Foreword

Using the Super SEL Controller Serial Communication Protocol Operating Manual

Thank you very much for purchasing and using Intelligent Actuator products. This manual provides detailed protocol information for our Super SEL controller (version 3.0; 8/30/95). We hope that the explanations and information contained in this manual will give you a more thorough understanding of our controller and, more importantly, allow you to take full advantage of the capabilities of this controller.

This manual is not automatically provided to all IAI customers. It is provided only upon request and is not intended to explain protocols at a beginners level. The manual is written on the assumption that the reader has a background not only in BASIC but also in computer communications and computer programming in general.

Please note the following when using this manual:

- Every effort has been made to ensure the accuracy of the information contained in this publication. However, IAI America, Inc. does not assume liability for the contents of this publication or any damage or injury resulting from the use or misuse of this information.
- IAI America, Inc. reserves the right to make changes to products and/or documentation without notification.
- The user accepts full responsibility for actual implementation.
- This version of the *Super SEL Controller, Serial Communication Protocol Manual* supersedes any earlier versions.
- The communications setting is fixed at 9600 bps, 1 Stop, No Parity (please refer to the specification table provided in this publication).

Real-time interruptions are not possible while the unit is being run by a program. It is better to start a motion control program after communications are completed. Even with a 10 byte inquiry such as a simple version inquiry or output port inquiry, the minimum response will be 34 bytes of data. The amount of time required from inquiry to response is 1.0 x (10+34) = 44msec assuming that the communication has no play. This is determined by CPU processing capability as well as by the 9600 bps specification. The PC side will also require processing time. You can see that actual processing time will be quite time consuming.

As long as you aware of these points, we believe you will obtain good results. IAI's PC Interface Software Manual also incorporates this protocol.

August 1995
IAI America, Inc.

Table of Contents

Table of Contents

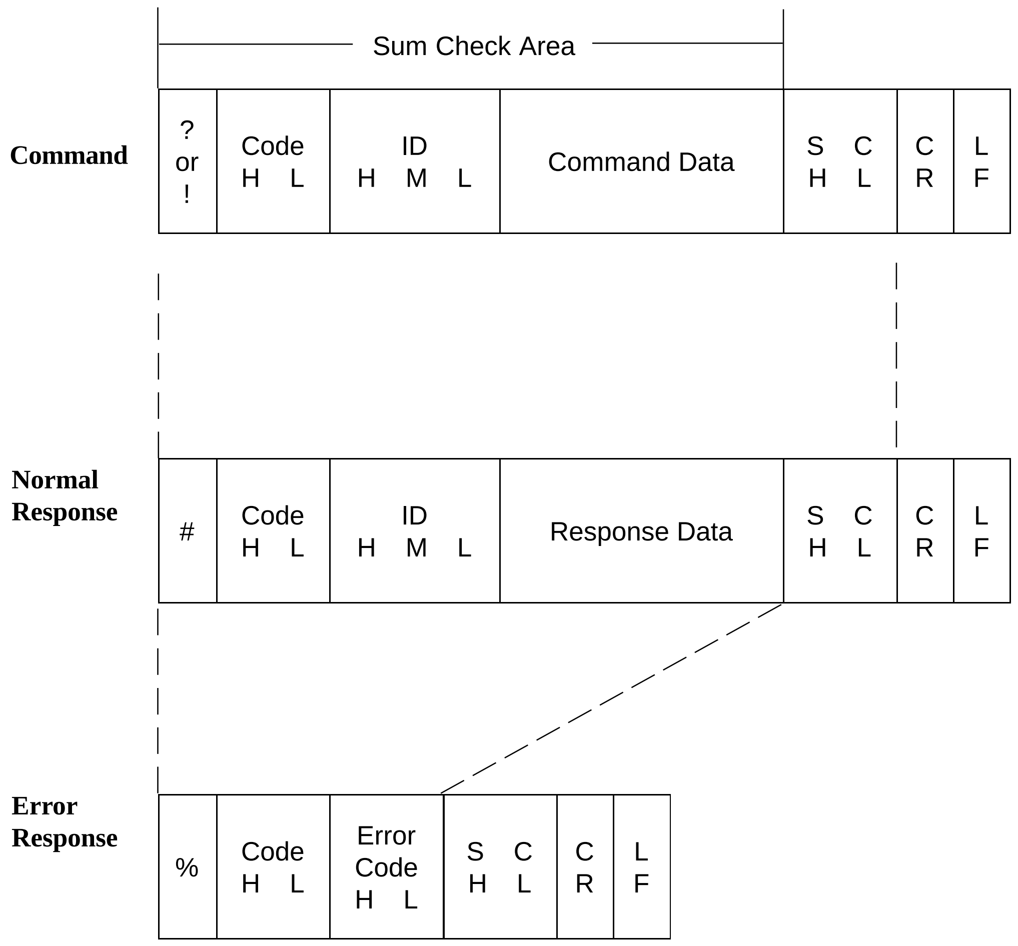
1. Specifications

No.	Item	Explanation
1	Communications	EIA RS232C
2	Baud Rate	9600BPS fixed
3	Cable Distance	Maximum 15m
4	Data Type	START(1)+DATA(8)+STOP(1)+NO PARITY
5	Character	ASCII Code
6	Data Error Check	Sum Check

2. General Conventions

2.1 Command/Response Format

A command is what the controller receives and a response is the answer given to the command. The format of the command and response is shown below.



?(3FH)	: Shows inquiry command	ID(H, M, L)	: Shows type of message
!(21H)	: Shows execution command	Error Code (H,L)	: Error code
#(23H)	: Shows normal response	SC(H,L)	: Check sum
%(25H)	: Shows error response	CR(ODH)	: Shows command/response end
Code(H, L)	: Shows controller code, used when a multi drop application is added	LF(OAH)	: Shows command/response end

2. General Conventions

2.2 Command Response Contents

(1) Axis number, multiple axes, speed, etc.

If there are no special instructions or in the case of a single unit only, indicate with integers. If the number is less than the places designated, fill in with spaces (can use zeros for the upper places). The placement does not matter because the response will be sent with numbers left justified.

Example: When five places are designated, you can write 123 in any of the following ways. (Note: _ indicates a space)

"_ _ 1 2 3"	"_ 1 2 3 _"
"0 0 1 2 3"	"1 2 3 _ _"

(2) Position data, acceleration speed, slice angle, etc.

A fraction in the field means you can indicate it as a decimal number. The fraction indicates the significant digit: 1/10, 1/100 and 1/1000 means you can enter to 1, 2, and 3 decimal places respectively. You do not need to enter anything after the decimal point but entering anything after the significant digit will result in an error. Placement can be done as in (1) above.

(3a) Axis Pattern, Input/Output Port, Flag Ports...etc.

7 ~ 0 written in the annotation indicates the value is a hexadecimal. When you represent this value as a binary number, each bit represents a different item. 0 or 1 in the bit indicates whether that item is valid or invalid.

Example: A 2,4,5,7 axis pattern would be indicated as 5A:

Item	Content	Binary Number	Hexadecimal
1	Axis 2	0 0 0 0 0 0 1 0	02
3	Axis 4	0 0 0 0 1 0 0 0	08
4	Axis 5	0 0 0 1 0 0 0 0	10
6	Axis 7	+0 1 0 0 0 0 0 0	+ 40
		0 1 0 1 1 0 1 0	5A

(3b) Axis Selection

Selection of an axis is specified by either "1" or "0"

Type A

Axis No.	θ	Z	Y	X
Used	1	1	1	1
Not Used	0	0	0	0

Type B

Axis No.	8	7	6	5	4	3	2	1
Used	1	1	1	1	1	1	1	1
Not Used	0	0	0	0	0	0	0	0

Example

If Axis 1 and Axis 2 are in use, then this is signified by ...

```
0  0  0  0  0  0  1  1
                  |  |
       Axis 2 ----┘  └---- Axis 1
```

Example

If Axis 1 and Axis 2 are in use, then this is signified by ...

```
(1 1)            =          (3)16
 | └──── axis 1
 └────── axis 2
```

Axis pattern is used to designate more than one axis at the same time.

2.3 Check Sum

This is used to confirm whether the actual (received) protocol response corresponds to the calculated (expected) response. The comparison of the received and calculated check sums must be done in the user's program. For the check sum, convert each ASCII character in the normal protocol response (not an error response) into a hexadecimal, add them together and use the two least significant bytes as the check sum.

Example #1:

Executable command: !99EXT0186
Response from Super SEL: #99EXT86

	Hex	Dec
#	23	35
9	39	57
9	39	57
E	45	69
X	58	88
T	54	84
	186	390

Example #2:

Inquiry command: ?99IPO3F
Response from Super SEL: #99IPO20003F

	Hex	Dec
#	23	35
9	39	57
9	39	57
I	49	73
P	50	80
O	4F	79
2	32	50
0	30	48
0	30	48
0	30	48
	23F	575

You can bypass the check sum by using @@ in place of the 2 bytes reserved for the sum check. For example, the above inquiry command would become ?99IPO@@

2.4 Important Notes

(1) The controller ID code (bytes 2 & 3 of the ASCII string) was intended to be used for a multi drop network but this function was never implemented. A default value of 99 should be used. If this multi drop function is needed please refer to the SelNet description in the Sel G operating manual.

(2) All protocol commands are case sensitive.

(3) An axis number is not the same as an axis pattern. An axis number pertains to a single axis (for example, axis#1). An axis pattern pertains to multiple axes (for example, if axes 1 & 2 are selected, then the axis pattern is $(3)_{16}$.

(4) For commands that require a value with a decimal point, the decimal point is considered a byte.

3. Inquiry Text

No.	Text Name	ID	Description
1	Test Call Inquiry	TST	Asks test call
2	Version Inquiry	VER	Asks version date
3	Input Port Inquiry	INP	Asks input port
4	Output Port Inquiry	OUT	Asks output port
5	Flag Inquiry	FLG	Asks internal flags
6	Available Memory Inquiry	RMS	Asks available memory
7	Program Parameter Inquiry	IPG	Asks program parameters
8	Program Status Inquiry	PRG	Asks program status
9	Program Step Inquiry	STP	Asks step contents
10	SIO Parameter Inquiry	ISI	Asks SIO parameters
11	Point Parameter Inquiry	IPO	Asks point parameters
12	Servo Parameter Inquiry	ISV	Asks servo parameters
13	Servo Parameter Inquiry By Axis	IAG	Asks servo parameters by axis
14	Homing Parameter Inquiry by Axis	IAH	Asks homing parameters by axis
15	Motor Parameter Inquiry by Axis	IAM	Asks motor parameters by axis
16	Circular Parameter Inquiry	ICR	Asks circular parameters
17	Axis Status Inquiry	STA	Asks axis status
18	Task Status Inquiry	TSK	Asks task status
19	Step Quantity Inquiry	DIR	Asks number of program steps
20	Point Data Inquiry	POS	Asks point data
21	Error Message Inquiry	MSG	Asks error message
22	Variable Inquiry	VAR	Asks variable

3. Inquiry Text

3.1 Test Call

(1) Function
Executes communication test. The same data as the command is transmitted back.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
?	Code H	L	T	S	T	Any Letters (10 characters)										SH	CL	CR	LF

(3) Response

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
#	Code H	L	T	S	T	Same characters as the command										SH	CL	CR	LF

(4) Error Response
General error

Example

Command: ?99TST0123456789@@

Response: #99TST0123456789@@

3. Inquiry Text

3.2 Version Inquiry

(1) Function
Inquires about the ROM stamp of the controller.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1
?	Code H L		V E R			Type	Axis No.	S C H L		C R	L F

Type: M : Main / S : Servo

Axis No.: 0 : 1 Axis ~ 7 : 8 Axis

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9
#	Code H L		V E R			Type	Axis No.	Version											

Axis No.: 0 : 1 Axis ~ 7 : 8 Axis

Type: M : Main / S : Servo

20	1	2	3	4	5	6	7	8	9	30	1	2	3	4	5	6	7	8	9
Month		/	Day		/	Year		Hour		:	Minute		:	Second		S C H L		C R	L F

(4) Error Response
① General error
② Axis error

Example

Command: ?99VERM 0@@
Response: #99VERM0IAMain V2.3003/27/9510:31:39@@

3. Inquiry Text

3.3 Input Port Inquiry

(1) Function
Inquires about the input port.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I N P			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	0	10	70	1	2	3	4	5	6	7	8	9	80	1
#	Code H L		I N P			1 7~0		2 15~8		3 23~16		33		34		35		36		S C H L		C R	L F

7	6	5	4	3	2	1	0
General input	General input	General input	System reserve	System reserve	Emergency stop	General input	External drive input

(4) Error Response
General error

Example

When input ports 2 (E-Stop), 6 & 7 are on, the input port inquiry is as follows,

Command: ?99INP@@
Response: #99INPC40000$\underbrace{\text{FFF...}\quad\text{(66 F's)}}$@@

$$(\overbrace{1\ 1\ 0\ 0}^{\text{byte 6}}\ \overbrace{0\ 1\ 0\ 0}^{\text{byte 7}})_2 \qquad = \qquad (C4)_{16}$$

Port #7 Port #0

3.4 Output Port Inquiry

(1) Function
Inquires about the output port.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		O U T			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	0	10	≈	70	1	2	3	4	5	6	7	8	9	80	1
#	Code H L		O U T			1 307~300		2 315~308		3 323~316		≈	33		34		35		36		S C H L		C R	L F

7	6	5	4	3	2	1	0
General output	General output	General output	General output	General output	General output	Ready	Alarm

(4) Error Response
General error

Example

When output port 301 (ready) is on, then the response is as follows,

Command: ?99OUT@@
Response: #99OUT02000... (70 0's) @@

```
          byte 6  byte 7
        (0 0 0 0 0 0 1 0)2          =       (02)16
         └Port 307 │   └Port 300
                   └────Ready output
```

3. Inquiry Text

3.5 Flag Inquiry

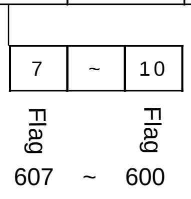
(1) Function
Inquires about the flag.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		F L G			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	0	10	≀	70	1	2	3	4	5	6	7	8	9	80	1
#	Code H L		F L G			1 607~600		2 615~608		3 623~616		≀	33		34		35		36		S C H L		C R	L F

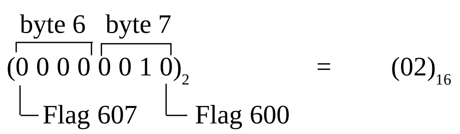


(4) Error Response
General error

Example

When flag 601 is on, then the response is as follows,

Command: ?99FLG@@
Response: #99FLG02000... (70 0's) @@



3.6 Available Memory Inquiry

(1) Function
Inquires about the available memory of the program.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		R M S			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3
#	Code H L		R M S			Remaining Step Quantity				S C H L		C R	L F

(4) Error Response
General error

Example

Command: ?99RMS@@
Response: #99RMS2608@@

3.7 Program Parameter Inquiry

(1) Function
Inquires about the program parameters.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I P G			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5
#	Code H L		I P G			Auto Start Prog#		E-Stop Prog #		No. of Programs		No. of tasks		No. of program steps				Time Slice Value (1/100 sec)				S C H L		C R	L F

(4) Error Response
General error

Example

Command: ?99IPG@@
Response: #99IPG0 0 641630000.01@@

3.8 Program Status Inquiry

(1) Function
Inquires about the program status.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
?	Code H L		P R G			Program #		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8
#	Code H L		P R G			Prog #		Status	Cont. Error Code		Step #				S C H L		C R	L F

Status:
0: Stop
1: Executing

(4) Error Response
① General error
② Axis number error

Example

Command: ?99PRG01@@
Response: #99PRG010000 @@

3. Inquiry Text

3.9 Program Step Content Inquiry

(1) Function
Inquires about program step.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
?	Code H L		S T P			Program Number		Step Number				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
#	Code H L		STP			Program Number		Step Number				A / D	Condition 1				Command				Operand #1					
													N	I/O Flag												

7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
Operand #1			Operand #2									Post			Comment											

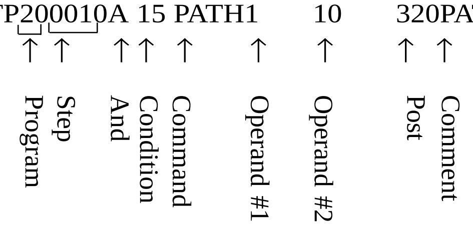
4	5	6	7	8	9	0	1	2	3
(18 digits)						S C H L		C R	L F

* When the step number is zero, that is the step currently being executed.

