

61 Stream motion

The stream motion is a feature that controls the robot according to the path planned by arbitrary external device by transmitting the desired position data in real-time (usually 8 ms interval) via Ethernet. This feature is used for directly specifying the path, that is difficult to be formed with ordinal motion instructions such as LINEAR move or JOINT move, from an external device, or for applications which requires online path generation based on such as sensor data.

To use this feature, the stream motion (A05B-2600-J519) option is necessary.

In addition, the customer is required to develop software that can generate path in consideration of robot-specific constraints such as the upper limit of speed, acceleration, and jerk. (For more details, please refer to “61.5 Conditions for executable motion commands”).

61.1 Setup procedure

Cable Connection

Connect the the shielded Category 5 or more twisted-pair cable (STP cable) to a physical port (CD38A or CD38B), according to “D. CABLE CONNECTION” in “Ethernet function OPERATOR’S MANUAL (B-82974)”. Strip a part of the sheath to expose the metal shield, push the shield against the grounding plate, and then clamp with the bracket. Since this feature sends data using the UDP protocol, noise suppression is very important.

Communication settings

Set the system variable \$STMO.\$PHYS_PORT according to the physical port number used. (1: CD38A, 2:CD38B)

About details of communication settings in robot controller, please refer to “2.1 OVERVIEW”, “2.2 HARDWARE REQUIREMENTS AND INSTALLATION”, and “2.4 SETTING UP TCP/IP” in “Ethernet function OPERATOR’S MANUAL (B-82974)”

Program Creation

To start the robot control with this feature, the robot controller need to be running a program includes a specific instruction.

In the program edit screen, select the control instructions “ASCII interface”, and specify the following instructions (IBGN start [*], IBGN end [*]) included in that item. For the general procedure for creating and editing a program, please refer to “5 PROGRAMMING” in “OPERATOR’S MANUAL (Basic Function) (B-83284EN)”.

IBGN start[*]

* : Set an arbitrary integer of from 1 to 16.

While executing this instruction, the robot moves based on the desired position data sent from external device.

IBGN end[*]

* : Set the same value as the integer set in IBGN start[*].

When the control from the external device is finished, execution moves to this instruction and continues.

The following is an example of creating a program.

```

STREAM_TEST
5/5
1: OVERRIDE=100%
2: J @P[1] 100% FINE
3: IBGN start[1]
4: IBGN end[1]
[End]

```

Note

1 IBGN start [*] must be specified in the line before the paired IBGN end [*].

2 If there are instructions between IBGN start [*] and IBGN end [*], they will not be executed.

61.2 Execution of Stream motion

Execute the program created in “61.1 Setup procedure” in AUTO mode with 100% OVERRIDE.

When the program lines are executed from the top and the cursor moves to the IBGN Start [*] instruction, the robot is ready to receive the desired position data sent from external device. After that, start communication according to “61.3 Communication Protocol”.

If you need to start the program via Ethernet, please refer to “55 HMI DEVICE COMMUNICATION” in “OPERATOR'S MANUAL (Optional Function) (B-83284EN-2)”.

WARNING

- 1 This feature cause unintended motion if there is an error in the command from the external device, and can cause severe personal injury as the result of the worst case.
- 2 Use a simulation software such as ROBOGUIDE and check the operation sufficiently in advance.
- 3 Consider using it in combination with other safety software such as “DCS position / speed check”.

61.3 Communication Protocol

Communicate between the external device and the robot controller (hereinafter referred to as the robot) according to the following protocol.

The UDP packet in big endian byte order is used to transfer data, and the robot side uses 60015 for port address.

The communication interval depends on the robot model and is 8ms or 4ms. (For 4ms models, refer to “61.8 support models”).

Status Packet

Send a packet of “Table 1: Status output start packet” from the external device to the robot. After that, a packet of “Table 2: Status packet” is sent from the robot to the external device using the same socket every communication interval (usually 8ms). The status packets have information about the current robot position, motor current value, etc.

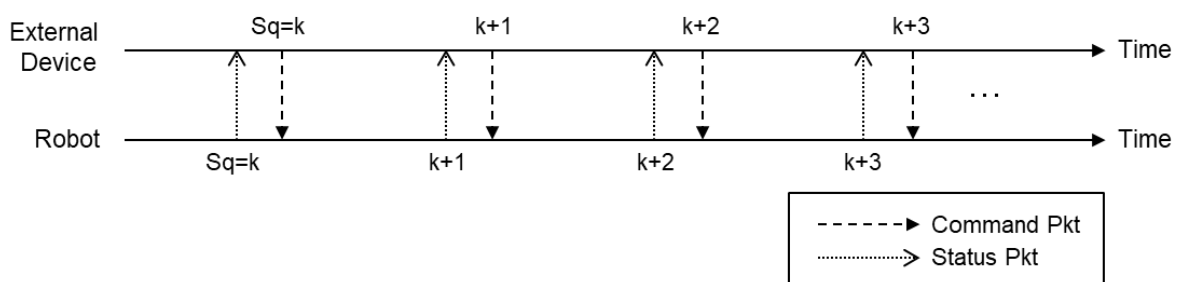
To stop the output of the status packets, send the packet of “Table 4: Status output stop packet” from the external device to the robot. The communication of status packet described above can be started/ended at any time without executing the IBGN start [*] instruction.

Command packet

After starting execution of the IBGN start [*] instruction, confirm that bit 0 (LSB) of the “status” data contained in the status packet has changed from 0 to 1. Then, set the desired position of the robot to the packet according to “Table 3: Command packet”. Send it to the robot to start controlling the robot.

After receiving the first command packet, the robot continues to be controlled by continuously reading the desired position from the subsequent command packet for each communication interval. As a guide, each time the external device receives one status packet from the robot, immediately send the next command packet to the robot.

The figure below shows communication started from the sequence number k.



To terminate this communication, set the value of “Last data” included in the command packet to 1 and send it to the robot. When the robot processes this command packet, this communication ends normally.

Note

- 1 An alarm will be posted if the robot cannot receive the second and subsequent command packets within the communication interval.
- 2 An alarm may occur if the status packet communication is terminated before the command packet communication is terminated.
- 3 It is highly recommended to use a real-time OS for accurate response within the communication interval.
- 4 If the total communication rate of Ethernet Port 1-2 exceeds 1 packet/ms, communication error may occur.

Table 1 : Status output start packet (External Device → Robot)

Data type	Name	Description
4 byte Unsigned integer	Packet type	Set 0 : Status output start packet.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. Set 1.

