (Cont'd)

```
        ORG     80H
CMDLN   DB      128     DUP (?)

        ORG     100H
MAIN:
        MOV     BL,BYTE PTR CMDLN+0
        TEST    BL,BL
        JNZ     FILE_RCV

        JMP     ERROR
FILE_RCV:
        XOR     BH,BH
        ADD     BX,OFFSET CMDLN
        MOV     BYTE PTR [BX+1],00

        MOV     AH,3CH
        MOV     DX,OFFSET CMDLN+2
        MOV     CX,00
        INT     21H
        JNC     HDL_GET

        JMP     ERROR2
HDL_GET:
        MOV     WORD PTR FHDL,AX

        MOV     AX,OFFSET CMD_CHAR
        MOV     WORD PTR RP,AX
        CALL    INIT_RS232C_P0

COM_START:
        check_receive_num
        CMP     CX,0
        JNE     RECEIVE_MOTION
        JMP     TRANSMIT_MOTION

RECEIVE_MOTION:
        LP01:
        PUSH    CX
        receive_rs232c
        MOV     BYTE PTR BUF,AL
        file_write
        display_char    BUF
        POP     CX
        LOOP    LP01
        JMP     COM_START

TRANSMIT_MOTION:
        check_kbd_status
        CMP     AL,00
        JNE     EXIST_KEY_INPUT

        MOV     BX,WORD PTR RP
        MOV     AL,BYTE PTR [BX]
        CMP     AL,00
        JE      COM_START

        INC     WORD PTR RP
        JMP     SNDDAT

EXIST_KEY_INPUT:
        CMP     AL,1BH
        JE      EXIT

        CMP     AL,'Q'
        JE      EXIT
```

```
        CMP     AL,'q'
        JE      EXIT

        CMP     AL,'V'
        JNE     COM_START

        CALL    FILE_CLOSE
        MOV     AL,03H
SNDDAT: transmit_rs232c AL
        JMP     COM_START

ERROR:  MOV     DX,OFFSET ERRORMSG
        JMP     ERROUT

ERROR2: MOV     DX,OFFSET ERRORMSG2
ERROUT: MOV     AH,09
        INT     21H
        JMP     EX1

EXIT:   CALL    FILE_CLOSE
        MOV     AL,03
        transmit_rs232c AL
EX1:    MOV     AH,4CH
        INT     21H

;Procedures-----------------------------------------------------------
INIT_RS232C_P0 PROC NEAR
        PUSH    DS
        MOV     AX,RS232C_BIOS_SEGMENT_BASE
        MOV     DS,AX
        MOV     BX,0068H
        MOV     BYTE PTR DS:[BX],01111101B
        MOV     BYTE PTR DS:[BX+1],00001000B

        MOV     AH,0
        INT     RS232C_BIOS_INT_NUM
        POP     DS
        RET
INIT_RS232C_P0 ENDP

FILE_CLOSE      PROC    NEAR
        TEST    BYTE PTR F_MAK_FLG,0FFH
        JNZ     FC_EXIT

        MOV     BYTE PTR BUF,1AH
        file_write
        MOV     AH,3EH
        MOV     BX,WORD PTR FHDL
        INT     21H
        MOV     BYTE PTR F_MAK_FLG,0FFH
FC_EXIT:
        RET
FILE_CLOSE      ENDP


BUF             DB      0
FHDL            DW      ?
F_MAK_FLG       DB      0
ERRORMSG        DB      CR,LF," File not found.",CR,LF,'$'
ERRORMSG2       DB      CR,LF," File open err.",CR,LF,'$'
RP              DW      ?
CMD_CHAR        DB      'X','P','C',CR,00
CODE    ENDS
        END     MAIN
```

Motionpack FD Model 1
DESCRIPTIVE INFORMATION

TOKYO OFFICE
New Pier Takeshiba South Tower, 1-16-1, Kaigan, Minatoku, Tokyo 105-6891 Japan
Phone 81-3-5402-4511 Fax 81-3-5402-4580

YASKAWA ELECTRIC AMERICA, INC.
2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone 1-847-887-7000 Fax 1-847-887-7370

MOTOMAN INC. HEADQUARTERS
805 Liberty Lane West Carrollton, OH 45449, U.S.A.
Phone 1-937-847-6200 Fax 1-937-847-6277

YASKAWA ELÉTRICO DO BRASIL COMÉRCIO LTDA.
Avenida Fagundes Filho, 620 Bairro Saude-Sao Pãulo-SP, Brazil CEP: 04304-000
Phone 55-11-5071-2552 Fax 55-11-5581-8795

YASKAWA ELECTRIC EUROPE GmbH
Am Kronberger Hang 2, 65824 Schwalbach, Germany
Phone 49-6196-569-300 Fax 49-6196-888-301

Motoman Robotics Europe AB
Box 504 S38525 Torsås, Sweden
Phone 46-486-48800 Fax 46-486-41410

Motoman Robotec GmbH
Kammerfeldstraβe1, 85391 Allershausen, Germany
Phone 49-8166-900 Fax 49-8166-9039

YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, United Kingdom
Phone 44-1236-735000 Fax 44-1236-458182

YASKAWA ELECTRIC KOREA CORPORATION
Kfpa Bldg #1201, 35-4 Youido-dong, Yeongdungpo-Ku, Seoul 150-010, Korea
Phone 82-2-784-7844 Fax 82-2-784-8495

YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.
151 Lorong Chuan, #04-01, New Tech Park Singapore 556741, Singapore
Phone 65-282-3003 Fax 65-289-3003

YASKAWA ELECTRIC (SHANGHAI) CO., LTD.
4F No.18 Aona Road, Waigaoqiao Free Trade Zone, Pudong New Area, Shanghai 200131, China
Phone 86-21-5866-3470 Fax 86-21-5866-3869

YATEC ENGINEERING CORPORATION
Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan
Phone 886-2-2563-0010 Fax 886-2-2567-4677

YASKAWA ELECTRIC (HK) COMPANY LIMITED
Rm. 2909-10, Hong Kong Plaza, 186-191 Connaught Road West, Hong Kong
Phone 852-2803-2385 Fax 852-2547-5773

BEIJING OFFICE
Room No. 301 Office Building of Beijing International Club, 21
Jianguomenwai Avenue, Beijing 100020, China
Phone 86-10-6532-1850 Fax 86-10-6532-1851

TAIPEI OFFICE
Shen Hsiang Tang Sung Chiang Building 10F 146 Sung Chiang Road, Taipei, Taiwan
Phone 886-2-2563-0010 Fax 886-2-2567-4677

SHANGHAI YASKAWA-TONGJI M & E CO., LTD.
27 Hui He Road Shanghai China 200437
Phone 86-21-6531-4242 Fax 86-21-6553-6060

BEIJING YASKAWA BEIKE AUTOMATION ENGINEERING CO., LTD.
30 Xue Yuan Road, Haidian, Beijing P.R. China Post Code: 100083
Phone 86-10-6233-2782 Fax 86-10-6232-1536

SHOUGANG MOTOMAN ROBOT CO., LTD.
7, Yongchang-North Street, Beijing Economic Technological Investment & Development Area,
Beijing 100076, P.R. China
Phone 86-10-6788-0551 Fax 86-10-6788-2878

YASKAWA ELECTRIC CORPORATION

YASKAWA

SIE-C883-1.2

99-7③
590-206