(4) Error Response
① General error
② Program number error
③ Step number error

Example

Command: ?99STP200010@@
Response: #99STP200010A 15 PATH1 10 320PATH/TURN ON 320 @@



3.10 SIO Parameter Inquiry

(1) Function
Inquires about the SIO parameter.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I S I			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4
#	Code H L		I S I			Time Out	Baud Rate	Bit Length	Parity	Stop Bit	S C H L		C R	L F

0 : 1 Bit
1 : 2 Bit

0 : None
1 : Even
2 : Odd

0 : 8 Bit
1 : 7 Bit

0 : 1200
1 : 2400
2 : 4800
3 : 9600
4 : 19200

(4) Error Response
General error

Example

Command: ?99ISI@@
Response: #99ISI03000@@

3.11 Point Parameter Inquiry

(1) Function
Inquires about the point parameters.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I P O			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3
#	Code H L		I P O			Point Quantity				S C H L		C R	L F

(4) Error Response
General error

Example

Command: ?99IPO@@
Response: #99IPO02000@@

3.12 Servo Parameter Inquiry

(1) Function
Inquires about the servo parameters.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I S V			S C H L		C R	L F

(2) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
#	Code H L		I S V			No. of Axes	Numerator				Denominator				Override				Operation Velocity (mm/sec)				Maximum Velocity (mm/sec)			

7	8	9	30	1	2	3	4	5	6	7	8
Acceleration (1/100g)				Maximum Acceleration (1/100g)				S C H L		C R	L F

(4) Error Response
General error

Example

Command: ?99ISV@@
Response: #99ISV21 1 100 100 30000.3020.0@@

3.13 Servo Parameter Inquiry by Axis

(1) Function
Inquires about the servo parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10
?	Code H L		I A G			Axis #	S C H L		C R	L F

Axis #: 0 : 1 Axis ~ 7 : 8 Axis

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7
#	Code H L		I A G			Axis No.	Axis Name	Service frequency (times/sec)				Numerator				Denominator				Override				Jog Velocity (mm/sec)			

8	9	30	1	2	3	4	5	6	7	8	9	40	1	2	3	4	5	6	7	8	9	50	1	2
Position End Band (Pulse)				Soft Limit (+) (1/1000mm)				Soft Limit (-) (1/1000mm)				Soft Limit Offset (1/1000mm)					Acceleration (1/100g)				S C H L		C R	L F

(4) Error Response
① General error
② Axis number error

Example

Command: ?99IAG1@@
Response: #99IAG12400 1 1 100 30 20 150 0 1.6000.30@@

3. Inquiry Text

3.14 Homing Parameter Inquiry by Axis

(1) Function
Inquires about the homing parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10
?	Code H L		I A H			Axis #	S C H L		C R	L F

0 : 1 Axis
~
7 : 8 Axis

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7
#	Code H L		I A H			Axis No.	Direction	Type	Sequence	Limit Polarity	Z Axis Edge	Creep Velocity (mm/sec)				Position-end Search Velocity (mm/sec)				Z Axis Pulse Search Velocity (mm/sec)				Offset Moving Length (mm)			

8	9	30	1	2	3	4	5	6	7	8	9
Home Deviation (Pulse)				Home Current Limit				S C H L		C R	L F

(4) Error Response
① General error
② Axis number error

Example

Command: ?99IAH1@@
Response: #99IAH1001110   10 4   0   480 55 @@

3. Inquiry Text

3.15 Motor Parameter Inquiry by Axis

(1) Function
Inquires about the motor parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10
?	Code H L		I A M			Axis No.	S C H L		C R	L F

0 : 1 Axis
~
7 : 8 Axis

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
#	Code H L		I A M			Axis No.	Maximum Motor rpm				Encoder Pulse				Screen Lead (mm)				Encoder Multiply				Position Gain			

7	8	9	30	1	2	3	4	5	6	7	8	9	40	1	2	3	4	5	6	7	8	9	50
Speed Gain				Feed Forward Gain				Integral Gain				Total Gain				Integral Voltage Limit				Over Speed Constant			

1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1
Error Range (pulse)				Maximum Motor Current				Brake Time (1/100 sec)				Motor Overload Minimum Limit					S C H L		C R	L F

(4) Error Response

① General error
② Axis number error

Example

```
Command:   ?99IAM1@@
Response:  #99IAM14000384 16  4   30  80  0   15  60  60  400 384090  0.1023600@@
```

3.16 Circular Parameter Inquiry

(1) Function
Inquires about the circular parameter.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		I C R			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7
#	Code H L		I C R			Time Slice (1/10 degrees)					Speed Increment (mm/sec)			S C H L		C R	L F

(4) Error Response

① General Error

Example

Command: ?99ICR@@
Response: #99ICR15.0 0 @@

3. Inquiry Text

3.17 Axis Status Inquiry

(1) Function
Inquires about the axis status.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		S T A			S H	C L	C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	≀				
#	Code H L		S T A			Axis Qty	1 Axis															S H	C L	C R	L F
							Servo	Home	Move	Error Code		Current Position (1/1000mm)									≀				

Move:
0: Stop
1: Moving

Home:
0: Homing incomplete
1: Homing complete

Servo:
0: Servo off
1: Servo on

Maximum length:
7 + no. of axes (8) x 14 + 4

(4) Error Response
① General error
② Axis number error

Example

Command: ?99STA@@
Response: #99STA200000150.000 00000150.000 @@

3. Inquiry Text

3.18 Task Status Inquiry

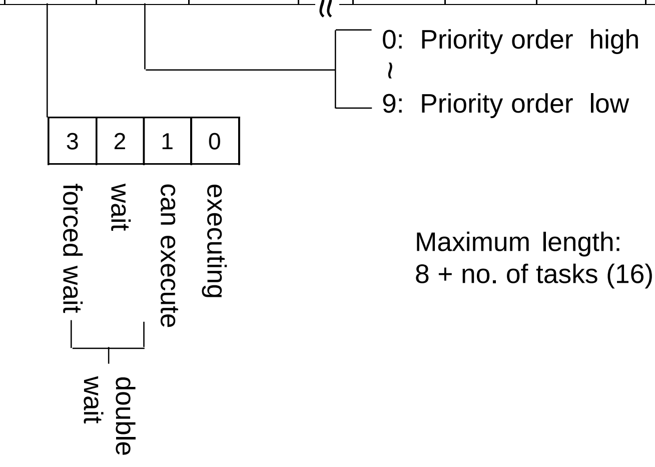
(1) Function
Inquires about the task status.

(2) Command

0	1	2	3	4	5	6	7	8	9
?	Code H L		T S K			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1						
#	Code H L		T S K			Task Qty		Task 1			≈	Task 16			S C H L	C R	L F
								Status	Level	Prog #		Status	Level	Prog #			



Maximum length:
8 + no. of tasks (16) x 4 + 4

(4) Error Response
① General error

Example

Command: ?99TSK@@
Response: #99TSK16090 090 090 090 090 090 090 090 090 090 090 090 090 090 090 090 @@

3.19 Step Quantity Inquiry

(1) Function
Inquires about the step quantity specified by the program number.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
?	Code H L		D I R			Program #		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5
#	Code H L		D I R			Program Number		Number of Steps				S C H L		C R	L F

(4) Error Response
① General error
② Program number error

Example

Command: ?99DIR01@@
Response: #99DIR0111 @@

3. Inquiry Text

3.20 Point Data Inquiry

(1) Function
Inquires about the point data which is specified by the point number.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3
?	Code H L		P O S			Point #				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9
#	Code H L		P O S			Point Number				Axis Pattern		Acceleration (1/100g)				Velocity (mm/sec)			

7	~	0

Axis 8 ~ Axis 1

20	1	2	3	4	5	6	7	8	≀≀			
Position Data (1/1000mm)									Position Data	S C H L	C R	L F

Maximum length:
19 + no. of axes (8) x 9 + 4 bytes

(4) Error Response
① General error
② Program number error
③ Axis error
④ Data error

Example

Command: ?99POS0001@@
Response: #99POS0001030.30100 25.000   0   @@

3. Inquiry Text

3.21 Error Message Inquiry

(1) Function
Inquires about the error message.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
?	Code H L		M S G			Error Code		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3
#	Code H L		M S G			Cont. Error Code		Error Message (16 characters)															

4	5	6	7
S C H L		C R	L F

(4) Error Response
1) General error
2) Axis number error

Example

Command: ?99MSGA3@@
Response: #99MSGA3DEV_ERR @@

3.22 Variable Inquiry

(1) Function
Inquires about the variable.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7
?	Code H	L	V	A	R	Prog #		Starting #			Ending #			S H	C L	C R	L F

Prog #:
00: Global variable only
01: For every program
~ : Local variable possible

Starting # / Ending #:
1
~ : Local variable
99
200
~ : Global variable
299

(3) Response

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5					
#	Code H	L	V	A	R	Prog #		Starting Variable								Variable	S H	C L	C R	L F

(4) Error Response
① General error
② Axis number error
③ No program error

Example

Command: ?99VAR00200202@@
Response: #99VAR00000000680000006800000069@@

4. Execution Text

No.	Text Name	ID	Explanation
1	Servo ON/OFF	SRV	Turns servo on and off
2	Homing	HOM	Executes homing
3	Move to Specified Position	MOV	Moves actuator to specified position
4	Jog Move	JOG	Executes Jog move
5	Point Number Move	PMV	Moves to position specified by point number
6	Erase Program	PDL	Erases program specified by number
7	Add Program Step	APD	Inserts single steps
8	Change Program	ALT	Changes single steps
9	Execute Program	RUN	Executes specified program
10	Stop Program	EXT	Stops program being executed
11	Insert Program Step	INS	Inserts 1 line before specified step
12	Reorganize Program Memory	PRS	Reorganizes program
13	Erase Program Step	DEL	Erases specific step of a specific program
14	Set Point Data	PSE	Sets data at specified point number
15	Clear Point Data	CLE	Clears point data
16	Copy Point Data	CPY	Copies point data
17	Shift Point Data	SFT	Moves point data
18	Set Servo Parameters	RSV	Sets servo parameters
19	Set Servo Parameters by Axis	RAG	Sets servo parameters by axis
20	Set Homing Parameters by Axis	RAH	Sets homing parameters by axis
21	Set Motor Parameters by Axis	RAM	Sets motor parameters by axis
22	Set Arc Parameters	RCR	Sets arc parameters
23	Slow To A Stop	HLT	Slows actuator to a halt
24	Set Output Port	OTS	Sets output port
25	Set Global Flags	GFS	Sets global flags
26	Clear Memory	RCL	Clears memory
27	Reset	RST	Resets driver

4. Execution Text

4.1 Servo ON/OFF

(1) Function
Inquires about the variable.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2
!	Code H L		S R V			Axis Pattern		ON OFF	S C H L		C R	L F

ON OFF:
0: Off
1: On

Axis Pattern:

7	~	0

Axis 8 ~ Axis 1

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		S R V			S C H L		C R	L F

(4) Error Response
① General error
② Axis error

Example

Command: !99SRV031@@
Response: #99SRV@@

4. Execution Text

4.2 Homing

(1) Function
Initiates homing sequence. Servo ON function also included.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3
	Code					Axis		Vel		S	C	C	L
!	H	L	H	O	M	Pattern		(mm/sec)		H	L	R	F

Parameter goes into effect when this is zero

7	~	0

Axis 8 ~ Axis 1

(3) Response

0	1	2	3	4	5	6	7	8	9
	Code					S	C	C	L
#	H	L	H	O	M	H	L	R	F

(4) Error Response
① General error
② Axis error

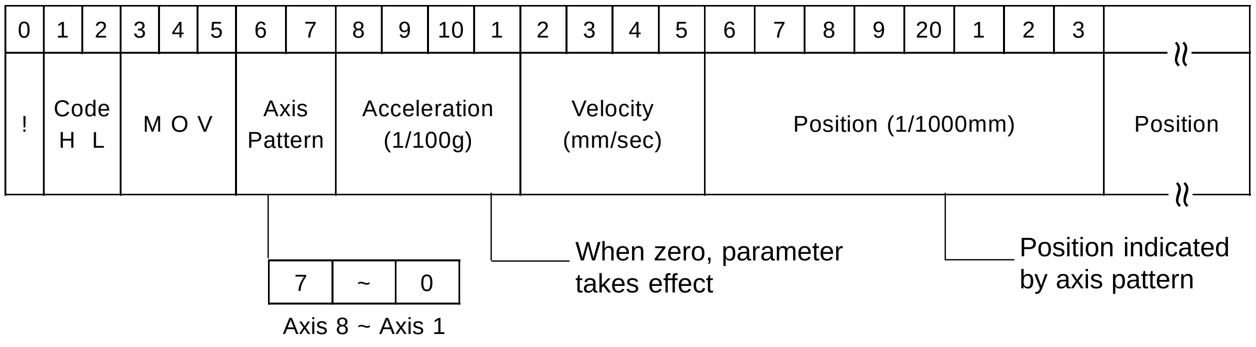
Example

Command: !99HOM0300@@
Response: #99HOM@@

4.3 Move To Specified Position

(1) Function
Moves actuator to designated position.