Table 2 : Status packet (Robot → External Device)

Data type	Name	Description
4 byte Unsigned integer	Packet type	0 is set.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. 1 is set.
4 byte Unsigned integer	Sequence No.	Starts from 1 and incremented 1 each transmission. 0xFFFFFFFF is followed by 0. Sending the status output start packet resets this value to 1.
1 byte Unsigned integer	Status	Bit 0 (LSB): shows the waiting status for receiving the command packet of the table 3. (It becomes 1 at the start of execution of IBGN start[*] instruction, and becomes 0 when the program is stopped or the last data flag in command packets)
		Bit 1 : Changes to 1 when command packet is received by the robot. Changes to 0 at the same timing that Bit0 is changed to 0.
		Bit 2 : Shows SYSRDY status (1:SYSRDY ON, 0:SYSRDY OFF)
		Bit 3 : Changes to 1 when the robot start moving and 0 when stopped.
		Bit 4-7: Unused
1 byte Unsigned integer	Read I/O type	The same value set for the read I/O type in Table 3.
2 byte Unsigned integer	Read I/O index	The same value set for the read I/O index in Table 3.
2 byte Unsigned integer	Read I/O mask	The same value set for the read I/O mask in Table 3.
2 byte Unsigned integer	Read I/O value	The read I/O value. Represents the state of I/O for 16 consecutive points. Each bit of this data represents ON(1), OFF(0) of each I/O, and LSB is the I/O set by the read I/O index. 0 is set while no command packet is sent. Example 1: 1 (0x1) → Only the first point is ON Example 2: 255 (0xFF) → All the first 8 points are ON Example 3: 65535 (0xFFFF) → All 16 points are ON
4 byte Unsigned integer	Time stamp	The time stamp when the robot checks the current position and motor current value. The unit is ms and the resolution is 2ms. 0xFFFFFFFF is followed by 0.
4 byte float	X	The current robot position. The unit is mm.
4 byte float	Y	
4 byte float	Z	
4 byte float	W	The current robot position (Orientation). The unit is degree.
4 byte float	P	
4 byte float	R	

Data type	Name	Description
4 byte float	Extended axis 1	The extended axis position of the robot. The unit is mm for the linear axis and degree for rotary axis.
4 byte float	Extended axis 2	
4 byte float	Extended axis 3	
4 byte float	J1	The current robot position (Joint). The unit is mm for the linear axis and degree for rotary axis.
4 byte float	J2	
4 byte float	J3	
4 byte float	J4	
4 byte float	J5	
4 byte float	J6	
4 byte float	J7	
4 byte float	J8	
4 byte float	J9	
4 byte float	J1	Mortor current of the robot. The unit is A.
4 byte float	J2	
4 byte float	J3	
4 byte float	J4	
4 byte float	J5	
4 byte float	J6	
4 byte float	J7	
4 byte float	J8	
4 byte float	J9	

Table 3 : Command packet (External Device → Robot)

Data type	Name	Description
4 byte Unsigned integer	Packet type	Set 1 : Command packet.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. Set 1.
4 byte Unsigned integer	Sequence No.	For the first command packet that is sent after the robot starts waiting, set the same sequence number as the status packet that was received immediately before from the robot. Increment this value every transmission. 0xFFFFFFFF is followed by 0.
1 byte Unsigned integer	Last data	Normally set to 0. To terminate control of the robot from an external device, set 1 only in the last command packet to be sent. There may be a delay of one packet or more from the transmission of the command packet with 1 set in this data until it is reflected in the "status" of the status packet.
1 byte Unsigned integer	Reading I/O type	DI : 1, DO : 2, RI : 8, RO : 9, SI : 11, SO : 12, WI : 16, WO : 17, UI : 20, UO : 21, WSI : 26, WSO : 27, F : 35, M : 36 Set to 0 if do not need to read I/O.
2 byte Unsigned integer	Reading I/O index	Set the index of I/O to read. Up to 16 consecutive points from the specified index can be read at the same time.
2 byte Unsigned integer	Reading I/O mask	Only the I/O corresponding to the set bit of this data is read from the 16 consecutive points specified by the Reading I/O index. The LSB corresponds to the I/O specified by the Reading I/O index. Example 1: Read only the first point → 1 (0x1) Example 2: Read only the first 8 points → 255 (0xFF) Example 3: Read all 16 points → 65535 (0xFFFF)
1 byte Unsigned integer	Data format	Set the data format of the desired position used in this packet. 0: Cartesian, 1: Joint.
1 byte Unsigned integer	Writing I/O type	DI : 1, DO : 2, RI : 8, RO : 9, SI : 11, SO : 12, WI : 16, WO : 17, UI : 20, UO : 21, WSI : 26, WSO : 27, F : 35, M : 36 Set to 0 if do not need to write I/O.

2 byte Unsigned integer	Writing I/O index	Set the index of I/O to write. Up to 16 consecutive points from the specified index can be read at the same time.	
2 byte Unsigned integer	Writing I/O mask	<p>Only the I/O corresponding to the set bit of this data is written from the 16 consecutive points specified by the Writing I/O index. The LSB corresponds to the I/O specified by the Writing I/O index.</p> <p>Example 1: Write only the first point → 1 (0x1) Example 2: Write only the first 8 points → 255 (0xFF) Example 3: Write all 16 points → 65535 (0xFFFF)</p>	
2 byte Unsigned integer	Writing I/O value	<p>The I/O value to write to the robot. Represents the state of 16 consecutive I/O points. Each bit of this data represents ON (1) and OFF (0) of each I/O, and the LSB corresponds to the I / O specified by the Writing I / O index.</p> <p>Example 1: 1 (0x1) → Turn on only the first point, turn off the others Example 2: 255 (0xFF) → Turn on all the first 8 points, turn off the others Example 3: 65535 (0xFFFF) → Turn on all 16 points Note that, in Examples 1-3, the Writing I/O mask is assumed to be 0xFFFF.</p>	
2 byte	unused		
4 byte float	X or J1	0 : Cartesian format	1 : Joint format
4 byte float	Y or J2	The desired robot position. The unit is mm.	The desired robot position. The unit is degree.
4 byte float	Z or J3		
4 byte float	W or J4	The desired robot position. The unit is degree.	
4 byte float	P or J5		
4 byte float	R or J6	The desired robot position (extended axes). The unit is mm for the linear axis and degree for rotary axis.	
4 byte float	Extended axis 1		
4 byte float	Extended axis 2		
4 byte float	Extended axis 3		

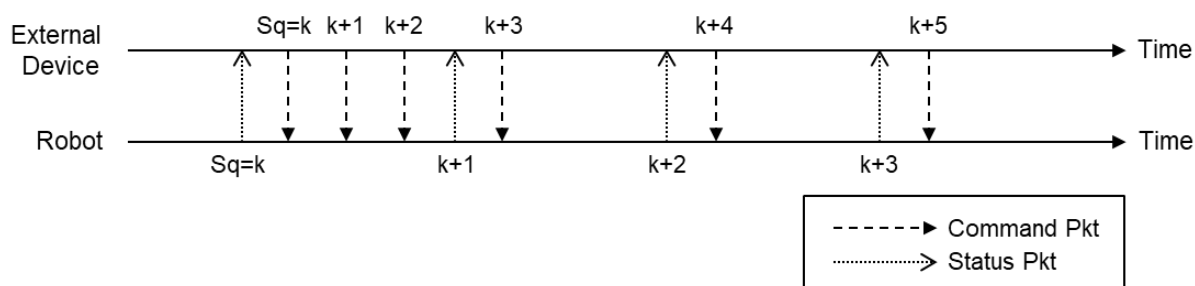
Table 4 : Status output stop packet (External Device → Robot)

Data type	Name	Description
4 byte Unsigned integer	Packet type	Set 2 : Status output stop packet.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. Set 1.

61.4 Buffering command packets

In this feature, if the second and subsequent command packets from an external device arrive at the robot earlier than the communication interval, the packets can be stored in the packet buffer without being processed (Enabled by default). The robot moves by reading the command packets in the buffer in order from the one that arrived first. The packet can be stored up to \$STMO.\$PKT_STACK-1 and an alarm will not be posted even if the robot does not receive a new command packet from the external device as long as there are unprocessed packets inside the buffer.

By using this packet buffer, it is possible to prevent an alarm that occurs when a packet does not arrive within the communication interval due to communication delay. The figure below is an example of controlling the robot while keeping two packets in the buffer at all times by sending two command packets in advance. If a command packet with the last data set to 1 is sent when using the buffer, there will be a time delay until all command packets inside the buffer are processed first.