(2) Command



0	1 2	3 4 5	6 7	8 9 10 1	2 3 4 5	6 7 8 9 20 1 2 3	
!	Code H L	M O V	Axis Pattern	Acceleration (1/100g)	Velocity (mm/sec)	Position (1/1000mm)	Position

7 ~ 0
Axis 8 ~ Axis 1
When zero, parameter takes effect
Position indicated by axis pattern

S C H L	C R	L F

Maximum length: 15 + No. of axes (8) x 8 + 4

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H	L	M	O	V	S H	C L	C R	L F

(4) Error Response
① General error
② Axis error
③ Acceleration error
④ Speed error
⑤ Position error

Example

Command: !99MOV030000020000050.0000075.00@@
Response: #99MOV@@

4. Execution Text

4.4 Jog Move

(1) Function
Executes Jog move. When there is no deceleration stop command, it stops at the soft limit.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20
!	Code H L		J O G			Axis Pattern		Acceleration (1/100g)				Velocity (mm/sec)				Direction	S C H L		C R	L F

Axis Pattern:

7	~	0

Axis 8 ~ Axis 1

Direction:
0: Home side
1: Side opposite home

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		J O G			S C H L		C R	L F

(4) Error Response
① General error
② Axis error
③ Acceleration error
④ Speed error

Example

Command: !99JOG030.3000501@@
Response: #99JOG@@

4.5 Point Number Move

(1) Function
Moves the actuator to the position designated by the assigned point number.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3
!	Code H L		P	M	V	Axis Pattern		Acceleration (1/100g)				Velocity (mm/sec)				Point #				S H	C L	C R	L F

Velocity: Follows parameters when zero
Acceleration: Follows parameters when zero
Axis Pattern: Follows data when there is no axis pattern

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		P	M	V	S H	C L	C R	L F

(4) Error Response
① General error
② Axis error
③ Acceleration error
④ Speed error
⑤ Point number error

Example

Command: !99PMV03000002000001@@
Response: #99PMV@@

4.6 Erase Program

(1) Function
Deletes the specified program number.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1
!	Code H L		P D L			Program Number		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		P D L			S C H L		C R	L F

(4) Error Response
① General error
② Program number error

Example

Command: !99PDL01@@
Response: #99PDL@@

4.7 Add Program Step

(1) Function
Adds a program step. This is used to make additions after each single step. Use the INS command to insert additions during the middle of a step.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
!	Code H L		A P D			Program Number		Step Number				A / O	Condition 1				Command				Operand #1					
													N	I/O Flag												

7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
Operand #1			Operand #2									Post			Comment											

4	5	6	7	8	9	0	1	2	3
(18 characters)						S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		A P D			S C H L		C R	L F

(4) Error Response
① General error
② Program number error
③ Data error

Example

```
Command:   !99APD200010A 15 PATH1      10      320PATH/TURN ON 320 @@
Response:  #99APD@@
```

4. Execution Text

4.8 Change Program

(1) Function
Changes program step.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
!	Code H L		A L T			Program Number		Step Number				A / O	Condition 1				Command				Operand #1					
													N	I/O Flag												

7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
Operand #1			Operand #2									Post			Comment											

4	5	6	7	8	9	0	1	2	3
(18 characters)						S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		A L T			S C H L		C R	L F

(4) Error Response
① General error
② Program number error
③ Step number error
④ Data error

Example

Command: !99ALT200010A 15 PATH1 10 320PATH/TURN ON 320 @@
Response: #99ALT@@

4.9 Execute Program

(1) Function
Executes the designated program.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
!	Code H L		R U N			Prog #		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R U N			S C H L		C R	L F

(4) Error Response
① General error
② Program number error
③ Error during program execution

Example

Command: !99RUN01@@
Response: #99RUN@@

4.10 Stop Program

(1) Function
Stops execution of the program.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
!	Code H L		E X T			Prog #		S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		E X T			S C H L		C R	L F

(4) Error Response
① General error
② Program number error
③ Error during program stop

Example

Command: !99EXT01@@
Response: #99EXT@@

4. Execution Text

4.11 Insert Program Step

(1) Function
Inserts step data before a specified program step.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
!	Code		I N S			Program Number		Step Number				A / O	Condition 1				Command				Operand #1					
	H	L											N	I/O Flag												

7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
Operand #1			Operand #2									Post			Comment											

4	5	6	7	8	9	0	1	2	3
(18 characters)						S H	C L	C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code		I N S			S	C	C	L
	H	L				H	L	R	F

(4) Error Response
① General error
② Program number error
③ Step number error
④ Program being executed

Example

Command: !99INS200010A 15 PATH1   0   320PATH/TURN ON 320 @@
Response: #99INS@@

4.12 Reorganize Program Memory

(1) Function
Reorganizes program memory.

(2) Command

0	1	2	3	4	5	6	7	8	9
!	Code H L		P R S			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		P R S			S C H L		C R	L F

(4) Error Response
① General error

Example

Command: !99PRS@@
Response: #99PRS@@

4.13 Erase Program Step

(1) Function
Deletes specified program step.

(2) Command

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
!	Code H L		D E L			Program Number		Step Number				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		D E L			S C H L		C R	L F

(4) Error Response
① General error
② Program number error
③ Step number error

Example

Command: !99DEL010001@@
Response: #99DEL@@

4. Execution Text

4.14 Set Point Data

(1) Function
Sets the data for the specified point numbers.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9
!	Code H	L	P	S	E	Point #				Axis Pattern		Acceleration (1/100g)				Velocity (mm/sec)			

Axis Pattern:

7	~	0

Axis 8 ~ Axis 1

20	1	2	3	4	5	6	7	8								
Position Data (1/1000mm)									Position Data				S H	C L	C R	L F

Maximum length:
19 +no. of axes (8)
x 9 + 4 bytes

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H	L	P	S	E	S H	C L	C R	L F

(4) Error Response
① General error
② Point number error
③ Axis error
④ Data error

Example

Command: !99PSE0001010.30020000050.000@@
Response: #99PSE@@

4.15 Clear Point Data

(1) Function
Clears point data such as acceleration speed, speed, and position specified by the point number.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7
!	Code H	L	C	L	R	Starting Point #				Ending Point #				SH	CL	CR	LF

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H	L	C	L	R	SH	CL	CR	LF

(4) Error Response
① General error
② Program number error
③ Error during program execution

Example

Command: !99CLR00010010@@
Response: #99CLR@@

4.16 Copy Point Data

(1) Function
Copies specified point data.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1
#	Code H L		C P Y			Copy Source Start Point #				Copy Source End Point #				Copy Target Start Point #				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		C P Y			S C H L		C R	L F

(4) Error Response
① General error
② Point number error

Example

Command: !99CPY000100100020@@
Response: #99CPY@@

4.17 Shift Point Data

(1) Function
Moves specified point data.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1
#	Code H L		S F T			Shift Source Start Point #				Shift Source End Point #				Shift Target Start Point #				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		S F T			S C H L		C R	L F

(4) Error Response
① General error
② Axis number error

Example

Command: !99SFT000100100020@@
Response: #99SFT@@

4. Execution Text

4.18 Set Servo Parameters

(1) Function
Sets servo parameters.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
!	Code H	L	R	S	V	Axis Qty	Numerator				Denominator				Override				Operation Vel (mm/sec)				Maximum Vel (mm/sec)			

7	8	9	30	1	2	3	4	5	6	7	8
Acceleration (1/100g)				Maximum Acceleration (1/100g)				SH	CL	CR	LF

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H	L	R	S	V	SH	CL	CR	LF

(4) Error Response
① General error
② Data error

Example

Command: !99RSV21　1　100 100 30000.3020.0@@
Response: #99RSV@@

4.19 Set Servo Parameters By Axis

(1) Function
Sets servo parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7
!	Code H L		R A G			Axis No.	Axis Name	Service frequency (times/sec)				Numerator				Denominator				Override				Jog Velocity (mm/sec)			

8	9	30	1	2	3	4	5	6	7	8	9	40	1	2	3	4	5	6	7	8	9	50	1	2
Position End Band (Pulse)				Soft Limit (+) (1/1000mm)				Soft Limit (-) (1/1000mm)				Soft Limit Offset (mm)					Acceleration (1/100g)				S C H L		C R	L F

Note: Service frequency refers to the PID speed. If you set this at other than 0400, we cannot guarantee servo operation. Service frequency should always be specified 0400.

$$\frac{1}{400} = 2.5 \times 10^{-3} \text{ (msec)}$$

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R A G			S C H L		C R	L F

(4) Error Response
① General error
② Axis number error
③ Data error

Example

Command: !99RAG01400 1 1 100 30 20 150 0 1.6000.30@@
Response: #99RAG@@

4. Execution Text

4.20 Set Homing Parameters By Axis

(1) Function
Sets homing parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7
#	Code H L		R A H			Axis No.	Direction	Type	Sequence	Limit Polarity	Z Pulse Edge	Creep Velocity (mm/sec)				Position-end Search Velocity (mm/sec)				Z Pulse Search Velocity (mm/sec)				Offset Moving Length (mm)			

Direction:
0: Home side
1: Side across home

Type:
0: Hard stop
1: Limit

Sequence:
0 : No. 1
~
9 : No. 10

Limit Polarity:
0 : 0 for limit
1 : 1 for limit

Z Pulse Edge:
0 : 0 for Z phase
1 : 1 for Z phase

8	9	30	1	2	3	4	5	6	7	8	9
Home Deviation (Pulse)				Home Current Limit				S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R A H			S C H L		C R	L F

(4) Error Response
① General error
② Axis number error
③ Data error

Example

Command: !99RAH1001110 10 4 0 480 55 @@
Response: #99RAH@@

4.21 Set Motor Parameters By Axis

(1) Function
Sets motor parameters by axis.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
#	Code H L		R A M			Axis No.	Maximum Motor RPM				Encoder Pulse				Screw Lead (mm)				Multiplier				Position Gain			

7	8	9	30	1	2	3	4	5	6	7	8	9	40	1	2	3	4	5	6	7	8	9	50
Speed Gain				Feed/Forward Gain				Integral Gain				Total Gain				Integral Voltage Limit				Over Speed Constant			

1	2	3	4	5	6	7	8	9	60	1	2	3	4	5	6	7	8	9	70	1
Cumulative Error (Pulse)				Maximum Motor Current				Brake Time (1/100sec)				Motor Overload Lower Limit					S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R A M			S C H L		C R	L F

(4) Error Response
① General error
② Axis number error
③ Data error

Example

```
Command:   !99RAM14000384 1 16  4   30  80  0   15  60  60  400 384090  0.1023600@@
Response:  #99RAM@@
```

4.22 Set Arc Parameters

(1) Function
Sets the circular parameters.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7
!	Code H	L	R	C	R	Slice Angle (1/10 degree)					Speed Increment (mm/sec)			S H	C L	C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H	L	R	C	R	S H	C L	C R	L F

(4) Error Response
① General error
② Data error

Example

Command: !99RCR15.0 0 @@
Response: #99RCR@@

4.23 Halt*

(1) Function
Slows the axis to a stop specified by the axis pattern.

Note:** **Do not use the Halt protocol command during homing.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1
!	Code H L		H L T			Axis Pattern		S C H L		C R	L F

7	~	0

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		H L T			S C H L		C R	L F

(4) Error Response
① General error

Example

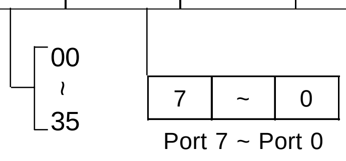
Command: !99HLT03@@
Response: #99HLT@@

4.24 Set Output Port

(1) Function
Sets the output port.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3
!	Code H L		O T S			Number		Data		S H	C L	C R	L F



(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		O T S			S H	C L	C R	L F

Note: *Caution is required because point number designation differs from IAI's standard expression. The 8 numbers from 300 ~ 307 begin at 0 and increases 1 for every 8 numbers. The data expresses this as a hexadecimal value.*

(4) Error Response
① General error
② Number error
③ Data error

Ex. 1: When output ports 305 and 307 are ON, both 305 and 307 belong to the same number group (same port 0). But because 305 has a value of 2 and 307 a value of 8, it becomes 2+8 = A...AØ. In this case, output from the same port other than 305 and 307 is OFF.

	8	4	2	1	8	4	2	1
	307	306	305	304	303	302	301	300

Ex. 2:

Number	Output Port #
00	300 ~ 307
01	308 ~ 315
⋮	⋮
35	580 ~ 587

To turn on output 302, select "Number 00" from the table above. Then find the "data" as follows,

0	0	0	0	0	1	0	0
↑	↑	↑	↑	↑	↑	↑	↑
307	306	305	304	303	302	301	300

$(00000100)_2 = (04)_{16}$

Command: !99OTS0004@@
Response: #99OTS@@

Ex. 3: Then to turn output 302 back off, the data would look as follows,

0	0	0	0	0	0	0	0
↑	↑	↑	↑	↑	↑	↑	↑
307	306	305	304	303	302	301	300

$(00000000)_2 = (00)_{16}$

↑

Command: !99OTS00$\overbrace{00}$@@

4.25 Set Global Flags

(1) Function
Sets global flags.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2	3
!	Code H L		G F S			Number		Data		S C H L		C R	L F

Number: 00 ~ 35

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		G F S			S C H L		C R	L F

*Basically the same as the output port set command except a 0 in the group number field corresponds to flags 600 ~607.

(4) Error Response
① General error
② Number error
③ Data error

Example

To turn on global flag 602, select "Number 00" from the table at right. Then find the "data" as follows,

Number	Global Flag
00	600 ~ 607
01	608 ~ 615
⋮	⋮
35	880 ~ 887

0	0	0	0	0	1	0	0
↑	↑	↑	↑	↑	↑	↑	↑
608	606	605	604	603	602	601	600

$(00000100)_2 \quad = \quad (04)_{16}$

```
Command:    !99GFS0004@@
Response:   #99GFS@@
```

4. Execution Text

4.26 Clear Memory

(1) Function
Erases the parameter, program and point area.

(2) Command

0	1	2	3	4	5	6	7	8	9	10	1	2
!	Code H L		R C L			Parameter	Program	Point	S C H L		C R	L F

0: will not erase
1: will erase

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R C L			S C H L		C R	L F

(4) Error Response
① General error

Example

Command: !99RCL000@@
Response: #99RCL@@

4.27 Reset

(1) Function
Resets the driver.