Also, by changing the value of the system variable (\$STMO.\$START_MOVE), it is possible to configure the robot to start moving after a certain number of unprocessed command packets have accumulated in the packet buffer.

(Available in software versions 7DC3/55, 7DF1/26, 7DF3/05, or later.)

Note

- 1 Only command packets are stored in the buffer. Status output stop packets are processed immediately after they are received.
- 2 Do not send the status output stop packet while the command packet remains in the buffer.

61.5 Conditions for executable motion commands

Reachable Positions

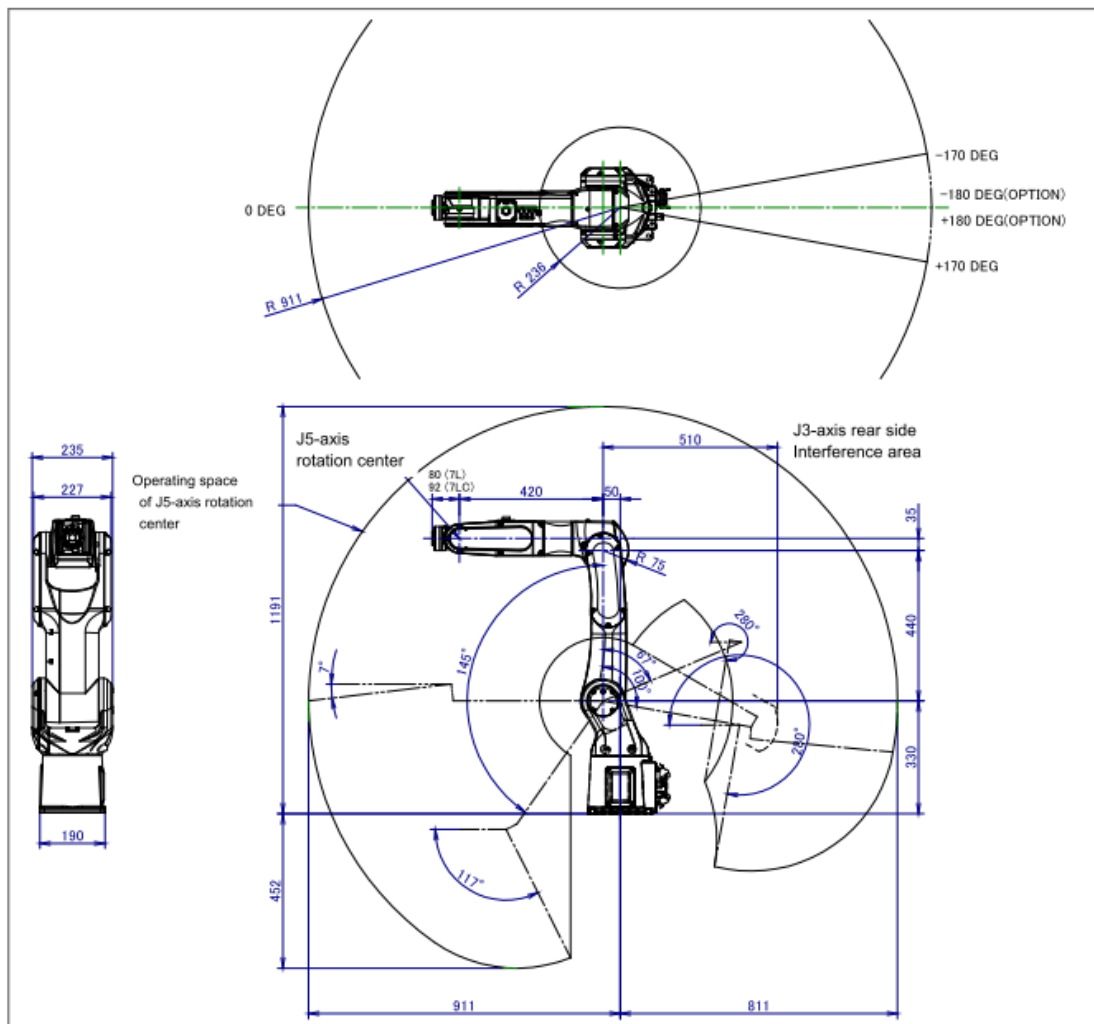
The desired position specified in the command packet must be reachable by the robot. If a command packet to an unreachable position is received, the robot will stop with an alarm.

The reachable desired position must mainly meet the following conditions.

- Joint position of the robot corresponding to the desired position must exist uniquely. (In case of Cartesian format)
- The configuration(*1) of the desired position matches that at the starting of IBGN start [*]. (In case of Cartesian format)
- The desired position satisfies the upper and the lower limit for each robot axis.
- The desired position must not cause the self collision of the robot.

In order to perform a strict check on the above on the external device side, software such as forward kinematics, inverse kinematics, and self-collision checker would be required. FANUC cannot provide the mathematical formulas and know-how regarding these matter.

Therefore, it is necessary for the customer to consider and develop it based on the public information such as the operating space diagram described in the operator's manual of mechanical unit.



Example of the operating space diagram

*1 refer to "4 Program Configuration" in "OPERATOR'S MANUAL (Basic Function) (B-83284EN)".

Definition of J3 axis

In generating the motion command, it is necessary to consider the definition of the J3 axis peculiar to the FANUC robot. The J3 axis definition of the FANUC robot is based on the angle between the horizontal plane as seen by the robot and the J3 arm, not the angle between the J2 arm. Therefore, even if you send a command packet to displace the J2 axis only, the J3 arm will also move in tandem. Please be careful about this specification so that you will not take an unintended posture and an accident will occur. There is no way to change this specification.

Restrictions on speed, acceleration, and jerk

While controlling the robot by sending a command packet, it is necessary to control so that the allowable upper limits of speed, acceleration, and jerk set for each axis of the robot are not exceeded. If a command packet that will violate the upper limit is detected, the robot may stop with an alarm. The purpose of this constraint is to prevent premature damage to the reducer and mechanical parts by continuing to execute excessive motion commands. (Extended axes are not included in the target of this constraint.)

For details on the upper limit, refer to the following system variables that exist under the system variable \$STMO_GRP. They does not depend on the format of the position data. For the lower limit value, add a negative sign to the upper limit values.

- \$JNT_VEL_LIM[*]: Allowable upper limit value for speed for each axis [deg/s]
- \$JNT_ACC_LIM[*]: Allowable upper limit value for acceleration for each axis [deg/s²]
- \$JNT_JRK_LIM[*]: Allowable upper limit of jerk for each axis [deg/s³]
- \$MAX_SPD: Maximum speed of the robot to which the above allowable upper limit is applied [mm/s]

The above system variables are Read-Only and cannot be used by changing their values. In addition, since the value is reference values, depending on the conditions, larger acceleration/jerk upper limit values may be set internally (see “Appendix B Acceleration/jerk upper limit calculation method”). The alarm may not occur even if a command packet that violates the reference values above is received. On the other hand, if the robot is moving at a speed exceeding \$MAX_SPD [mm/s], the robot may stop with an alarm even if the acceleration/jerk of the motion does not exceed the reference values above.

Note

- 1 This restriction cannot be deactivated or relaxed even for special reasons such as experimental purposes.
- 2 For the calculation of velocity, acceleration, and jerk (instantaneous), communication interval (usually 8ms) is used as a fixed time length.
- 3 For J3 axis, the speed, acceleration, and jerk of the J3 axis alone based on the above definition are limited.

Difference between Command position and Servo position

There is a distinction between the command position and the servo feedback position (hereinafter referred to as the servo position) in the current position of the robot. The servo position is controlled to follow the command position as much as possible, but there is a certain error between the two positions. Please note that the current position that can be read in the status packet is the servo position.