(2) Command

0	1	2	3	4	5	6	7	8	9
!	Code H L		R S T			S C H L		C R	L F

(3) Response

0	1	2	3	4	5	6	7	8	9
#	Code H L		R S T			S C H L		C R	L F

(4) Error Response
① General error

Example

Command: !99RST@@
Response: #99RST@@

5. Error Response

5.1 Format

0	1	2	3	4	5	6	7	8
%	Code H	L	Error Code H	L	S H	C L	C R	L F

5.2 General Error

Error Code	Error Name	Explanation
01	Command Error	Receiving something other than ? or !
02	Receive Length Error	Text length mismatch
03	ID Error	ID mismatch
04	Sum Check Error	Sum check error
05	Time Out Error	Time out occurs
06	Stopper Error	Not in CR, LF order
07	Parity Error	Parity error
08	Overrun Error	Overrun error
09	Framing Error	Framing error

5.3 Other Errors

Error Code	Error Name	Explanation
10H	Program Number Error	
11H	No Point Data Error	
12H	Point Number Error	
13H	Specified Speed Error	
14H	Specified Position Error	
15H	Specified Acceleration Error	
16H	Specified Axis Error	
17H	Data Error	
18H	Servo Error	
30H	Step Number Error	
31H	Step Number Over Error	

6. Program Sample

6.1 N88BASIC

No.	Program Name
1	Test call execution program
2	Input inquiry program
3	Point data inquiry program
4	Servo parameters by axis inquiry program
5	Axis status inquiry program
6	Task status inquiry program
7	Program status inquiry program
8	Error message inquiry program
9	Homing program
10	Specified position movement program
11	Point number specified movement program
12	Program execution program
13	Set point data program
14	Set output port program

6. Program Sample

6.1-1 Test Call Execution Program

```
1000  '
1010  '  Test call execution program
1020  '
1030  '
1040  '  This program exeuctes communication test,
1050  '  same data that was sent is returned.
1060  '
1070  '  SAVE  "A:  ¥SAMPLE¥S_ TST", A
1080     CLS
1090     OPEN  "COM:  N81NN"  AS #1                        'open communication line
1100       SRVTXT$ = "0123456789"
1110       LOCATE 22, 5
1120       PRINT "TEST CALL"
1130       LOCATE 25, 8
1140       PRINT "SEND = ";  SRVTXT$
1150       PRINT #1, "?99TST"+SRVTXT$+"@@"                 'send test command
1160       LINE INPUT #1, RCVTXT$                          'receive response
1170       LOCATE 25,11
1180       PRINT "SEND = ":  MID$(RCVTXT$,7,10)
1190       LOCATE 23,15
1200       IF MID$(RCVTXT$,7,10) = SRVTXT$ THEN            'error check
1210                  PRINT "TEST CALL OK !!"
1220       ELSE PRINT "TEST CALL ERR !!"
1230     CLOSE #1                                          'close communication line
1240     END
```

6. Program Sample

6.1-2 Input Inquiry Program

```
1000 '
1010 '  Input inquiry program
1020 '
1030 '
1040 '  This program executes input port inquiry.
1050 '
1060 '  SAVE "A:  ¥SAMPLE¥S_INP", A
1070    CLS
1080    OPEN "COM:  N81NN" AS #1                          'open communication line
1090      LOCATE 20, 6
1100     PRINT "INPUT PORT      (finish with a space)"
1110      LOCATE 21, 8
1120      PRINT "IN 0-7"
1130      LOCATE 21, 10
1140      PRINT "IN 8-15"
1150      LOCATE 21, 12
1160      PRINT "IN 16-23"
1170      L1 = 0
1180      WHILE L1 = 0
1190        IF INKEY$ = " " THEN L1 = 1
1200        PRINT #1, "99INP@@"                            'send input inquiry command
1210        LINE INPUT #1, RCVTXT$                         'receive response
1220        FOR I = 0 TO 2
1230          C = VAL(  "&H=MID$(RCVTXT$, 7+(I*2),2))      'giving a numerical value to data
1240          P$ = "00000000"
1250          J=1
1260          FOR K=0 TO 7
1270            IF (C AND J) >< 0 THEN MID$(P$, 8-K, 1) = "1"    'converting to binary
1280            J = J * 2
1290          NEXT
1300          LOCATE 36, I*2+8
1310          PRINT P$
1320        NEXT
1330     WEND
1340    CLOSE #1
1350    END                                               'close communication line
```

6. Program Sample

6.1-3 Point Data Inquiry Program

```
1000  '
1010  '  Point data inquiry program
1020  '
1030  '
1040  '  This program executes an inquiry of the point data
1050  '  specified by the point number.
1060  '
1070  '  SAVE "A:¥SAMPLE¥S_POS", A
1080     CLS
1090     OPEN "COM:N81NN" AS #1                                   'open communiation line
1100       PRINT #1, "99IPO@@"                                    'point parameter inquiry
1110       LINE INPUT #1, RCVTXT$                                 'receive response
1120       PMAX = VAL( MID$(RCVTXT$,7,4) )                        'check number of points
1130       L1 = 0
1140       WHILE L1 = 0
1150         L2 = 0
1160         WHILE L2 = 0
1170           LOCATE 3,0
1180           PRINT "INPUT POINT NUMBER.";
1190           PRINT "0-";PMAX;",END=-1)   ";
1200           INPUT PNUM
1210           CLS
1220           LOCATE 30,1
1230           IF PNUM < 0 THEN END
1240           IF PMAX<PNUM THEN PRINT "OUT OF RANGE" ELSE L2=1
1250         WEND
1260         PNUM$ = MID$(STR$(PNUM),2)
1270         PNUM$ = STRING$(4-LEN(PNUM$), "0")+PNUM$
1280         PRINT #1, "?99POS"+PNUM$+"@@"                          'send point inquiry command
1290         LINE INPUT #1, RCVTXT$                                 'receive response
1300         LOCATE 18,2
1310         PRINT "POINT NUMBER-   = ";
1320         PRINT MID$( RCVTXT$,  7, 4 )                           'point number display
1330         LOCATE 18,3
1340         PRINT "AXIS PATTERN    = ";
1350         PRINT MID$(RCVTXT$ 11,2)                               'axis pattern display
1360         AXIS = 0
1370         J = 1
1380         FOR I=1 TO 8                                           'check axis pattern
1390           IF (VAL("&h"+MID$(RCVTXT$,11,2))AND J)<>0 THEN AXIS=AXIS+1
1400           J = J * 2
1410         NEXT I
1420         LOCATE 18,4
1430         PRINT "ACCELERATION SPEED (g)    =  ";
1440         PRINT MID$(  RCVTXT$, 13, 4 )
1450         LOCATE 18,5
1460         PRINT "ACCELERATION (m/sec)      =  ";
1470         PRINT MID$( RCVTXT$, 17, 4 )
1480         FOR I=0 TO AXIS - 1                                    'data display
1490           LOCATE 18,I+6
1500           PRINT "POSITION DATA ("+CHR$(49+I)+")  =  ";
1510           PRINT MID$( RCVTXT$, I*9+21, 9 )
1520         NEXT I
1530       WEND
1540     CLOSE #1
1550     END                                                        'close communication line'
```

6. Program Sample

6.1-4 Servo Parameters by Axis Inquiry Program

```
1000    '
1010    '  Servo parameters by axis inquiry program
1020    '
1030    '
1040    '  This program executes servo parameter inquiries by axis.
1050    '
1060    '  SAVE "A:¥SAMPLE¥S_IAG",A
1070       CLS
1080       OPEN "COM:N81NN AS # 1                                  'open communication line
1090         RCVTXT$ = ""
1100         LOCATE 0, 0
1110         PRINT "SERVO PARAMETER"
1120         LOCATE 0, 2
1130         PRINT "AXIS NUMBER  AXIS NAME  SERVICE  NUMERATOR  DENOMINATOR  OVER ";
1140         PRINT "JOG     POSITIONING  SOFT    SOFT    SOFT LIMIT  ACCELERATION SPEED";
1150         PRINT "    NO.          SPEED                    WRITE  ";
1160         PRINT "SPEED  RANGE         LIMIT+  LIMIT- OFFSET";
1170         PRINT "SPEED   TIMES/s          ";
1180         PRINT: "(mm/s)  (PULSE)  (mm)  (mm)  (mm)  (g)"
1190         PRINT #1, "?99STA@@"                               'axis status inquiry command
1200         LINE INPUT #1, RCVTXT$                             'receive response
1210         FOR I =0 TO VAL( MID$RCVTXT$,7,1) )-1
1220           PRINT #1, "?99IAG"+CHR$(I+48)+"@@"               'send inquiry command
1230           LINE INPUT #1, RCVTXT$                           'receive response
1240           LOCATE 2, I*2+6
1250           PRINT MID$(RCVTXT$,  7, 1 )                      'axis number
1260           LOCATE 7, I*2+6
1270           PRINT MID$(RCVTXT$,  8, 1 )                      'axis name
1280           LOCATE 12, I*2+6
1290           PRINT MID$(RCVTXT$,  9, 4)                       'service speed
1300           LOCATE 19, I*2+6
1310           PRINT MID$(RCVTXT$,  13, 4)                      'numerator
1320           LOCATE 26, I*2+6
1330           PRINT MID$(RCVTXT$,  17, 4)                      'denominator
1340           LOCATE 33, I*2+6
1350           PRINT MID$(RCVTXT$,  21, 4)                      'over write
1360           LOCATE 40, I*2+6
1370           PRINT MID$(RCVTXT$,  25, 4)                      'jog speed
1380           LOCATE 47, I*2+6
1390           PRINT MID$(RCVTXT$,  29, 4)                      'positioning range
1400           LOCATE 54, I*2+6
1410           PRINT MID$(RCVTXT$,  33, 4)                      'soft limit +
1420           LOCATE 61, I*2+6
1430           PRINT MID$(RCVTXT$,  37, 4)                      'soft limit -
1440           LOCATE 68, I*2+6
1450           PRINT MID$(RCVTXT$,  41, 5)                      'soft limit offset
1460           LOCATE 75, I*2+6
1470           PRINT MID$(RCVTXT$,  46, 4)                      'acceleration
1480         NEXT I
1490       CLOSE #1                                             'close communication line
1500       END
```

6. Program Sample

6.1-5 Axis Status Inquiry Program

```
1000  '
1010  '  Axis status inquiry program
1020  '
1030  '
1040  '  This program executes axis status inquiry.
1050  '
1060  '  SAVE "A:¥SAMPLE¥S_STA" ,A
1070     CLS
1080     OPEN "COM:N81NN AS # 1                              'open communication line
1090       LOCATE 5, 0
1100       PRINT "AXIS STATUS"
1110       LOCATE 9, 2
1120       PRINT "AXIS NUMBER  SERVO    HOME    ";
1130       PRINT "MOVE    ERROR CODE    CURRENT POSITION";
1140       PRINT #1, "?99STA@@"                              'axis status inquiry command
1150       LINE INPUT #1, RCVTXT$                            'receive response
1160       FOR I = 1 TO VAL( MID$(RCVTXT$,7,1) )
1170         LOCATE 10, I*2+2
1180         PRINT "(";I;")"                                 'axis number
1190         LOCATE 20, I*2+2
1200         PRINT MID$(RCVTXT$,  I*14-6, 1 )                'servo
1210         LOCATE 30, I*2+2
1220         PRINT MID$(RCVTXT$,  I*14-5, 1 )                'home
1230         LOCATE 40, I*2+2
1240         PRINT MID$(RCVTXT$,  I*14-4, 1 )                'move
1250         LOCATE 50, I*2+2
1260         PRINT MID$(RCVTXT$,  I*14-3, 2)                 'error code
1270         LOCATE 60, I*2+2
1280         PRINT MID$(RCVTXT$,  I*14-1, 9)                 'current position
1290     NEXT
1300     CLOSE #1                                            'close communication line
1310     END
```

6. Program Sample

6.1-6 Task Status Inquiry Program

```
1000  '
1010  '  Task status inquiry program
1020  '
1030  '
1040  '  This program executes task status inquiry.
1050  '
1060  '  SAVE "A:¥SAMPLE¥S_TSK" ,A
1070     CLS
1080     OPEN "COM:N81NN AS # 1                                  'open communication line
1090       LOCATE 5, 0
1100       PRINT "TASK STATUS"
1110       LOCATE 9, 2
1120       PRINT :TASK NO. STATUS  LEVEL P. N. ";
1130       PRINT :    TASK NO.  STATUS  LEVEL P. N. ";
1140       PRINT #1, "?99TSK@@"                                  'task status inquiry command
1150       LINE INPUT #1, RCVTXT$                                'receive response
1160       FOR I = 0 TO INT( VAL( MID$(RCVTXT$,7,2) )/2 )-1      'number of axes
1170         FOR J = 0 TO 1
1180           LOCATE J*33+10, I*2+4
1190           PRINT "(";I*2+J+1;")"                             'task number
1200           LOCATE J*33+20, I*2+4
1210           PRINT MID$(RCVTXT$, I*2+J)*4+9, 1)                'status
1220           LOCATE J*33+26, I*2+4
1230           PRINT MID$(RCVTXT$, I*2+J)*4+10, 1)               'level
1240           LOCATE J*33+33, I*2+4
1250           PRINT MID$(RCVTXT$, I*2+J)*4+11, 2)               'program No
1260         NEXT
1270       NEXT
1280     CLOSE #1                                                'close communication line
1290     END
```

6. Program Sample

6.1-7 Program Status Inquiry Program

```
1000    '
1010    '  Program status inquiry program
1020    '
1030    '
1040    '  This program executes program status inquiry
1050    '
1060    '  SAVE "A:¥SAMPLE¥S_PRG" ,A
1070       CLS
1080       PNE$ = THERE IS NO PROGRAM
1090       OPEN "COM:N81NN AS # 1                                        'open communication line
1100         PRINT #1, "?99IPG@@"                                        'program status inquiry
1110         LINE INPUT #1, RCVTXT$                                      'receive response
1120         PMAX = VAL( MID$(RCVTXT$,11, 2) )                           'check number of points
1130         L1 = 0
1140         WHILE L1 = 0
1150           L2 = 0
1160           WHILE L2 = 0
1170             LOCATE 5, 1
1180             PRINT "PLEASE INPUT PROGRAM NO.";
1190             PRINT "(0 -";PMAX;",END=-1)
1200             LOCATE 60, 1
1210             INPUT PNUM
1220             IF PNUM < 0 THEN END
1230             CLS
1240             IF PMAX < PNUM THEN LOCATE 30,2:PRINT "OUT OF RANGE" ELSE L2 = 1
1250           WEND
1260           LOCATE 15, 5
1270           PRINT "PROGRAM NUMBER- ="
1280           LOCATE 15, 7
1290           PRINT "STATUS- ="
1300           LOCATE 15, 9
1310           PRINT "ERROR CODE- ="
1320           LOCATE 15, 11
1330           PRINT "STEP NUMBER- ="
1340           PNUM$ = MID$( STR$(PNUM)2,2 )
1350           PNUM$ = PNUM$+STRING$(2-LEN(PNUM$)," ")
1360           PRINT #1, "?99PRG"+PNUM$+"@@"                             'send inquiry command
1370           LINE INPUT #1, RCVTXT$                                    'receive response
1380           IF RCVTXT$ = "%9910@@" THEN LOCATE 24,2:PRINT PNE$
1390             LOCATE 44,5
1400             PRINT MID$( RCVTXT$,  7, 2 )                            'program number
1410             LOCATE 44, 7
1420             PRINT MID$( RCVTXT$,  9, 1 )                            'status
1430             LOCATE 44, 9
1440             PRINT MID$( RCVTXT$,  10, 2 )                           'error code
1450             LOCATE 44, 11
1460             PRINT MID$( RCVTXT$,  12, 4 )                           'status number
1470         WEND
1480       CLOSE #1                                                      'close communication line
1490       END
```

6.1-8 Error Message Inquiry Program

```
1000   '
1010   '  Error message inquiry program
1020   '
1030   '
1040   '  This program executes error message inquiry.
1050   '
1060   '  SAVE "A:¥SAMPLE¥S_MSG" ,A
1070      CLS:
1080      OPEN "COM:N81NN AS # 1                                    'open communication line
1090        NO$ = "THERE IS NO SUCH ERROR CODE"
1100        LI = 0
1110        WHILE E = 0
1120          LOCATE 10, 2
1130          PRINT "PLEASE INPUT ERROR CODE.'"
1140          PRINT "(END=-1)
1150          LOCATE 50, 2
1160          INPUT ECORD$                                          'input error code
1170          IF ECORD$ = "-1" THEN END
1180          CLS
1190          LOCATE 16, 7
1200          PRINT "ERROR CODE   ="
1210          LOCATE 16, 9
1220          PRINT "ERROR MESSAGE   ="
1230          PRINT #1, "?99MSG"+ECORD$+"@@"                        'send inquiry command
1240          LINE INPUT #1, RCVTXT$                                'receive response
1250          IF "%"=MID$(RCVTXT$,1,1) THEN LOCATE 40,9:PRINT NO$
1260          LOCATE 40, 7
1270          PRINT MID$( RCVTXT$,  7, 2 )                          'error code
1280        LOCATE 40, 9
1290          PRINT MID$( RCVTXT$,  9, 16)                          'message
1300        WEND
1310      CLOSE #1                                                  'close communication line
1320      END
```

6. Program Sample

6.1-9 Homing Program

```
1000  '
1010  '  Homing program
1020  '
1030  '
1040  '  This program executes homing.
1050  '
1060  '  SAVE "A:¥SAMPLE¥Z_HOM" ,A
1070     CLS
1080     OPEN "COM:N81NN AS # 1                                   'open communication line
1090       PRINT "RUNNING INQUIRY ON AXIS DATA"
1100       SRVTXT$ = "?99STA@@"                                   'axis parameter inquiry
1110       GOSUB *SND
1120       AXIS = VAL( MID$(RCVTXT$,7,1) )                        'reading number of axes
1130       J = 1                                                  'converting to axis pattern
1140       FOR I = 1 TO AXIS
1150         A = A + J
1160         J = J * 2
1170       NEXT
1180       AXIS$ = HEX$( A )
1190       IF LEN(AXIS$) = 1 THEN AXIS$ = "0"+AXIS$
1200       PRINT "EXECUTING  HOMING    "
1210       SRVTXT$ = "!99HOM"+AXIS$+"40@@"                        'send homing command
1220       GOSUB *SND
1230       LI = 1
1240       WHILE LI <> 0                                          'check homing
1250         LI = AXIS
1260         SRVTXT$ = "?99STA@@"                                 'axis parameter inquiry
1270         GOSUB *SND
1280         FOR I = 1 TO AXIS
1290           LI = LI - VAL( MID$(RCVTXT$,I*14-5,1) )
1300         NEXT
1310       WEND
1320       PRINT "HOMING COMPLETE    "
1330     CLOSE #1                                                 'complete
1340     END
1350  '  ****************************
1360  '  **   RECEIVE RESPONSE    **
1370  '  ****************************
1380  *  SND
1390     PRINT #1,SRVTXT$
1400     LINE INPUT #1,RCVTXT$                                    'receive response
1410     IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                    'error check
1420     PRINT "RESPONSE ERROR =",RCVTXT$                         'error processing
1430     BEEP
1440     END
```

6. Program Sample

6.1-10 Specified Position Movement Program

```
1000   '
1010   '  Move to specified position program
1020   '
1030   '
1040   '  This program executes movement to a specified position.
1050   '
1060   '  SAVE "A:¥SAMPLE¥Z_MOV" ,A
1070      CLS
1080      OPEN "COM:N81NN AS # 1                                      'open communication line
1090        GOSUB *STA                                                'axis data inquiry
1100        GOSUB *SRV                                                'servo ON
1110        GOSUB *HOM                                                'homing
1120        GOSUB *MOV                                                'specified move
1130        LOCATE 30, 22
1140        PRINT "ACTUATOR STOP    "
1150        SRVTXT$ = "!99HLT"+AXIS$+"@@"                             'stop
1160        GOSUB *SND
1170        LOCATE 30, 22
1180        PRINT "SERVO OFF    "
1190        SRVTXT$ = "!99SRV"+AXIS$+"0@@"                            'servo OFF
1200        GOSUB *SND
1210      CLOSE #1
1220      END
1230   '  **********************
1240   '  **   AXIS INQUIRY   **
1250   '  **********************
1260    *STA
1270      LOCATE 30, 22
1280      PRINT "RUNNING AXIS DATA INQUIRY"
1290      SRVTXT$ = "?99STA@@"                                        'axis parameter inquiry
1300      GOSUB *SND
1310      AXIS = VAL( MID$(RCVTXT$,7,1) )                             'reading number of axes
1320      J = 1                                                       'converting to axis pattern
1330      FOR I = 1 TO AXIS
1340        A = A + J
1350        J = J * 2
1360      NEXT
1370      AXIS$ = HEX$( A )
1380      IF LEN(AXIS$) = 1 THEN AXIS$ = "0"+AXIS$
1390      RETURN
1400   '  ******************
1410   '  **   SERVO ON   **
1420   '  ******************
1430    *SRV
1440      LOCATE 30, 22
1450      PRINT "SERVO CHECK"
1460        SRVTXT$ = "?99STA@@"                                      'axis parameter inquiry
1470        GOSUB *SND
1480      J = 1
1490      FOR I = 1 TO AXIS
1500        IF MID$(RCVTXT$,I*14-6,1) = "1" THEN *SKIP                'servo check
1510        SRVTXT$ = STRING $(2-LEN(HEX$(J)),"0")+HEX$(J)
1520        SRVTXT$ = "!99SRV"+SRVTXT$+"1@@"
1530        GOSUB *SND                                                'servo ON
```

6. Program Sample

```
1540        J = J * 2
1550        LOCATE 30, 22
1560        PRINT "SERVO ON    "
1570     *SKIP
1580       NEXT
1590       RETURN
1600   '   ****************
1610   '   **   HOMING   **
1620   '   ****************
1630    *HOM
1640       GOSUB *JPS
1650       IF L1 = 0 THEN *GEND
1660       LOCATE 30, 22
1670       PRINT "EXECUTING HOMING"
1680       SRVTXT$ = "!99HOM"+AXIS$+"40@@"                               'homing command
1690       GOSUB *SND
1700       L1 = 1
1710       WHILE L1 <> 0
1720         GOSUB *JPS
1730       WEND
1740    *GEND
1750       LOCATE 30, 22
1760       PRINT "HOMING COMPLETE"
1770       RETURN
1780   '   **********************
1790   '   **   CHECK HOMING   **
1800   '   **********************
1810    *JPS
1820       L1 = AXIS
1830       SRVTXT$ = "?99STA@@"                                          'axis parameter inquiry
1840       GOSUB *SND
1850       FOR I = 1 TO AXIS
1860      L1 = L1 - VAL( MID$(RCVTXT$,I*14-5,1) )
1870       NEXT
1880       RETURN
1890   '   ************************************
1900   '   **   MOVE TO SPECIFIED POSITION   **
1910   '   ************************************
1920    *MOV
1930       RANDOMIZE TIME/4
1940       LOCATE 14, 2
1950       PRINT "ACTUATOR SPECIFIED MOVE (END WITH A SPACE)"
1960       LOCATE 18, 4
1970       PRINT "AXIS NUMBER    SPECIFIED POSITION    CURRENT POSITION"
1980       FOR I = 1 TO AXIS
1990         LOCATE 20, I*2+4
2000          PRINT "ACTUATOR(";I;")"
2010       NEXT
2020       L1 = 0
2030       L2 = 0
2040       WHILE L1 = 0
2050         J = 1
2060         AST = 0
2070         FOR I = 1 TO AXIS
```

6. Program Sample

```
2080        SRVTXT$ = "?99STA@@"                                  'axis parameter inquiry
2090        GOSUB *SND
2100        LOCATE 50, I*2+4
2110        PRINT MID$(RCVTXT$,I*14-1,9)                          'displays current position
2120        IF MID$(RCVTXT$,I*14-1)="0" THEN GOSUB *SET           'if at rest, goes to set
2130        J = J * 2
2140      NEXT
2150      LOCATE 30, 22
2160      PRINT "ACTUATOR IN OPERATION"
2170      IF INKEY$ = " " THEN L2 = 1
2180      IF AST = AXIS THEN L1 = 1                               'checking stop
2190    WEND
2200    RETURN
2210  ' **********************
2220  ' **   SET POSITION   **
2230  ' **********************
2240  *SET
2250    IF L2 = 1 THEN AST = AST + 1: RETURN
2260    SRVTXT$ = "!99IAG"+CHR$(I+47)+"@@"                        'servo parameter inquiry
2270    GOSUB *SND
2280    LIMIT = VAL( MID$(RCVTXT$,33,4)                           'reading soft limit
2290    A$ = MID$(STR$(INT(RND*LIMIT)),2,8)
2300    A$ = STRING$(3-LEN(A$),"0")+A$                            'creating specified position
2310    JP$ = STRING$(2-LEN(HEX$(J)),"0")+HEX$(J)
2320    SRVTXT$ = "!99MOV"+JP$+"0.100100"+A$+"   @@"              'send  specified move command
2330    GOSUB *SND
2340    LOCATE 30, 22
2350    PRINT "SET POSITION"
2360    LOCATE 38, I*2+4
2370    PRINT A$                                                  'displays specified position
2380    RETURN
2390  ' **************************
2400  ' **   RECEIVE RESPONSE   **
2410  ' **************************
2420  *SND
2430    PRINT #1, SRVTXT$
2440    LINE INPUT #1, RCVTXT$                                    'receive response
2450    IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                     'error check
2460    LOCATE 30, 22
2470    PRINT "RESPONSE ERROR =",RCVTXT$                          'error processing
2480    BEEP
2490    END
```

6.1-11 Point Number Specified Movement Program

```
1000  '
1010  '  Point number specified move program
1020  '
1030  '
1040  '  This program executes move by specified point number.
1050  '
1060  '  SAVE "A:¥SAMPLE¥Z_PMV" ,A
1070     CLS
1080     OPEN "COM:N81NN AS # 1                                  'open communication line
1090       GOSUB *STA                                            'axis data inquiry
1100       GOSUB *SRV                                            'servo ON
1110       GOSUB *HOM                                            'homing
1120       GOSUB *PMV                                            'move by specified point
1130       LOCATE 30, 22
1140       PRINT "ACTUATOR STOP    "
1150       SRVTXT$ = "!99HLT"+AXIS$+"@@"                         'stop
1160       GOSUB *SND
1170       SRVTXT$ = "!99SRV"+AXIS$+"0@@"                        'servo OFF
1180       GOSUB *SND
1190     CLOSE #1
1200     END
1210  '  **********************
1220  '  **   AXIS INQUIRY   **
1230  '  **********************
1240   *STA
1250     LOCATE 30, 22
1260     PRINT "RUNNING AXIS DATA INQUIRY"
1270     SRVTXT$ = "?99STA@@"                                    'axis parameter inquiry
1280     GOSUB *SND
1290     AXIS = VAL( MID$(RCVTXT$,7,1) )                         'reading number of axes
1300     J = 1                                                   'converting to axis pattern
1310     FOR I = 1 TO AXIS
1320       A = A + J
1330       J = J * 2
1340     NEXT
1350     AXIS$ = HEX$( A )
1360     IF LEN(AXIS$) = 1 THEN AXIS$ = "0"+AXIS$
1370     RETURN
1380  '  ******************
1390  '  **   SERVO ON   **
1400  '  ******************
1410   *SRV
1420     LOCATE 30, 22
1430     PRINT "SERVO CHECK  "
1440     SRVTXT$ = "?99STA@@"                                    'axis parameter inquiry
1450     GOSUB *SND
1460     STA$ = RCVTXT$
1470     J = 1
1480     FOR I = 1 TO AXIS
1490       IF MID$(STA$,I*14-6,1) = "1" THEN *SKIP               'servo check
1500       SRV$ = STRING $(2-LEN(HEX$(J)),"0")+HEX$(J)
1510       SRVTXT$ = "!99SRV"+SRV$+"1@@"
1520       GOSUB *SND                                            'servo ON
1530       J = J * 2
```

```
1540        LOCATE 30, 22:  PRINT "SERVO ON   "
1550     *SKIP
1560       NEXT
1570       RETURN
1580    '   ****************
1590    '   **   HOMING   **
1600    '   ****************
1610     *HOM
1620       GOSUB *JPS
1630       IF LI = 0 THEN *GEND
1640       LOCATE 30, 22
1650       PRINT "EXECUTING HOMING"
1660       SRVTXT$ = "!99HOM"+AXIS$+"40@@"                          'homing command
1670       GOSUB *SND
1680       LI = 1
1690       WHILE LI <> 0
1700         GOSUB *JPS
1710       WEND
1720     *GEND
1730       LOCATE 30, 22
1740       PRINT "HOMING COMPLETE"
1750       RETURN
1760    '   **********************
1770    '   **   CHECK HOMING   **
1780    '   **********************
1790     *JPS
1800       LI = AXIS
1810       SRVTXT$ = "?99STA@@"                                     'axis parameter inquiry
1820       GOSUB *SND
1830       FOR I = 1 TO AXIS
1840       LI = LI - VAL( MID$(RCVTXT$,I*14-5,1) )
1850       NEXT
1860       RETURN
1870    '   *************************************
1880    '   **   POINT NUMBER SPECIFIED MOVE   **
1890    '   *************************************
1900     *PMV
1910       SRVTXT$ = "?99IPO@@"                                     'point  parameter inquiry
1920       GOSUB *SND
1930       PMAX = VAL( MID$(RCVTXT$,7,4) )                          'reading number of points
1940       LI = 0
1950       WHILE LI = 0
1960         LOCATE 14, 2
1970         PRINT "POINT NUMBER SPECIFIED MOVE"
1975         PRINT "(0 -";PMAX;",END=-1)"
2000         LOCATE 18,4
2010         INPUT PN                                               'input point number
2020         IF PN <0 THEN RETURN
2030         PN$ = MID$( STR$( PN ), 2, 4 )
2040         PN$ = STRING$(4-LEN(PN$),"0")+PN$
2050         SRVTXT$ = "!99PMV0300000000"+PN$+"@@"                  'send point move command
2060         GOSUB *SND
2070         LOCATE 30,22
2080         PRINT "MOVING"
```

```
2090         L2 = 1
2100         WHILE L2 <> 0
2110         SRVTXT$ = "?99STA@@"                                  'axis parameter inquiry
2120         GOSUB *SND
2130           L2 = 0
2140           FOR J = 1 TO AXIS
2150             L2 = L2 + VAL( MID$(RCVTXT$,J*14-4,1) )           'check stop
2160           NEXT
2170         WEND
2180         CLS
2190         LOCATE 30, 22
2200         PRINT "MOVE COMPLETE    "
2210       WEND
2220       RETURN
2230   '   ***************************
2240   '   **   RECEIVE RESPONSE   **
2250   '   ***************************
2260     *SND
2270       PRINT #1, SRVTXT$
2280       LINE INPUT #1, RCVTXT$                                  'receive response
2290       IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                   'error check
2300       LOCATE 30, 22
2310       PRINT "RESPONSE ERROR =",RCVTXT$                        'error processing
2320       BEEP
2330       END
```