The desired position in the command packet needs to be set so that the motion can be started smoothly from the command position of the robot. If a path is generated starting from the servo position or current servo position is set as the desired position of the command packet and transmitted, Acceleration/Jerk may become excessive due to the above-mentioned error, and an unexpected alarm may occur.

As shown in the program example described in “61.1 Setup procedure”, if a Joint move is executed before the IBGN start[*] instruction, the robot can be moved to the taught command position. If it is necessary to acquire the command position of the robot via Ethernet, please refer to “55 HMI DEVICE COMMUNICATION” in “OPERATOR'S MANUAL (Optional Function) (B-83284EN-2)”, separately.

Reducer load

If the load level applied to the reducer unit of the robot exceeds a certain threshold, the robot may be stopped with an alarm. This is because the drive system component such as the reducer unit and gears may be damaged early.

It rarely occurs under conditions that satisfies the constraints of speed, acceleration, and jerk, but please consider measures such as making the robot motion more conservative so that the load on the reducer is reduced. The method of calculating the load level on the reducer unit is confidential and not disclosed.

61.6 System variables

System variable		default	Description
\$STMO <i>Needs a cycle-power after changing the setting.</i>	\$PHYS_PORT	2	Set the physical port number to use. (1: CD38A, 2: CD38B)
	\$COM_INT	8	Communication interval. (Unit:ms, Read-Only)
	\$PKT_STACK	10	This is the maximum number of command packets that can be kept unprocessed. (Minimum value 2, Maximum value 10)
	\$START_MOVE	1	The robot starts moving after unprocessed command packets has accumulated to this value. (Minimum value 1, Maximum value \$ PKT_STACK-1)
\$STMO_GRP <i>Needs a cycle-power after changing the setting.</i>	\$JNT_VEL_LIM[*]		Allowable upper limits [deg/s] for the speed of each axis. (Read-Only)
	\$JNT_ACC_LIM[*]		Allowable upper limits [deg/s^2] for the acceleration of each axis. (Read-Only)
	\$JNT_JRK_LIM[*]		Allowable upper limits [deg/s^3] for the jerk of each axis. (Read-Only)
	\$LMT_MODE	0	Please refer to "Appendix B Calculation method of acceleration / jerk upper limit".
	\$WARN_LIM	80	A warning is posted when one of the speed, acceleration, or jerk of each axis of the robot exceeds the upper limit of \$WARN_LIM[%].
	\$FLTR_LN	1	If a value greater than 1 is set, smoothing by moving average is applied to the desired positions of command packets. The larger the value, the smoother the path, but the greater the deviation from the desired positions of command packet.
	\$MAX_SPD		The maximum speed [mm/s] of the robot to which the allowable upper limit values of acceleration and jerk shown in the above system variables are applied.

61.7 Restrictions

Precautions when creating a program

- In the program that includes the IBGN start[*]/end[*] instructions cannot use the numbers more than 30,000 as the index for the jump/label instruction.
- This feature can be used only with the robots in motion group 1. When using in a system with two or more groups, disable the motion of the second and subsequent groups in the detailed program information settings.

Precautions when executing the program

- If you try to read or write I / O that contains an index that has not been assigned, an alarm will occur.
- When specifying a desired position in Cartesian format, specify the position of the flange center point in the world coordinate system. Neither the tool coordinate system or user coordinate system can be used with this feature.
- If the correct payload is not set according to the weight of work piece, an alarm may occur due to such as false collision detection.
- HOLD stop is not supported during execution of the IBGN start [*] instruction. It is necessary to generate a deceleration path equivalent to HOLD stop by an external device itself and execute it via command packets.

Option software constraints

- This feature cannot be installed at the same time as the ascii interface function (J829).
- This feature cannot be used at the same time as the following option softwares. (It is possible to installed at the same time.)

Continuous turn	J613	Torque Limit	J611
Touch sensor	J536	Soft Float	J612
Weaving	J504	Coordinated Motion Package	J686
Line tracking	J512	Simultaneous Robot Link	J777
ARC sensor	J511	Arc related functions	
Root pass memorization	J532	AVC	J526

61.8 Supported robot models

In general, this feature supports all 6-axis robots that have no linear axes.
(For safety, collaborative robots and the M-2000 series will not be supported.)

R-2000iC/210F R-2000iC/165F R-2000iC/125L R-2000iC/210R R-2000iC/165R R-2000iC/270F R-2000iC/210L R-2000iC/220U R-2000iC/100P R-2000iC/210WE R-2000iC/270R ★ R-2000iC/240F ★ R-2000iD/165FH ★ R-2000iD/210FH ★ R-2000iD/100FH ★ R-1000iA/80F R-1000iA/100F R-1000iA/130F	M-900iB/330L ★ M-900iB/280 M-900iB/280L M-900iB/360 M-900iB/700 M-900iB/400L M710iC/50S M710iC/50 M710iC/70 M710iC/20L M710iC/50E M710iC/45M M710iC/12L M710iC/20M M-20iA M-20iA/10L M-20iA/20M M-20iA/35M M-20iA/12L M-20iB/25 M-20iB/25C M-20iD/25 ★ ARC Mate 120iD/12L ★ ARC Mate 120iD/35 ★	M-10iA/8L M-10iA/12S M-10iA/7L M-10iA/10MS M-10iA/12 M-10iA/10M M-10iAe M-10iAe/6L M-10iA/10S M-10iA M-10iA/6L M-10iD/8L ★ M-10iD/12 ★ ARC Mate 100iD/16S ★ LRMate200iD/7C LRMate200iD/7LC LRMate200iD/4SC LRMate200iD/7L LRMate200iD/4S LRMate200iD ER-4iA P-350iA/45 (★: supports 4ms for communication interval)
---	--	---

Supported robot models

61.9 Alarm Code List

MOTN-600 ST: Sequence No error. PC xx, Rob yy

[Cause] The sequence number (= xx) of the command packet did not match that (= yy) requested by the robot.

[Remedy] Make sure that the sequence number of the command packet to be sent is set correctly.

Make sure that the data structure, endianness, and message size to be sent are correct.

MOTN-602 ST: Data type error

[Cause] The value of the command packet data format (0: Cartesian, 1: Joint) is invalid.

[Remedy] Make sure that the value of the data format of the command packet to be sent is set correctly.

Make sure that the data structure, endianness, and message size to be sent are correct.

MOTN-603 ST: Receiving interval over

[Cause] The robot could not confirm the command packet with the next sequence number within the communication interval.

[Remedy] If the cause is thought to be communication delay, consider using "61.4 Buffering command packet".

If you suspect that packet loss is the cause, please consider replacing the Ethernet cable or installing an external shield.

Make sure that there is no large processing time before the external program sends the command packet.

MOTN-604 ST: Too many packet received

[Cause] The robot has received too many command packets.

[Remedy] Make sure that the command packet is not sent in excess of the bufferable size.

Or, check the alarm history to see if the robot has stopped due to the occurrence of another alarm.

MOTN-605 ST: Command is NaN or infinity. (A:xx, Sq.yy)

[Cause] An abnormal value (NaN or infinity) is set in the desired position data of the command packet.

[Remedy] Make sure that the desired position data is set correctly.

Make sure that the data structure, endianness, and message size to be sent are correct.

MOTN-606 ST: Need to AUTO and 100% ovrd

[Cause] When executing the IBGN start [*] instruction, it is necessary to be in AUTO mode and 100% OVERRIDE.

[Remedy] Execute the program in AUTO mode and 100% OVERRIDE.

Do not change the override during execution.

MOTN-607 ST: Protocol version mismatch PC xx, Rob yy

[Cause] A packet (= xx) with an incorrect protocol version was received.

[Remedy] Check the protocol version (= yy) used on the robot side, and use the correct version.

MOTN-608 ST: TCP/IP error. See Cause & Remedy.

[Cause] The status packet could not be communicated correctly from the robot.

[Remedy] Check the cable status, communication settings, and Ethernet is not busy.

MOTN-609 ST: Exceeded Jnt Velocity Limit. (A:xx, Sq.yy)

[Cause] As the speed of axis xx exceeded the allowable upper limit at the command packet with Sq. yy, the robot stopped.

[Remedy] Check the speed of the relevant part from the sent trajectory data, and then consider improving the trajectory generation algorithm. For inquiries related to this alarm, please be sure to confirm the above in advance.

MOTN-610 ST: Exceeded Jnt Acc/Dec Limit. (A:xx, Sq.yy)

[Cause] As the acceleration of axis xx exceeded the allowable upper limit at the command packet with Sq. yy, the robot stopped.

[Remedy] Check the acceleration at the relevant part from the sent trajectory data, and then consider improving the trajectory generation algorithm. For inquiries related to this alarm, please be sure to confirm the above in advance.

MOTN-611 ST: Exceeded Jnt Jerk Limit. (A:xx, Sq.yy)

[Cause] As the jerk of axis xx exceeds the allowable upper limit at the command packet with Sq. yy, the robot stopped.

[Remedy] Check the jerk of the relevant part from the sent trajectory data, and then consider improving the trajectory generation algorithm. For inquiries related to this alarm, please be sure to confirm the above in advance.

MOTN-612 ST:Close to Jnt Velocity Limit.(A:xx,Sq.yy)

[Cause] In the command packet of sequence number yy, the speed of axis xx approaches the allowable upper limit.

[Remedy] No special countermeasure is required.

If you do not need this warning, set the system variable \$STMO_GRP[].\$WARN_LIM to 100.

MOTN-613 ST:Close to Jnt Acc/Dec Limit.(A:xx,Sq.yy)

[Cause] In the command packet with sequence number yy, the acceleration of axis xx approaches the allowable upper limit.

[Remedy] No special countermeasure is required.

If you do not need this warning, set the system variable \$STMO_GRP[].\$WARN_LIM to 100.

MOTN-614 ST:Close to Jnt Jerk Limit.(A:xx,Sq.yy)

[Cause] In the command packet with sequence number yy, the jerk of axis xx approaches the allowable upper limit.

[Remedy] No special countermeasure is required.

If you do not need this warning, set the system variable \$STMO_GRP[].\$WARN_LIM to 100.

MOTN-615 ST:Please Disable Brake Control.

[Cause] The IBGN start [*] instruction could not be started because brake control is enabled.

[Remedy] Set \$PARAM_GROUP[].\$SV_OFF_ENB[1-9] = FALSE, and cycle power of the controller.

MOTN-616 ST:Reducer load excess.(A:xx)

[Cause] The robot stopped because the load level applied to the reducer unit on axis xx exceeds a threshold.

[Remedy] Consider measures such as making the motion more conservative so that the load on the xx-axis reducer is reduced.

MOTN-617 ST:Discontinuous command.(Sq.xx)

[Cause] A large discrepancy was found between the desired position and the current position at the command packet of Sq.xx.

[Remedy] Make sure that the desired position data that connects smoothly from the current position is set correctly.

Make sure that the data structure, endianness, and message size to be sent are correct.

MOTN-618 ST:Internal Error.

[Cause] This is an internal error.

[Remedy] Cycle power of the controller.

MOTN-619 ST:Cartesian type is not supported

[Cause] Cartesian format is used for with the robot model that cannot support Cartesian format for the desired position data.

[Remedy] Use Joint format for the target position data.

MOTN-623 ST:Please Disable Resume Offset.

[Cause] The IBGN start [*] instruction could not be started because Resume Offset function is enabled.

[Remedy] Disable the Resume Offset. It can be set by selecting [Screen Selection] → [Settings] → [Resume Offset].

For details, refer to “OPERATOR'S MANUAL (Basic Function) (B-83284EN)”.

MOTN-156 PM: Configuration changed

[Cause] The configuration of desired position data (Cartesian format) is different from that at the start of the IBGN start [*] instruction. Therefore the robot stopped with the alarm.

[Remedy] In this feature, if the desired position data is in Cartesian format, the robot motion cannot across the configuration.

For the definition of configurations, refer to “4 Program Configuration” in “OPERATOR'S MANUAL (Basic Function) (B-83284EN)”.

PRIO-023 There is no I/O for this type

[Cause] The I/O to be read or written via the command packet is not set for allocation.

[Remedy] Allocate the I/O to be read or written, or specify another I/O.

Appendix

A Sample program

This sample program sends only one command packet that specifies a posture of 0 degrees on all axes.

Please execute this program after fully confirming the safety around the robot.

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#include <iostream>
#include <ws2tcpip.h>
#include <winsock2.h>

typedef short      SHORT;
typedef long       LONG;
typedef unsigned char  UBYTE;
typedef unsigned short USHORT;
typedef unsigned long  ULONG;

#define ROBOT_IP_ADDRESS "127.0.0.1" /* Please specify the IP address of the actual robot. */
/* No change is required when using ROBOGUIDE on the same PC. */

typedef struct INIT_PKT_T {
    ULONG    pkt_type;
    ULONG    version;
} INIT_PKT_T;

typedef struct STAT_PKT_T {
    ULONG    pkt_type;
    ULONG    version;
    ULONG    sequence_no;
    UBYTE    status;
    UBYTE    io_r_type_o;
    USHORT   io_r_idx_o;
    USHORT   io_r_mask_o;
    USHORT   r_io_val;
    ULONG    time_stamp;
    float    cart[9];
    float    ang[9];
    float    q_current[9];
} STAT_PKT_T;

typedef struct CMD_PKT_T {
    ULONG    pkt_type;
    ULONG    version;
    ULONG    sequence_no;
    UBYTE    last;
    UBYTE    io_r_type_i;
    USHORT   io_r_idx_i;
    USHORT   io_r_mask_i;
    UBYTE    data_type;
    UBYTE    io_w_type;
    USHORT   io_w_idx;
    USHORT   io_w_mask;
    USHORT   io_w_val;
    USHORT   unused;
    float    cmd[9];
} CMD_PKT_T;

typedef struct END_PKT_T {
    ULONG    pkt_type;
    ULONG    version;
} END_PKT_T;
```

```

int main()
{
    INIT_PKT_T  init_pkt = { htonl(0),htonl(1) };
    END_PKT_T   end_pkt = { htonl(2),htonl(1) };

    STAT_PKT_T  stat_pkt;
    CMD_PKT_T   cmd_pkt = { htonl(1),htonl(1),htonl(1),1,0,htons(0),htons(0),1,0,
                          htons(0),htons(0),htons(0),htons(0), {0,0,0,0,0,0,0,0} };

    WSADATA     data;
    struct sockaddr_in rc_addr;
    struct sockaddr clitSockAddr;

    int err = WSStartup(MAKEWORD(2, 0), &data);
    rc_addr.sin_addr.s_addr = inet_addr(ROBOT_IP_ADDRESS);

    int rc_sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    rc_addr.sin_family = AF_INET;
    rc_addr.sin_port = htons(60015);