6.1-12 Program Execution Program

```
1000  '
1010  '  Perform program
1020  '
1030  '
1040  '  Executes a program.
1050  '
1060  '  SAVE "A:¥SAMPLE¥Z_RUN",A
1070     CLS
1080     PNE$ = "THERE IS NO PROGRAM "
1090     OPEN "COM:N81NN AS # 1                                  'open communication line
1100              SRVTXT$ ="?991PG@@"                            'program parameter inquiry
1110              GOSUB *SND
1120              PMAX = VAL(MID$(RCVTXT$,11,2))                 'checks number of programs
1130      *LOOP
1140      LI = 0
1150      WHILE LI = 0
1160               PRINT "EXECUTE PROGRAM NUMBER?(0-";PMAX;")"
1170               INPUT PN                                      'input program number
1180               IF PN<0 OR PMAX<PN THEN PRINT"OUT OF RANGE"  ELSE LI = 1
1190      WEND
1200      PRINT "EXECUTING PROGRAM "
1210      PN$= MID$(STR$(PN), 2, 2)
1220      PN$ =STRING$(2-LEN(PN$), "0")+PN$
1230      SRVTXT$="!99RUN"+PN$+"@@"                             'execute  program
1240      GOSUB *SND
1250      PRINT "PROGRAM ENDS WITH A SPACE "
1260      LI = 0
1270      WHILE LI = 0
1280              IF INKEY$ ="" THEN LI = 1
1290      WEND
1300      SRVTXT$ = "!99EXT" +PN$ +"@@"                         'stop program
1310      GOSUB *SND
1320      PRINT "PROGRAM COMPLETED"
1330     CLOSE #1                                                'close communication line
1340     END                                                     'complete
1350  '  ***************************
1360  '  **   RECEIVE RESPONSE   **
1370  '  ***************************
1380     *SND
1390     PRINT #1, SRVTXT$
1400     LINE INPUT #1, RCVTXT$                                  'receive response
1410     IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                   'error check
1420     IF RCVTXT$ = "%9910@@" THEN PRINT PNE$:GOTO *LOOP
1430     PRINT "RESPONSE ERROR =",RCVTXT$                        'error processing
1440     BEEP
1450     END
```

6. Program Sample

6.1-13 Set Point Data Program

```
1000    '
1010    '  Point data set program
1020    '
1030    '
1040    '  This program executes point data set.
1050    '
1060    '  'SAVE "Z_PSE",A
1070       CLS
1080       OPEN "COM:N81NN AS # 1                                          'open communication line
1090              LI = 0
1100              WHILE LI = 0
1110                     PRINT "POINT NUMBER   (END=-1) ="                 'input point number
1120                     INPUT PN
1130                     IF PN < 0 THEN LI = 1:  GOTO *LEND                'check if complete
1140                     IF 9999 < PN THEN PRINT "OUTSIDE OF RANGE":GOTO *LEND
1150                     PN$ = MID$(STR$(PN), 2, 4)
1160                     PN$ = STRING$(4 -LEN(PN$), "0") + PN$
1170                     PRINT "AXIS PATTERN    =";                        'input axis pattern
1180                     INPUT JP$
1190                     JP$ = STRING$(2-LEN(JP$), "0") +JP$
1200                     SRVTXT$ = "?99ISV@@"                              'servo parameter inquiry
1210                     GOSUB *SND
1220                     KM$ = MID$(RCVTXT$, 32, 4)                        'read maximum acceleration speed
1230                     PRINT "ACCELERATION SPEED (g)  (MAX "+KM$+")="     ;  'input acceleration speed
1240                     INPUT KD
1250                     KD$ = MID$(STR$(KD), 2, 4)
1260                     KD$ = STRING$(4-LEN(KD$), "0") + KD$
1270                     SM$ = MID$(RCVTXT$, 24, 4)                        'read maximum speed
1280                     PRINT "SPEED (m/sec) (MAX "+SM$+") = "            'input speed
1290                     INPUT SD
1300                     SD$ = MID$(STR$(SD), 2, 4)
1310                     SD$ = STRING$(4-LEN(SD$), "0") + SD$
1320                     JP = VAL ("&h" + JP$)
1330                     ITI$ = ""                                         'initialize position data
1340                     J = 1
1350                     FOR I - 1 TO 8
1360                            IF (JP AND J) >< 0 THEN GOSUB *ITI         'input position data
1370                            J = J * 2
1380                     NEXT
1390                     PRINT " SET POINT DATA (Y/N)"
1400                     INPUT A$
1410                     IF A$ = "N" THEN GOTO *LEND
1420                     SRVTXT$ = "!99PSE" + PN$+JP$+KD$+SD$+ITI$+"@@"   'point data set
1430                     GOSUB *SND
1440               *LEND
1450                     PRINT""
1460        WEND
1470       CLOSE #1                                                        'close communication line
1480       END                                                             'complete
1490    '  **************************
1500    '  **   RECEIVE RESPONSE   **
1510    '  **************************
1520    *SND
1530    PRINT #1, SRVTXT$
```

6. Program Sample

```
1540      LINE INPUT #1, RCVTXT$                                  'receive response
1550      IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                   'error check
1560      PRINT "RESPONSE ERROR ", RCVTXT$                        'error processing
1570      BEEP
1580      END
1590    ' **************************
1600    ' **   POINT DATA SET   **
1610    ' **************************
1620     *ITI
1630       SRVTXT$ = "?991AG" + CHR$(I+47)+"@@"
1640       GOSUB *SND
1650       LS$ = MID$(RCVTXT$, 37, 4)                             'read minimum value
1660       LL$ = MID$(RCVTXT$, 33, 4)                             'read maximum value
1670       PRINT I; "AXIS POSITION  ("+LS$ +"--" + LL$ +") =";    'input position data
1680       INPUT ITI
1690       ITI$ = ITI$ + MID$(STR$(ITI), 2, 7)
1700       ITI$ = ITI$ + STRING$(7 - LEN(IT$),"")
1710       RETURN
```

6. Program Sample

6.1-14 Set Output Port Program

```
1000  '
1010  'Output port set program
1020  '
1030  '
1040  'This program executes setting of output port
1050  '
1060  'SAV "Z_OTS",A
1070  CLS
1080  OPEN "COM:N81NN" AS #1                                  'open communication line
1090  PRINT "OUTPUT PORT SET   (END WITH A SPACE.)"
1100  LI = 0
1110  WHILE LI = 0
1120     FOR I = 0 TO 2
1130              NUM$ = MID$(STR$(I),2,2)
1140              NUM$ =STRING$(2-LEN(NUM$), "0" + NUM$          'set number
1150              K = 1
1160              FOR J = 0 TO 7
1170              DAT$ = HEX$(K)
1180              DAT$ = STRING$(2-LEN(DAT$), "0") + DAT$        'set data
1190              SRVTXT$ = "!99OTS" + NUM$ + DAT$ + "@@"        'output port set
1200              GOSUB *SND
1210              FOR L = 0 TO 500:NEXT                          'wait
1220              K = K * 2
1230     NEXT
1240     SRVTXT$ = "!99OTS" + NUM$ + "00@@"
1250     GOSUB * SND
1260  NEXT
1270  IF INKEY$ = "" THEN LI = 1
1280  WEND
1290  CLOSE #1
1300  END
1310  '  **************************
1320  '  **   RECEIVE RESPONSE   **
1330  '  **************************
1340  *SND
1350  PRINT #1, SRVTXT$
1540  LINE INPUT #1, RCVTXT$                                  'receive response
1550  IF LEFT$(RCVTXT$,1) = "#" THEN RETURN                   'error check
1560  PRINT "RESPONSE ERROR ", RCVTXT$                        'error processing
1570  BEEP
1580  END
```

6. Program Sample

6.2 Q-BASIC

No.	Program Name
1	Execute test call program
2	Input inquiry program
3	Point data inquiry program
4	Servo parameters by axis inquiry program
5	Axis status inquiry program
6	Task status inquiry program
7	Program status inquiry program
8	Error message inquiry program
9	Homing program
10	Move to specified position program
11	Point number specified move program
12	Chosen program execution program
13	Point data set program
14	Output port set program

6. Program Sample

6.2-1 Execute Test Call Program

```
'*****************************************************************
'* TEST FOR ALL CALL
'*       (CHECKER FOR COMMAND RESPONSE)
'*
'*                   PROGRAM For QB45
'* RESPONSE CHECKER
'*
'*
'* Copyright (C) 1994 I.A.I. Corporation
'* Sales Engineering Department
'*
'* PNAME "S_CHKQ"
'*****************************************************************

   DEFINT I-J: DEFSTR W

  CLS

RSINI:
   OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1               'open communication line

   WSNDCMD = "VER"
   W20PLAD = "S1"
   LOCATE 5, 22, 1
   PRINT "COMMAND CHECK"
   LOCATE 8, 25, 1
   PRINT "SEND  =  "; "?99" + WSNDCMD + W20PLAD + "@@"
   PRINT #1, "?99" + WSNDCMD + W20PLAD + "@@"               'send test command
   LINE INPUT #1, WRCVTXT                                   'receive response
   WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
   LOCATE 11, 25, 1
   PRINT "RECEIVE  ="; WRCVTXT
  CLOSE #1                                                  'close communication line
END
```

6.2-2 Input Inquiry Program

```
'****************************************************************
'* INPUT PORT DATA INQUIRY & GET SAMPLE
'*
'*                  PROGRAM For QB45
'* INPUT PORT CHECK (DIAGNOSE) PROGRAM
'*
'* This program executes input port inquiry
'*
'* Copyright (C) 1994 I.A.I. Corporation
'* Sales Engineering Department
'*
'* PNAME "S_INPQ"
'****************************************************************

     DEFINT I-J: DEFSTR W

     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1        'open communication line

     LOCATE 6, 20, 1
     PRINT "INPUT PORT  (END WITH A SPACE)"
     LOCATE 8, 21, 1
     PRINT "IN 0-7"
     LOCATE 10, 21, 1
     PRINT "IN 8-15"
     LOCATE 12, 21, 1
     PRINT "IN 16-23"
     L1 = 0
     WHILE L1 = 0
     IF INKEY$ =""THEN L1 = 1
     PRINT #1, "?99INP@@"                              'send input inquiry command
     LINE INPUT #1, WRCVTXT                            'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
       FOR I = 0 TO 2
             C = VAL("&H" + MID$(WRCVTXT, 7 + (I * 2), 2))
                                                       'converting data to numerics
             WP = "00000000"
             J = 1
             FOR K = 0 TO 7
                         IF (C AND J) <> 0 THEN MID$(WP, 8 - K, 1) = "1"
                                                       'converting to binary
                         J = J * 2
             NEXT
             LOCATE I * 2 + 8, 36, 1
             PRINT WP
          NEXT
     WEND
   CLOSE #1                                            'close communication line
END
```

6.2-3 Point Data Inquiry Program

```
'***************************************************************
'*  POINT DATA REQUEST & GET SAMPLE
'*
'*                    PROGRAM For QB45
'*
'*  POINT DATA INQUIRY SAMPLE
'*
'*  This program executes inquiry for point
'*
'*  data specified by point numbers
'*
'*  Copyright (C) 1994 I.A.I. Corporation
'*
'*  Sales Engineering Department
'*
'*  PNAME "S_POSQ"
'***************************************************************
     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1             'open communication line

START:
        PRINT #1, "?99IPO@@"                                'point parameter inquiry

        LINE INPUT #1, WRCVTXT                              'receive response
        WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

        WT = LEFT$(WRCVTXT, 1)
        IF WT = "%" THEN GOTO ERDISP

     PMAX = VAL(MID$(WRCVTXT, 7, 4))                        'check number of points
           L1 = 0
            WHILE L1 = 0
                  L2 = 0
                        WHILE L2 = 0
                              LOCATE 10, 6, 1
           PRINT "INPUT REFERENCED POINT NUMBER.";
           PRINT "(0 -": PMAX; ",END=-1) ";
           INPUT PNUM
            CLS
            LOCATE 1, 25, 1

            IF PNUM < 0 THEN END
            IF PMAX < PNUM THEN PRINT "OUTSIDE OF RANGE" ELSE L2 = 1
                        WEND
        WPNUM = MID$(STR$(PNUM), 2)

        WPNUM = STRING$(4 - LEN(WPNUM), "0") + WPNUM

        PRINT #1, "?99POS" + WPNUM + "@@"                   'send point inquiry command
        LINE INPUT #1, WRCVTXT                              'receive response
        WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
        LOCATE 2, 18, 1
        PRINT "POINT NUMBER  =  ";
        PRINT MID$(WRCVTXT, 7, 4)                       'displays point number
        LOCATE 3, 18, 1
        PRINT "AXIS PATTERN  =  ";
        PRINT MID$(WRCVTXT, 11, 2)                      'displays axis pattern
   AXIS = 0
      J = 1
      FOR I = 1 TO 8                                    'check axis pattern

  IF (VAL("&h" + MID$(WRCVTXT, 11, 2)) AND J) <> 0 THEN AXIS = AXIS + 1
        J = J * 2
         NEXT I

    LOCATE 4, 18, 1
  PRINT "ACCELERATION SPEED (g) = ";:PRINT MID$(WRCVTXT, 13, 4)

    LOCATE 5, 18, 1
  PRINT "SPEED (m/sec)          = ";:PRINT MID$(WRCVTXT, 17, 4)

    FOR I = 0 TO AXIS -1                                'display data

    LOCATE I + 6, 18, 1
       PRINT "POSITION DATA (" + CHR$(49 + I) + ") ="
       PRINT MID$(WRCVTXT, I * 9 + 21, 9)
       NEXT I
    WEND
   CLOSE #1                                             'close communication line
END

ERDISP:                                                 'error display
   WERR = MID$(WRCVTXT, 4, 2)
   PRINT "ERROR.  ERROR CODE="; WERR: INPUT "CAN CONTINUE? (Y/N)"; WYN
   IF WYN = "Y" OR WYN = "y" OR WYN = "N" GOTO START
   CLOSE #1
END
```

6.2-4 Servo Parameter by Axis Inquiry Program

```
'*************************************************************
'* SERVO PARAMETER (EACH AXIS)
'*             INQUIRY & GET SAMPLE
'*                   PROGRAM For QB45
'*
'* PROGRAM STATUS CHECK (INQUIRY) PROGRAM
'* This program executes servo parameter
'* inquiry by axis
'* Copyright (C) 1994 I.A.I. Corporation
'*
'* Sales Engineering Department
'* PNAME "S_IAGQ"
'*************************************************************


     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1                  'open communication line

START:
        WRCVTXT =""
        LOCATE 1, 1, 1
        PRINT "SERVO PARAMETER"
        LOCATE 3, 1, 1
        PRINT "AXIS NUMBER AXIS NAME   SERVICE NUMERATOR DENOMINATOR   OVER";
        PRINT "JOG   POSITIONING   SOFT     SOFT     SOFT LIMIT   ACCELERATION SPEED";
        PRINT "      NO.               SPEED                           WRITE";
        PRINT "SPEED RANGE         LIMIT +  LIMIT-   OFFSET                 ";
        PRINT "        TIMES/s                                                   ";
        PRINT "(mm/s)(PULSE)       (mm)     (mm)     (mm)         (g)      ";
        PRINT #1, "?99STA@@"                                     'axis status inquiry command
        LINE INPUT #1, WRCVTXT                                   'receive response
                   WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
                   FOR I = 0 TO VAL(MID$(WRCVTXT, 7, 1)) - 1
                   PRINT #1, "?99IAG" + CHR$(I + 48) + "@@"      'send inquiry command
                   LINE INPUT #1, WRCVTXT                        'receive response
                   LOCATE I * 2 + 6, 2, 1
                   PRINT MID$(WRCVTXT, 7, 1)                     'axis number
                   LOCATE I * 2 + 6, 7, 1
                   PRINT MID$(WRCVTXT, 8, 1)                     'axis name
                   LOCATE I * 2 + 6, 12, 1
                   PRINT MID$(WRCVTXT, 9, 4)                     service speed
                   LOCATE I * 2 + 6, 19, 1
                   PRINT MID$(WRCVTXT, 13, 4)                    'numerator
                   LOCATE I * 2 + 6, 26, 1
                   PRINT MID$(WRCVTXT, 17, 4)                    'denominator
                   LOCATE I * 2 + 6, 33, 1
                   PRINT MID$(WRCVTXT, 21, 4)                    'overwrite
                   LOCATE I * 2 + 6, 40, 1
                   PRINT MID$(WRCVTXT, 25, 4)                    'jog speed
                   LOCATE I * 2 + 6, 47, 1
```