    DWORD timeout = 1000;
    err = setsockopt(rc_sock, SOL_SOCKET, SO_RCVTIMEO, (char *)&timeout, sizeof(DWORD));

    sendto(rc_sock, (const char *)&init_pkt, sizeof(init_pkt), 0, (struct sockaddr *)&rc_addr, sizeof(rc_addr));

    int sockaddrlen = sizeof(clitSockAddr);

    for (;;) {
        int receivedSize = recvfrom(rc_sock, (char *)&stat_pkt, sizeof(stat_pkt), 0,
                                   (sockaddr *)&rc_addr, &sockaddrlen);
        if (receivedSize != -1) {
            stat_pkt.sequence_no = ntohl(stat_pkt.sequence_no);
            stat_pkt.version = ntohl(stat_pkt.version);
            stat_pkt.time_stamp = ntohl(stat_pkt.time_stamp);
            for (int ii = 0; ii < 9; ii++) {
                *(long *)&stat_pkt.cart[ii] = ntohl(*(long *)&stat_pkt.cart[ii]);
                *(long *)&stat_pkt.ang[ii] = ntohl(*(long *)&stat_pkt.ang[ii]);
                *(long *)&stat_pkt.q_current[ii] = ntohl(*(long *)&stat_pkt.q_current[ii]);
            }

            if (stat_pkt.status & 0x1) {
                cmd_pkt.sequence_no = htonl(stat_pkt.sequence_no);
                for (int jj = 0; jj < 9; jj++) *(long *)&cmd_pkt.cmd[jj] = htonl(*(long *)&cmd_pkt.cmd[jj]);

                sendto(rc_sock, (const char *)&cmd_pkt, sizeof(cmd_pkt), 0,
                      (struct sockaddr *)&rc_addr, sizeof(rc_addr));

                break;
            }
        }
    }

    Sleep(1000);
    sendto(rc_sock, (const char *)&end_pkt, sizeof(end_pkt), 0, (struct sockaddr *)&rc_addr, sizeof(rc_addr));

    closesocket(rc_sock);
    WSACleanup();

    return 0;
}

```

B Calculation method of acceleration / jerk upper limit

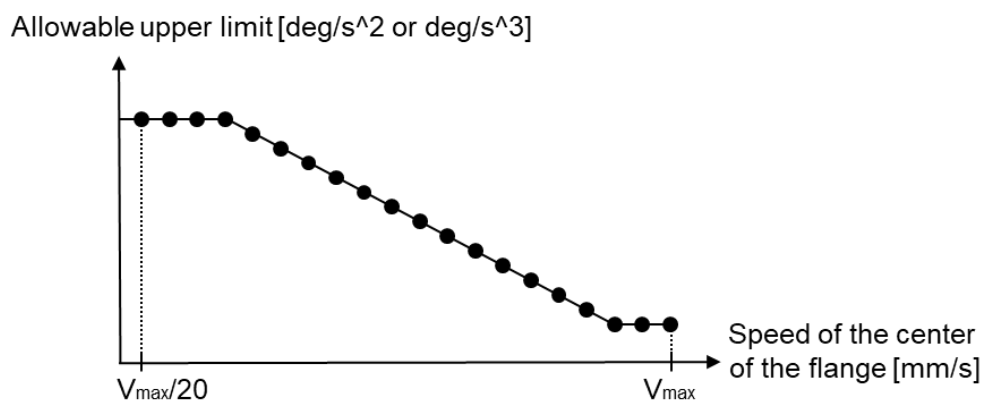
In this feature, the allowable upper limit of acceleration and jerk set for each axis is calculated as follows based on the Cartesian speed [mm/s] at the center of the flange and the payload [kg]. (Only when \$STMO_GRP[.].\$LMT_MODE is 0)
The calculation result is used only for internal judgment and is not reflected in the system variables.

Change of allowable upper limits according to the Cartesian speed at the center of flange

Each allowable upper limit has a table of 20 stages according to the Cartesian speed [mm/s] at the center of the flange.

We define the maximum speed of the center of flange as $V_{max}(=\$STMO_GRP[.].\$MAX_SPD)$, at this time, each speed obtained by dividing V_{max} into 20 stages corresponds to each index of that table.

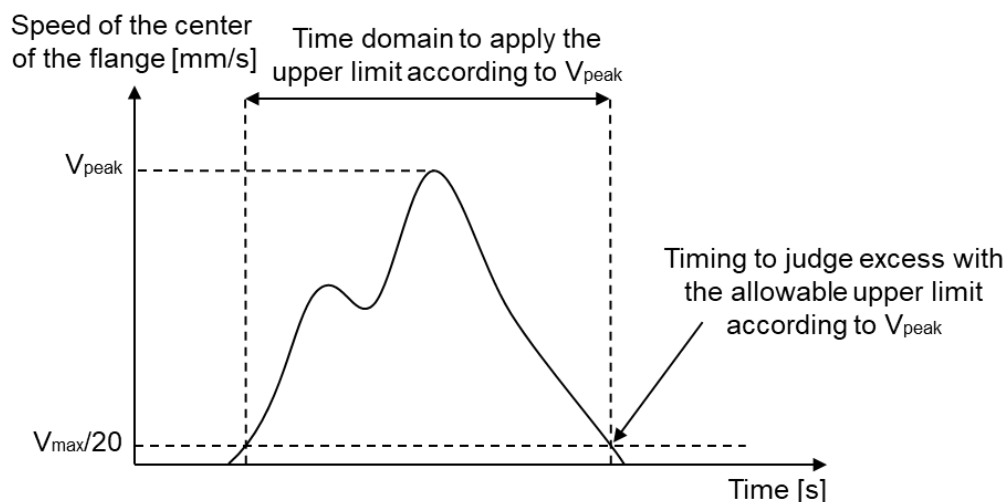
In general, the lower the speed at the center of the flange, the higher the allowable upper limit, and the higher the speed, the lower the allowable upper limit. This is illustrated as follows.



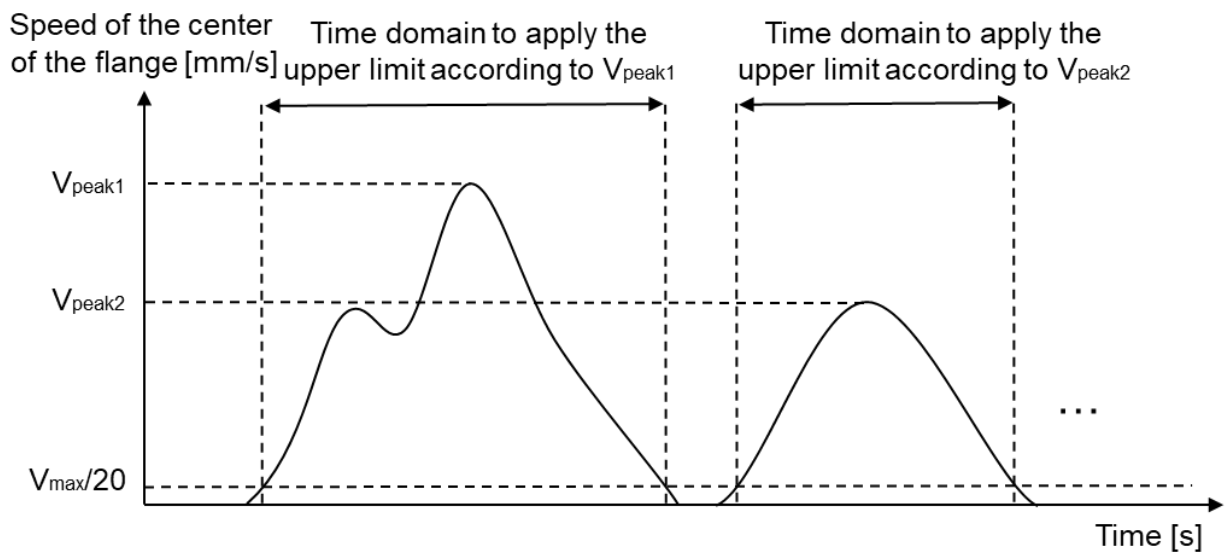
- The limit value at arbitrary speed is calculated by linearly interpolating the values at two adjacent indexes.
- For speeds less than $V_{max}/20$, the limit value at $V_{max}/20$ is applied as it is.
- The limit value at the speed higher than V_{max} is calculated by linearly extrapolating the limit values at V_{max} and the adjacent.

In addition, the allowable upper limit is not updated every communication interval. In the entire time domain until the flange center speed exceeds $V_{max}/20$ and falls below it again, the allowable upper limit value according to the maximum observed speed V_{peak} is applied.

Since V_{peak} is identified at the end of the above time domain, the calculation of the allowable upper limit value and the excess judgment are also performed at this time.



After the flange center speed falls below $V_{max}/20$, the V_{peak} value is reset to 0 and the allowable upper limit value at $V_{max}/20$ is applied. When the flange center velocity exceeds $V_{max}/20$, V_{peak} observation is restarted and the same process is repeated.

**Note**

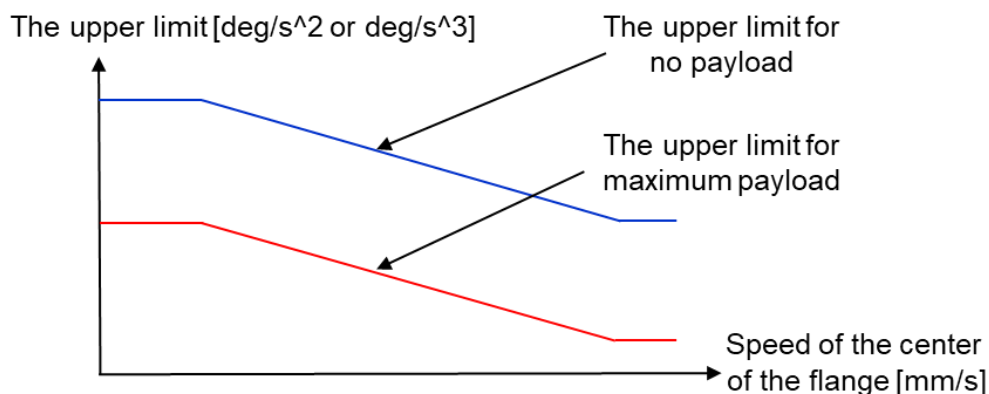
- 1 Due to this specification, there may be a time lag between the excess and the alarm occurrence.
- 2 Whenever the limit value at $V_{max}/20$ is exceeded, the robot will stop immediately with an alarm.
- 3 If more than one allowable upper limit is exceeded, the alarm will be raised only for the largest excess.

As an exceptional situation, if the speed at the center of the flange exceeds $V_{max}/20$ and does not fall below it again after a certain period of time, a tentative check will be made based on the limit value according to V_{peak} at that time. If no excess is detected, the robot motion continues as it is, and after that, the tentative judgment is made again every time a certain period of time elapses. The time for this tentative judgment (intermediate check time) depends on the robot model, but is usually set to about 5 [sec].

Change of allowable upper limit value according to the payload

There are two types of twenty-staged table that is according to the speed at the center of flange, one for a payload of 0 [kg] and the other for a maximum payload [kg], which are used by linear interpolation according to the actual payload.

In general, the smaller the payload, the larger the allowable upper limit value, and the larger the payload, the smaller the allowable upper limit value. The figure is as follows.

**Note**

- 1 It is necessary to set the payload information according to the actual weight of the work piece. Refer to "3.17 PAYLOAD SETTING" in "OPERATOR'S MANUAL (Basic Function) (B-83284EN)".
- 2 If the payload information has never been set, the setting is based on the maximum payload [kg].
- 3 If the payload is significantly incorrect, vibration may increase or a collision may be falsely detected.

How to get the allowable upper limit table

Each table of allowable upper limits can be obtained by sending and receiving the following packets in the same way as status or command packets.

Each response packet in Table 6 provides a table for one axis. (Any of speed, acceleration, jerk) The allowable upper limit for speed can be obtained too, but the table will always be constant regardless of Cartesian speed or payload.

Table 5: Request packet for allowable upper limit table (external device → robot)

Data type	Name	Description
4 byte Unsigned integer	Packet type	Set 3 : Request packet for allowable upper limit table.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. Set 1.
4 byte Unsigned integer	Axis number	Specify one of the integers 1-9.
4 byte Unsigned integer	Type of limit	Specify one of 0:velocity [deg/s], 1:acceleration [deg/s ²], 2:jerk [deg/s ³].

Table 6: Response packet for allowable upper limit table (robot → external device)

Data type	Name	Description
4 byte Unsigned integer	Packet type	3 is set.
4 byte Unsigned integer	Version No.	Version No. of packet format and function. 1 is set.
4 byte Unsigned integer	Robot axis number	The axis number specified in the request packet.
4 byte Unsigned integer	Type of limit	The type of allowable upper limit specified in the request packet.
4 byte Unsigned integer	Vmax	The value of \$STMO_GRP[].\$MAX_SPD. The unit is [mm/s].
4 byte Unsigned integer	Intermediate check time	Time interval for making a tentative judgment about exceeding the allowable upper limit. The unit is [sec].
4 byte float	Limit value at no payload(1)	This is the allowable upper limit when moving at a speed of 1/20 or less of Vmax[mm/s].
4 byte float	Limit value at no payload(2)	This is the allowable upper limit when moving at a speed of 2/20 or less of Vmax[mm/s].
:	:	::
4 byte float	Limit value at no payload(19)	This is the allowable upper limit when moving at a speed of 19/20 or less of Vmax[mm/s].
4 byte float	Limit value at no payload(20)	This is the allowable upper limit when moving at a speed or less of Vmax[mm/s].
4 byte float	Limit value at max payload(1)	This is the allowable upper limit when moving at a speed of 1/20 or less of Vmax[mm/s].
4 byte float	Limit value at max payload(2)	This is the allowable upper limit when moving at a speed of 2/20 or less of Vmax[mm/s].
:	:	:
4 byte float	Limit value at max payload(19)	This is the allowable upper limit when moving at a speed of 19/20 or less of Vmax[mm/s].
4 byte float	Limit value at max payload(20)	This is the allowable upper limit when moving at a speed or less of Vmax[mm/s].

Periodic update mode of allowable upper limit (\$LMT_MODE=1)

For some robot models, the mode with \$LMT_MODE=1 that updates the allowable upper limit of acceleration and jerk for each communication interval based on another algorithm is enabled by default. It may allow greater acceleration and jerk than the conventional. To disable this mode, set the system variable \$STMO_GRP[].\$LMT_MODE to 0 and cycle power.


C Frequently Asked Questions

Category	Contents of inquiry	Answer example