6. Program Sample

```
PRINT MID$(WRCVTXT, 29, 4)                              'positioning range
LOCATE I * 2 + 6, 54, 1
     PRINT MID$(WRCVTXT, 33, 4)                         'soft limit+
     LOCATE 1 * 2 + 6, 61, 1
     PRINT MID$(WRCVTXT, 37, 4)                         'soft limit-
     LOCATE 1 * 2 + 6, 68, 1
     PRINT MID$(WRCVTXT, 41, 5)                         'soft limit offset
     LOCATE 1 * 2 + 6, 75, 1
     PRINT MID$(WRCVTXT, 46, 4)                         'accleration speed
        NEXT I
      CLOSE #1                                          'close communication line
END
```

6.2-5 Axis Status Inquiry Program

```
'*****************************************************************
'*  SERVO AXIS STATUS GET SAMPLE
'*
'*                    PROGRAM For QB45
'*  SERVO AXIS STATUS CHECK (INQUIRY) PROGRAM
'*
'*  This program executes axis status inquiry
'*
'*  Copyright (C) 1994 I.A.I. Corporation
'*  Sales Engineering Department
'*
'*  PNAME "S_STAQ"
'*****************************************************************
     DEFINT I-J: DEFSTR W

     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1        'open communication line

     LOCATE 1, 5, 1
     PRINT "AXIS STATUS"
     LOCATE 3, 9, 1
     PRINT "AXIS NUMBER   SERVO   .  HOME";
     PRINT "MOVE   ERROR CODE   CURRENT POSITION";
     PRINT #1, "?99STA@@"                              'axis status inquiry command
     LINE INPUT #1, WRCVTXT                            'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

     FOR I = 1 TO VAL(MID$(WRCVTXT, 7, 1))
       LOCATE I * 2 + 2, 10, 1
       PRINT "("; I; ")"                               'axis number
       LOCATE 1 * 2 + 2, 20, 1
       PRINT MID$(WRCVTXT, I * 14 - 6, 1)              'servo
       LOCATE I * 2 + 2, 30, 1
       PRINT MID$(WRCVTXT, I * 14 - 5, 1)              'home
       LOCATE I * 2 + 2, 40, 1
       PRINT MID$(WRCVTXT, I * 14 - 4, 1)              'move
       LOCATE I * 2 + 2, 50, 1
       PRINT MID$(WRCVTXT, I * 14 - 3, 2)              'error code
       LOCATE I * 2 + 2, 60, 1
       PRINT MID$(WRCVTXT, I * 14 - 1, 9)              'current position
     NEXT I
   CLOSE #1                                            'close communication line
  END
```

6.2-6 Task Status Inquiry Program

```
'***************************************************************
'* TASK STATUS GET SAMPLE
'*
'*                    PROGRAM For QB45
'*TASK STATUS CHECK (INQUIRY) PROGRAM
'*
'* This program executes task status inquiry
'*
'* Copyright (C) 1994 I.A.I. Corporation
'*
'* Sales Engineering Department
'*
'* PNAME "S_TSKQ"
'***************************************************************
     DEFINT I-J: DEFSTR W

     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1              'open communication line

   LOCATE 1, 5, 1
   PRINT "TASK STATUS"
   LOCATE 2, 9, 1
   PRINT "TASK NUMBER  STATUS  LEVEL  P.N.";
   PRINT "    TASK NUMBER  STATUS  LEVEL P.N.";
   PRINT #1, "?99TSK@@"                                      'axis status inquiry command
   LINE INPUT #1, WRCVTXT                                    'receive response
   WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

   FOR I = 0 TO INT(VAL(MID$(RSCVTXT, 7, 2)) / 2) - 1       'number of axes
     FOR J = 0 TO 1
       LOCATE 1 * 2 + 4, J * 33 + 10, 1
       PRINT "("; I * 2 + J + 1; ")"                         'task number
       LOCATE I * 2 + 4, J * 33 + 20, 1
       PRINT MID$(WRCVTXT, (I * 2 + J) * 4 + 9, 1)          'status
       LOCATE I * 2 + 4, J * 33 + 26, 1
       PRINT MID$(WRCVTXT, (I * 2 + J) * 4 + 10, 1)         'level
       LOCATE I * 2 + 4, J * 33 + 33, 1
       PRINT MID$(WRCVTXT, (I * 2 + J) * 4 + 11, 2)         'program No
     NEXT
   NEXT
  CLOSE #1                                                   'close communication line
  END
```

6.2-7 Program Status Inquiry Program

```
'  ***********************************************
'  *   PROGRAM STATUS GET SAMPLE
'  *                       PROGRAM For QB45
'  *
'  *   PROGRAM STATUS CHECK (INQUIRY PROGRAM)
'  *   This program executes program status inquiry.
'  *   Copyright (C) 1994 I.A.I. Corporation
'  *   Sales Engineering Department
'  *   PNAME "S_PRGQ"
'  ***********************************************

     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1                'open communication line

     WPNE = "THERE IS NO PROGRAM"
     PRINT #1, "?99IPG@@"                                      'program parameter inquiry
     LINE INPUT #1, WRCVTXT                                    'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT = WRCVTXT + WR
     PMAX = VAL(MID$(WRCVTXT, 11, 2))                          'check no. of points
                    L1 = 0
                                 WHILE L1 = 0
                    L2 = 0
                                 WHILE L2 = 0
     LOCATE 1, 5, 1
     PRINT "INPUT PROGRAM NO."
     PRINT "(0-"; PMAX; ",END=-1)
     LOCATE 1, 60, 1
     INPUT WPNUM: PNUM = VAL(WPNUM)
     IF PNUM < 0 THEN END
     CLS
     IF PNUM < PNUM THEN LOCATE 2, 30, 1: PRINT "OUT OF RANGE" ELSE L2 = 1
     WEND
     LOCATE 5, 15, 1
     PRINT "PROGRAM NUMBER ="
     LOCATE 7, 15, 1
     PRINT "STATUS        ="
     LOCATE 9, 15, 1
     PRINT "ERROR CODE    ="
     LOCATE 11, 15, 1
     PRINT "STEP NUMBER   ="
     WPNUM = STRING$(2 - LEN(WPNUM), "0") + WPNUM
     PRINT #1, "?99PRG" + WPNUM + "@@"                         'send inquiry command
     LINE INPUT #1, WRCVTXT                                    'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
     IF WRCVTXT = "%9910@@" THEN LOCATE 2, 24, 1: PRINT WPNE
     LOCATE 5, 44, 1
     PRINT MID$(WRCVTXT, 7, 2)                                 'program number
     LOCATE 7, 44, 1
     PRINT MID$(WRCVTXT, 9, 1)                                 'status
     LOCATE 9, 44, 1
     PRINT MID$(WRCVTXT, 10, 2)                                'error code
     LOCATE 11, 44, 1
     PRINT MID$(WRCVTXT, 12, 4)                                'step number

   WEND
  CLOSE #1                                                     'close communication line
END
```

6.2-8 Error Message Inquiry Program

```
*********************************************************************
*  ERROR MESSAGE INQUIRY SAMPLE
*
*                    PROGRAM For QB45
*
*
*  ERROR MESSAGE INQUIRY PROGRAM
*
*  This program executes error message inquiry
*  Copyright (C) 1994 I.A.I. Corporation
*
*  Sales Engineering Department
*  PNAME "S_MSGQ"
*
********************************************************************
     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1        'open communication line

  WNO = "THERE IS NO SUCH ERROR CODE"
  L1 = 0
  WHILE E = 0
     LOCATE 2, 10, 1
     PRINT "INPUT ERROR CODE.";
     PRINT "(END=-1)
     LOCATE 2, 50, 1
     INPUT WECORD                                      'error code input
     IF WECORD = "-1" THEN END
     CLS
     LOCATE 7, 16, 1
     PRINT "ERROR CODE        ="
     LOCATE 9, 16, 1
     PRINT "ERROR MESSAGE     ="
     PRINT #1, "?99MSG" + WECORD + "@@"                'send inquiry command
     LINE INPUT #1, WRCVTXT                            'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
     IF "%" =MID$(WRCVTXT, 1, 1) THEN LOCATE 9, 40, 1: PRINT WNO
     LOCATE 7, 40, 1
     PRINT MID$(WRCVTXT, 7, 2)                         'error code
     LOCATE 9, 40, 1
     PRINT MID$(WRCVTXT, 9, 16)                        'message
   WEND
  CLOSE #1                                             'close communication line
  END
```

6.2-9 Homing Program

```
***********************************************************
*  HOME POSITION RETURN SAMPLE
*                     PROGRAM For QB45
*
*
*  HOME POSITION RETURN PROGRAM
*  This program executes homing.
*  Copyright (C) 1994 I.A.I. Corporation
*
*  Sales Engineering Department
*  PNAME "Z_HOMQ"
***********************************************************

     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1            'open communication line

   PRINT "INQUIRING AXIS DATA"
   WSRVTXT = "?99STA2@@"                                   'axis parameter inquiry
   GOSUB SND
   AXIS = VAL(MID$(WRCVTXT, 7, 1))                         'read number of axes
   J = 1                                                   'convert to axis pattern
   FOR I = 1 TO AXIS
     A = A + J
     J = J * 2
   NEXT
   WAXIS = HEX$(A)
   IF LEN(WAXIS) = 1 THEN WAXIS = "0" + WAXIS
   PRINT "EXECUTING HOMING"
   WSRVTXT = "!99HOM" + WAXIS + "40@@"                     'send homing command
   GOSUB SND
   L1 = 1
   WHILE L1 <> 0                                           'check homing
     L1 = AXIS
     WSRVTXT = "?99STA@@"                                  'axis parameter inquiry
     GOSUB SND
     FOR I = 1 TO AXIS
       L1 = L1 - VAL(MID$(WRCVTXT, 1 * 14 - 5, 1))
     NEXT
   WEND
   PRINT "HOMING COMPLETE"
  CLOSE #1                                                 'close communication line
  END                                                      'complete
' *************************
' **   RECEIVE RESPONSE   **
' *************************
SND:
  PRINT #1, WSRVTXT
  LINE INPUT #1, WRCVTXT
   WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                     'error check
LOCATE 22, 30, 1 PRINT "ERROR RESPONSE", WRCVTXT           'error processing
BEEP
END
```

6.2-10   Move to Specified Position Program

```
'****************************************************************************
'* MOVE TO COMMAND POSITION SAMPLE
'*
'*                    PROGRAM For QB45
'*
'*
'* MOVE TO COMMANDED POSITION PROGRAM
'*
'* This program executes move to specified position
'* Copyright (C) 1994 I.A.I. Corporation
'*
'* Sales Engineering Department
'* PNAME "Z_MOVQ"
'*
'****************************************************************************
     DEFINT I-J: DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600, N,8,1,LF" FOR RANDOM AS #1                'open communication line

   GOSUB STA                                                    'axis data inquiry
   GOSUB SRV                                                    'servo ON
   GOSUB HOM                                                    'homing
   GOSUB MOV                                                    'specified move
   LOCATE 22, 30, 1
   PRINT "ACTUATOR STOP   "
   WSRVTXT = "!99HLT" + WAXIS + "@@"                            'stop
   GOSUB SND
   LOCATE 22, 30, 1
   PRINT "SERVO OFF"
   WSRVTXT = "!99SRV" + WAXIS + "0@@"                           'servo OFF
   GOSUB SND
  CLOSE #1
  END                                                           'complete
' **************************
' **     AXIS INQUIRY     **
' **************************
STA:
  LOCATE 22, 30, 1
  PRINT ""
  WSRVTXT = "?99STA@@"                                          'axis parameter inquiry
  GOSUB SND
  AXIS = VAL(MID$(WRCVTXT, 7, 1))                               'read number of axes
  J = 1                                                         'convert to axis pattern
  FOR I = 1 TO AXIS
    A = A + J
    J = J * 2
  NEXT
  WAXIS = HEX$(A)
  IF LEN(WAXIS) = 1 THEN WAXIS = "0" + WAXIS
  RETURN
' **********************
' **     SERVO ON     **
' **********************
SRV:
  LOCATE 22, 30, 1
```

6. Program Sample

```
    PRINT "SERVO CHECK"
     WSRVTXT = "?99STA@@"                                   'axis parameter check
     GOSUB SND
    J = 1
    FOR I = 1 TO AXIS
    IF MID$(WRCVTXT, I * 14 - 6, 1) = "1" THEN GOTO SKIP
                                                            'servo check
     WSRVTXT = STRING$(2 - LEN(HEX$(J)), "0") + HEX(J)
     WSRVTXT = "!99SRV" + WSRVTXT + "1@@"
     GOSUB SND                                              'servo ON
     J = J * 2
     LOCATE 22, 30, 1
     PRINT "SERVO ON"
 SKIP:
    NEXT
    RETURN
' ******************
' **    HOMING    **
' ******************
 HOM:
    GOSUB JPS
    IF L1 = 0 THEN GOTO GEND
    LOCATE 22, 30, 1
    PRINT "EXECUTING HOMING"
    WSRVTXT = "!99HOM" + WAXIS + 40@@"                      'homing command
    GOSUB SND
    L1 = 1
    WHILE L1 <> 0
     GOSUB JPS
    WEND
 GEND:
    LOCATE 22, 30, 1
    PRINT "HOMING COMPLETE"
    RETURN
' ************************
' **    CHECK HOMING    **
' ************************
 JPS:
    L1 = AXIS
    WSRVTXT = "?99STA@@"                                    'axis parameter inquiry
    GOSUB SND
    FOR I = 1 TO AXIS
     L1 = L1 - VAL(MID$(WRCVTXT, I * 14 - 5, 1))
    NEXT
    RETURN
' ***************************************
' **    MOVE TO SPECIFIED POSITION    **
' ***************************************
 MOV:
    RANDOMIZE TIME / 4
    LOCATE 2, 14, 1
    PRINT"ACTUATOR SPECIFIED MOVE (END WITH A SPACE)  "
    LOCATE 4, 18, 1
```

```
   PRINT "AXIS NUMBER   SPECIFIED POSITION   CURRENT POSITION"
   FOR I = 1 TO AXIS
     LOCATE 1 * 2 + 4, 20, 1
     PRINT "ACTUATOR (": I: ")"
   NEXT
   L1 = 0
   L2 = 0
   WHILE L1 = 0
     J = 1
     AST = 0
     FOR I = 1 TO JIKU
     WSRVTXT = "?99STA@@"                                   'axis parameter inquiry
     GOSUB SND
     LOCATE I * 2 + 4, 50, 1
     PRINT MID$(WRCVTXT, I * 14 - 1, 9)                     'display current position
     IF MID$(WRCVTXT, I * 14 - 4, 1) = 0 THEN GOSUB SET
       J = J * 2
     NEXT
     LOCATE 22, 30, 1
     PRINT "ACTUATOR IN OPERATION"
     IF INKEY$ = "" THEN L2 = 1
     IF AST = AXIS THEN L1 = 1                              'confirm stop
   WEND
   RETURN
' ***********************
' **    SET POSITION    **
' ***********************
SET:
   IF L2 = 1 THEN AST = AST + 1: RETURN
   WSRVTXT = "?99IAG" + CHR$(I + 47) + "@@"                 'servo parameter inquiry
   GOSUB SND
   LIMIT = VAL(MID$(WRCVTXT, 3, 4))                         'read soft limit
   WA = MID$(STR$(INT(RND * LIMIT)), 2, 8)
   WA = STRING$(3 - LEN(WA), "0") + WA                      'create specified position
   WJP = STRING$(2 - LEN(HEX$(J)), "0") + HEX$(J)
   WSRVTXT = "!99MOV" + WJP + "0.100100" + WA+"     @@"
                                                            'send specified move command
   GOSUB SND
   LOCATE 22, 30, 1
   PRINT "SET POSITION"
   LOCATE I * 2 + 4, 38, 1
   PRINT WA                                                 'display specified position
   RETURN
' ***************************
' **    RECEIVE RESPONSE    **
' ***************************
SND:
   PRINT #1, WSRVTXT
   LINT INPUT #1: WRCVTXT                                   'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

     IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                 'error check
     LOCATE 22, 30, 1
     PRINT "RESPONSE ERROR", WRCVTXT                        'error processing
     BEEP
     END
```

6. Program Sample

6.2-11 Point Number Specified Move Program

```
*****************************************************************************
* MOVE WITH POINT NUMBER SAMPLE
*
*                    PROGRAM For QB45
*
*
* POINT NUMBER MOVE PROGRAM
*
* This program executes point number specified move
*
* Copyright (C) 1994 I.A.I. Corporation
*
* Sales Engineering Department
*
* PNAME "Z_PMVQ"
*
*****************************************************************************

     DEFINT I-J:  DEFSTR W
     CLS
RSINI:
     OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS#1              'open communication line

   GOSUB STA                                                'axis parameter inquiry
   GOSUB SRV                                                'servo ON
   GOSUB HOM                                                'homing
   GOSUB PMV                                                'point specified move
   LOCATE 22, 30, 1
   PRINT "ACTUATOR STOP   "
   WSRVTXT = "!99HLT" + WAXIS + "@@"                        'stop
   GOSUB SND
   WSRVTXT = "!99SRV" + WAXIS + "00@@"                      'servo OFF
   GOSUB SND
  CLOSE #1                                                  'close communication line
  END                                                       'complete
' ***********************
' **    AXIS INQUIRY    **
' ***********************
STA:
  LOCATE 22, 30, 1
  PRINT "INQUIRING AXIS DATA"
  WSRVTXT = "?99STA@@"                                      'axis parameter inquiry
  GOSUB SND
  AXIS = VAL(MID$(WRCVTXT, 7, 1))                           'read number of axes
  J = 1                                                     'convert to axis pattern
  FOR I = 1 TO AXIS
   A = A + J
   J = A * 2
  NEXT
  WAXIS = HEX$(A)
  IF LEN(WAXIS) = 1 THEN WAXIS = "0" + WAXIS
  RETURN
```

6. Program Sample

```
'*********************
'**    SERVO ON    **
'*********************
SRV:
  LOCATE 22, 30, 1
  PRINT ""
  WSRVTXT = "?99STA@@"                                        'axis parameter inquiry
    GOSUB SND
    WSTA = WRCVTXT
    J = 1
    FOR I = 1 TO AXIS
      IF MID$(WSTA, I * 14 - 6, 1) = "1" THEN GOTO SKIP       'servo check
      WSRV = STRING$(2 - LEN(HEX$(J)), "0" + HEX$(J)
      WSRVTXT = "!99SRV" + WSRV + "1@@"                       'servo ON
      GOSUB SND
      J = J * 2
      LOCATE 22, 30, 1: PRINT "SERVO ON
  SKIP:
      NEXT
      RETURN
' ******************
' **    HOMING    **
' ******************
HOM:
      GOSUB JPS
      IF L1 = 0 THEN GOTO GEND
      LOCATE 22, 30, 1
      PRINT "EXECUTING HOMING"
      WSRVTXT = "!99HOM" + WJIKU + "40@@"                     'homing command
      GOSUB SND
      L1 = 1
      WHILE L1 <> 0
        GOSUB JPS
      WEND
    GEND:
      LOCATE 22, 30, 1
      PRINT "HOMING COMPLETE    "
      RETURN
' ************************
' **    CHECK HOMING    **
' ************************
    JPS:
      L1 = AXIS
      WSRVTXT = "?99STA@@"                                    'axis parameter inquiry
      GOSUB SND
      FOR I = 1 TO AXIS
        L1 = L1 - VAL(MID$(WRCVTXT, I * 14 - 5, 1))
      NEXT
      RETURN
```

6. Program Sample

```
' ***************************************
' **    POINT NUMBER SPECIFIED MOVE    **
' ***************************************
   PMV:
     WSRVTXT = "?99IPO@@"                                 'point parameter inquiry
     GOSUB SND
     PMAX = VAL(MID$(WRCVTXT, 7, 4))                      'read number of points
     L1 = 0
     WHILE L1 = 0
       LOCATE 2, 14, 1
       PRINT ""
       PRINT "(0 ": PMAX: ",END=-1)"
     LOCATE 4, 18, 1
     INPUT PN                                             'input point number
     IF PN < 0 THEN RETURN
     WPN = MID$(STR$(PN), 2, 4)
     WPN = STRING$(4 - LEN(WPN), "0") + WPN
     WSRVTXT = "!99PMV0300000000" + WPN + "@@"            'send point move command
     GOSUB SND
     LOCATE 22, 30, 1
     PRINT "IN MOTION"
     L2 = 1
     WHILE L2 <> 0
     WSRVTXT = "?99STA@@"                                 'axis status inquiry
     GOSUB SND
       L2 = 0
       FOR J = 1 TO AXIS
             L2 = L2 + VAL(MID$(WRCVTXT, J * 14 - 4, 1))  'confirm stop
       NEXT
     WEND
     CLS
     LOCATE 22, 30, 1
     PRINT "MOVE COMPLETE"
   WEND
   RETURN
' ****************************
' **    RECEIVE RESPONSE    **
' ****************************
SND:
   PRINT #1, WSRVTXT
   LINE INPUT #1, WRCVTXT                                 'receive response
     WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

   IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                 'error check
   LOCATE 22, 30, 1
   PRINT "", WRCVTXT                                      'error processing
   BEEP
   END
```

6. Program Sample

6.2-12 Chosen Program Execution Program

```
**************************************************************
*
*  CHOSEN PROGRAM EXECUTE SAMPLE
*                    PROGRAM For QB45
*
*
*  CHOSEN PROGRAM EXECUTION PROGRAM
*
*  This program executes specified program
*
*  Copyright (C) 1994 I.A.I. Corporation
*
*  Sales Engineering Department
*
*  PNAME "Z_RUNQ"
**************************************************************
   DEFINT I-J: DEFSTR W
   CLS
RSINI:
   OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1                   'open communication line

WPNE = "CANNOT FIND PROGRAM"
  WSRVTXT = "?99IPG@@"                                          'program parameter inquiry
  GOSUB SND
  PMAX = VAL(MID$(WRCVTXT, 11, 2))                              'confirm number of programs
RLOOP:
  L1 = 0
  WHILE L1 = 0
    PRINT "EXECUTE PROGRAM NO.? (0-": PMAX: ")"
    INPUT PN                                                    'input program number
    IF PN < 0 OR PMAX <.PN THEN PRINT "OUTSIDE OF RANGE" ELSE L1 = 1
  WEND
  PRINT "EXECUTE PROGRAM"
  WPN = MID$(STR$(PN), 2, 2)
  WPN = STRING$(2 - LEN(WPN), "0") + WPN
  WSRVTXT = "!99RUN" + WPN + "@@"                               'execute program
  GOSUB SND
  PRINT "PROGRAM ENDS AT SPACE."
  L1 = 0
  WHILE L1 = 0
    IF INKEY$ = "" THEN L1 = 1
  WEND
WSRVTXT = "!99EXT" + WPN + "@@"                                 'stop program
GOSUB SND
PRINT "PROGRAM COMPLETE"
CLOSE #1                                                        'close communication line
END                                                             'complete
'  ****************************
' **    RECEIVE RESPONSE    **
'  ****************************
SND:
  PRINT #1, WSRVTXT
  LINE INPUT #1, WRCVTXT                                        'receive response
    WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                          'error check
LOCATE 22, 30, 1
PRINT  "RESPONSE ERROR  =" , WRCVTXT                            'error processing
BEEP
END
```

6.2-13 Point Data Set Program

```
***********************************************************
*
*  POINT DATA SET SAMPLE
*
*                      PROGRAM For QB45
*
*
*  POINT DATA  SET PROGRAM
*  This program executes point data set.
*
*  Copyright (C) 1994 I.A.I. Corporation
*
*  Sales Engineering Department
*
*  PNAME "Z_PSEQ"
*
***********************************************************


    DEFINIT I-J: DEFSTR W
    CLS
RSINI:
    OPEN "COM41:9600,N,8,1,LF" FOR RANDOM AS #1                      'open communication line
  L1 = 0
  WHILE L1 = 0
    PRINT "POINT NUMBER   (END=-1) ="
    INPUT PN                                                         'input point number
  IF PN < 0 THEN L1 = 1: GOTO LEND                                   'confirm complete
    IF 9999 < PN THEN PRINT "OUTSIDE OF RANGE.": GOTO LEND
    WPN = MID$(STR$(PN), 2, 4)
    WPN = STRING$(4 - LEN(WPN), "0" + WPN
    PRINT "AXIS PATTERN .      =";                                   'input axis pattern
    INPUT WJP
    WJP = STRING$(2 -LEN(WJP), "0") + WJP
    WSRVTXT = "?99ISV@@"                                             'servo parameter inquiry
    GOSUP SND
    WKM = MID$(WRCVTXT, 32, 4)                                       'read max. acceleration speed
    PRINT "ACCELERATION SPEED (g)  (MAX " + WKM + ") ="              'input acceleration speed
    INPUT KD
    WKD = MID$(STR$(KD), 2, 4)
    WKD = STRING$(4 - LEN(WKD), "0") + WKD
    WSM = MID$(WRCVTXT, 24, 4)                                       'read maximum speed
    PRINT "SPEED  (mm/sec) (MAX " + WSM + ") ="                      'input speed
    INPUT SD
    WSD = MID$(STR$(SD), 2, 4)
    WSD = STRING$(4 - LEN(WSD), "0") + WSD
    JP = VAL("&h" + WJP)
    WITI = ""                                                        'initialize position data
    J = 1
    FOR I = 1 TO 8
      IF (JP AND J)<> 0 THEN GOSUB ITI                               'input position data
      J = J * 2
    NEXT
    PRINT   (Y/N)"
    INPUT WA
    IF WA = "N" THEN GOTO LEND
    WSRVTXT = "!99PSE" + WPN + WJP + WKD + WSD + WITI + "@@"         'point data set
```

```
     GOSUB SND

LEND:
     PRINT ""
   WEND
  CLOSE #1                                                          'close communication line
  END                                                               'complete
' ***************************
' **    RECEIVE RESPONSE    **
' ***************************
SND:
  PRINT #1, WSRVTXT
  LINE INPUT #1, WRCVTXT                                            'receive response
   WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR
  IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                            'error check
  LOCATE 22, 30, 1
  PRINT "RESPONSE ERROR  =:,       WRCVTXT                          'error processing
  BEEP
  END

' *************************
' **    POINT DATA SET    **
' *************************
ITI:
     WSRVTXT = "?99IAG" + .CHR$(I + 47) + "@@"
     GOSUB SND
     WLS = MID$(WRCVTXT, 37, 4)                                     'read smallest value
     WLL = MID$(WRCVTXT, 3, 4)                                      'read largest value
     PRINT I: AXIS POSITION (" + WLS + "-" + LL$ + ")               'input position data
     INPUT ITI
     WITI = WITI + MID$(STR$(ITI), 2, 7)
     WITI = WITI + STRING$(7 - LEN(WIT), "")
     RETURN
```

6.2-14 Output Port Set Program

```
***********************************************************
* OUTPUT PORT SET
*
*                    PROGRAM For QB45
*
*
* OUTPUT PORT SET PROGRAM
*
* This program sets output ports.
*
* Copyright (C) 1994 I.A.I. Corporation
*
* Sales Engineering Department
*
* PNAME "Z_OTSQ"
*
***********************************************************

  DEFINIT I-J:  DEFSTR W
  CLS
RSINI:
  OPEN "COM1:9600,N,8,1,LF" FOR RANDOM AS #1                'open communication line

  PRINT "SET OUTPUT PORT (END WITH A SPACE.)"
  L1 = 0
  WHILE L1 = 0
    FOR I = 0 TO 2
      WNUM = MID$(STR$(I), 2, 2)
      WNUM = STRING$(2 - LEN(WNUM), "0") + WNUM              'set number
      K = 1
      FOR J = 0 TO 7
          WDAT = HEX$(K)
          WDAT = STRING$(2 - LEN(WDAT), "0") + WDAT          'set data
          WSRVTXT = "!99OTS" + WNUM + WDAT + "@@"            'output port set
          GOSUB SND
          FOR L = 0 TO 500: NEXT                             'wait
          K = K * 2
      NEXT
      WSRVTXT = "!99OTS" + WNUM + "00@@"
      GOSUB SND
    NEXT
    IF 1NKEY$ = "" THEN L1 = 1                               'determined complete
  WEND
CLOSE #1                                                     'close communication line
END                                                          'complete
' ***************************
' **    RECEIVE RESPONSE    **
' ***************************

SND:
  PRINT #1, WSRVTXT
  LINE INPUT #1, WRCVTXT                                     'receive response
    WR = INPUT$(1, #1): WRCVTXT = WRCVTXT + WR

    IF LEFT$(WRCVTXT, 1) = "#" THEN RETURN                   'error check
    LOCATE 22, 30, 1
    PRINT          WRCVTXT                                   'error processing
    BEEP
    END
```