Communication Interval	Is the communication interval changeable? (e.g. from 8ms to 4ms or 1ms.)	No. The communication interval matches the control cycle of the robot. The control cycle of the robot is part of the design and cannot be changed.
	There seems to be some variation in the transmission interval of state packets. Even if communication is performed at 8 ms intervals, the "Receiving interval over" alarm sometimes occur.	There may be some fluctuation depending on other tasks inside the robot controller and the processing time of Ethernet communication. Therefore, even if the communication interval is set to 8ms, the actual cycle may be about 6ms to 10ms. If this specification is unacceptable, please consider buffering command packets.
	Communication interval (= \$COM_INT) is set to a time longer than 8ms	Depending on the number of motion groups and option configuration, the control cycle (= communication interval) can be longer due to the increase in processing time.
Stop pattern when an alarm occurs	Is it possible to stop the robot smoothly without turning off the servo immediately when an alarm occurs?	No. Currently, the stop pattern when an alarm occurs while using the stream motion is power-off stop only (corresponding to category 0 stop of IEC 60204-1). (We plan to improve it in the future.)
Sequence number	Is it possible to send multiple command packets with the same sequence number in case of packet loss?	No. After processing a command packet that arrived first, an alarm of "MOTN-600 ST: Sequence No error." will occur when the next command packet is processed.
Specifications of the mechanical unit	How to obtain the operator's manual of mechanical unit that describes the operating space diagram?	Please contact our sales.
	Is it possible to disclose mathematical formulas for forward kinematics, inverse kinematics, and self-collisions?	No. Forward kinematics, inverse kinematics, and self-collision information are confidential and cannot be disclosed.
	Is it possible to disclose the rated or instantaneous maximum torque of each axis and the physical information of each link?	No. Regarding the specifications of the mechanical unit, we cannot disclose anything other than the information described in the operator's manual of mechanical unit. You need to estimate all the necessary information by yourself.
	Is CAD data provided?	Yes. Please contact our sales.
Acceleration and jerk	An alarm for excessive acceleration / jerk is posted even though the motion is not so fast.	Please calculate the speed, acceleration, and jerk of each axis from the time-series data of the desired position sent to the robot and confirm the validity. <u>Common cause:</u> "The speed was discontinuous at the start of the movement because the path was generated by the sin or cos function."
	Does the mount angle setting affect the allowable upper limit?	No. At this time, the mount angle setting is not considered.
	How is the acceleration / jerk calculated for the first 1 to 3 command packets?	The calculation is performed assuming that the desired position before receiving the first command packet was always the same as the desired position specified by the first command packet.
	Although the motion of J2 axis rotates J3 motor in tandem, the excess of allowable upper limits for J3 axis does not occur as long as the motion of J3 commanded is zero. Is this understanding correct?	Yes, as long as the motion of J3 commanded is zero, the allowable upper limits of acceleration and jerk will not be exceeded.

	Is there a difference in the allowable upper limit values between Cartesian and Joint format of the desired position data?	No. The value of the allowable upper limit does not change depending on the format of the desired position data. In case of Cartesian format, after converting to the Joint format internally, the same process is performed to check for excesses.
	Is it possible to deactivate the acceleration and jerk restriction under the responsibility of the user?	No. From the viewpoint of mechanical protection and safety, we do not provide a means to deactivate the acceleration and jerk restriction even under the user's responsibility.
Tracking performance	How long is the time delay for the robot to actually reach the desired position specified in the command packet?	In the case of an robot with 8ms interval, there is a delay of about 50 ms including communication delay and servo tracking delay. (It becomes about half of that in case of a robot with 4ms interval) There is no way to make this shorter.
	Is there a ROBOGUIDE feature to simulate the deviation of the trajectory between the command position and the servo position?	No. There is no ROBOGUIDE feature to simulate the deviation of the trajectory between the command position and the servo position.
Supported robot models	Is it possible to support collaborative robots under certain conditions, such as being available only while the safety feature is off?	No. Even if under certain conditions, this feature cannot support collaborative robots.
Payload setting	Is it possible to dynamically switch the payload setting while executing the IBGN start [*] instruction?	The payload setting cannot be switched while the IBGN start [*] instruction is being executed. By devising a program, for example, it is possible to execute the instruction to switch the payload setting after the IBGN end[*] once, and to start the execution of the IBGN start [*] instruction again by looping or jumping.

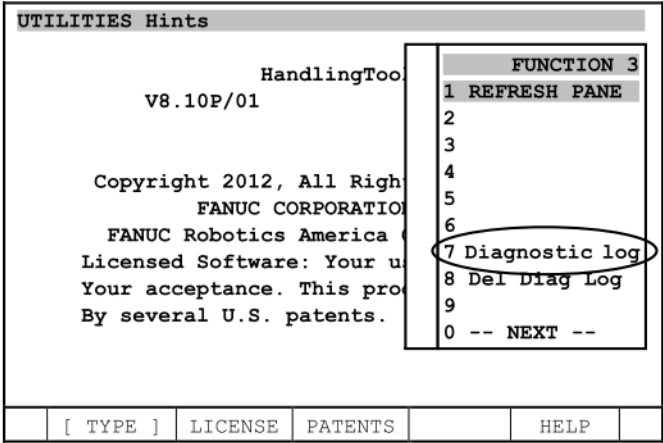
D Diagnostic Log

If you encountered a problem and make an inquiry, please obtain the diagnostic log by the following procedure.

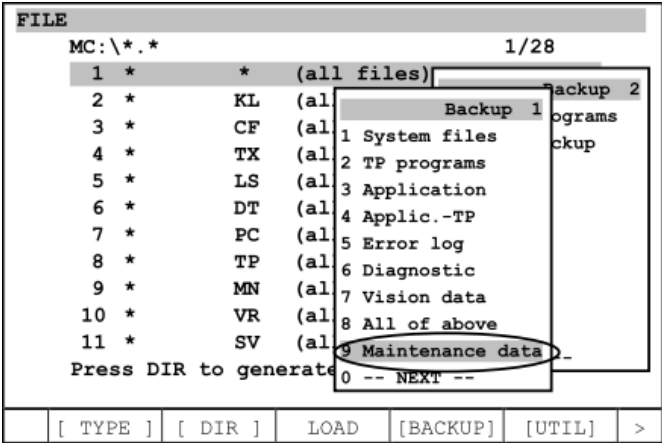
**WARNING**
If you execute the diagnostic log function, confirm that **the status of robot is STOP**.

Note
1 Do the following operation just after the problem occurs and BEFORE POWER OFF.
If you turn off the power, the useful log for investigation will be cleared.
2 Do it only one time just after the problem occurs. The saved log data will be deleted from the robot controller if the same operation is done twice.

- 1 Select [FCTN] key -> 0--NEXT-- -> 0--NEXT-- -> 7 Diagnostic log.
- 2 “Diagnostic log” is displayed on 3rd page in FCTN menu.



- 3 Set the cursor onto “Diagnostic log” and press the [ENTER] key. SYST – 276 Now getting log... is posted.
- 4 It takes about 1 minute to save.
- 5 SYST – 274 Diagnostic log done is posted after done saving.
If this warning does not appear, you can confirm by the disappearance of FUNCTION menu.
- 6 Insert the memory card or USB memory to the controller.
- 7 Press [MENU] key – 7 FILE.
- 8 Select – F4, [BACKUP] – Maintenance data.



- 9 Select F4, YES after displaying the confirm message.

FILE			
MC:*.*		1/28	
1	*	*	(all files)
2	*	KL	(all KAREL source)
3	*	CF	(all command files)
4	*	TX	(all text files)
5	*	LS	(all KAREL listings)
6	*	DT	(all KAREL data files)
7	*	PC	(all KAREL p-code)
8	*	TP	(all TP programs)
9	*	MN	(all MN programs)
10	*	VR	(all variable files)
11	*	SV	(all system files)
Save maintenance data. OK?			
		YES	NO

- 10 In above sample case, maintenance data is saved into MC:¥MNT_DATA¥.
 It is saved into MNT_DATA¥ under the current directory. MNT_DATA is newly created.
- 11 Please compress MNT_DATA and send it to us.