

FANUC Robot series

**R-30*i*B Plus/R-30*i*B Mate Plus/R-30*i*B Compact Plus
CONTROLLER**

Remote Motion Interface OPERATOR'S MANUAL

B-84184EN/02

- **Original Instructions**

Thank you very much for purchasing FANUC Robot.

Before using the Robot, be sure to read the "FANUC Robot series SAFETY HANDBOOK (B-80687EN)" and understand the content.

- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

The products in this manual are controlled based on Japan's "Foreign Exchange and Foreign Trade Law". The export from Japan may be subject to an export license by the government of Japan.

Further, re-export to another country may be subject to the license of the government of the country from where the product is re-exported. Furthermore, the product may also be controlled by re-export regulations of the United States government.

Should you wish to export or re-export these products, please contact FANUC for advice.

In this manual, we endeavor to include all pertinent matters. There are, however, a very large number of operations that must not or cannot be performed, and if the manual contained them all, it would be enormous in volume. It is, therefore, requested to assume that any operations that are not explicitly described as being possible are "not possible".

SAFETY PRECAUTIONS

This chapter describes the precautions which must be followed to enable the safe use of the robot. Before using the robot, be sure to read this chapter thoroughly.

For detailed functions of the robot operation, read the relevant operator's manual to understand fully its specification.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral equipment installed in a work cell.

For safe use of FANUC robots, you must read and follow the instructions in the “FANUC Robot series SAFETY HANDBOOK (B-80687EN)”.

1 PERSONNEL

Personnel can be classified as follows.

Operator:

- Turns the robot controller power ON/OFF
- Starts the robot program from operator panel

Programmer or Teaching operator:

- Operates the robot
- Teaches the robot inside the safeguarded space

Maintenance technician:

- Operates the robot
 - Teaches the robot inside the safeguarded space
 - Performs maintenance (repair, adjustment, replacement)
-
- The operator is not allowed to work in the safeguarded space.
 - The programmer or teaching operator and maintenance technician are allowed to work in the safeguarded space. Work carried out in the safeguarded space include transportation, installation, teaching, adjustment, and maintenance.
 - To work inside the safeguarded space, the person must be trained on proper robot operation.

Table 1 (a) lists the work outside the safeguarded space. In this table, the symbol “○” means the work allowed to be carried out by the specified personnel.

Table 1 (a) List of work outside the Safeguarded Space

	Operator	Programmer or Teaching operator	Maintenance technician
Turn power ON/OFF to Robot controller	○	○	○
Select operating mode (AUTO/T1/T2)		○	○
Select remote/local mode		○	○
Select robot program with teach pendant		○	○
Select robot program with external device		○	○
Start robot program with operator's panel	○	○	○
Start robot program with teach pendant		○	○
Reset alarm with operator's panel		○	○
Reset alarm with teach pendant		○	○
Set data on teach pendant		○	○
Teaching with teach pendant		○	○
Emergency stop with operator's panel	○	○	○
Emergency stop with teach pendant	○	○	○
Operator's panel maintenance			○
Teach pendant maintenance			○

During robot operation, programming and maintenance, the operator, programmer, teaching operator and maintenance technician take care of their safety using at least the following safety protectors:

- Use clothes, uniform, overall adequate for the work
- Safety shoes
- Helmet

2 DEFINITION OF SAFETY NOTATIONS

To ensure the safety of users and prevent damage to the machine, this manual indicates each precaution on safety with "WARNING" or "CAUTION" according to its severity. Supplementary information is indicated by "NOTE". Read the contents of each "WARNING", "CAUTION" and "NOTE" before using the robot.



Symbol	Definitions
 WARNING	Used if hazard resulting in the death or serious injury of the user will be expected to occur if he or she fails to follow the approved procedure.
 CAUTION	Used if a hazard resulting in the minor or moderate injury of the user, or equipment damage may be expected to occur if he or she fails to follow the approved procedure.
NOTE	Used if a supplementary explanation not related to any of WARNING and CAUTION is to be indicated.

TABLE OF CONTENTS

SAFETY PRECAUTIONS.....	s-1
1 REMOTE MOTION INTERFACE OVERVIEW	1
1.1 OVERVIEW / INTRODUCTION	1
1.2 HARDWARE AND SOFTWARE REQUIREMENTS	1
1.3 COMPATIBILITY	1
1.4 LIMITATIONS	2
1.4.1 Single Group Motion.....	2
1.4.2 Hot Start Support.....	2
1.4.3 Number Of Instruction Packets	2
1.4.4 Short Motion	2
1.4.5 Non-Motion Interface Support	2
2 REMOTE MOTION APPLICATION PROGRAM INTERFACE.....	3
2.1 REMOTE MOTION APPLICATION PROGRAM INTERFACE OVERVIEW ..	3
2.2 COMMUNICATION PACKETS	3
2.2.1 Packet Exchange.....	3
2.2.2 Controller Disconnect Packet.....	4
2.2.3 Timeout Terminate Packet	5
2.2.4 System Fault Packet	5
2.3 ROBOT COMMAND PACKETS	5
2.3.1 Packet To Initialize Remote Motion.....	6
2.3.2 Packet To Abort Remote Motion Program.....	7
2.3.3 Packet To Pause Remote Motion Program.....	8
2.3.4 Packet To Continue Remote Motion Program	8
2.3.5 Packet To Read Controller Error	8
2.3.6 Packet To Set The Current UFrame-UTool Number.....	9
2.3.7 Packet To Get Controller Status	9
2.3.8 Packet To Read User Frame Data.....	10
2.3.9 Packet To Set User Frame Data.....	11
2.3.10 Packet To Read User Tool Data	11
2.3.11 Packet To Set User Tool Data	12
2.3.12 Packet To Read Input Port.....	12
2.3.13 Packet To Write Digital Output Port.....	12
2.3.14 Packet To Read Current Robot Cartesian Position.....	13

2.3.15	Packet To Read Current Robot Joint Angles.....	14
2.3.16	Packet To Set Speed Override.....	14
2.3.17	Packet To Get Current User/Tool Frame Number.....	14
2.3.18	Packet To Read Position Register Data.....	15
2.3.19	Packet To Write Position Register Data.....	16
2.3.20	Packet to Reset Robot Controller	16
2.3.21	Packet To Read Current Tool Center Point Speed	17
2.4	TEACH PENDANT PROGRAM INSTRUCTION PACKETS	17
2.4.1	Packet To Wait For DIN Instruction	19
2.4.2	Packet To Set User Frame Instruction.....	19
2.4.3	Packet To Set User Tool Instruction	19
2.4.4	Packet To Add Wait Time Instruction.....	20
2.4.5	Packet To Set Payload Instruction.....	20
2.4.6	Packet To Call A Program.....	20
2.4.7	Packet To Add Linear Motion Instruction.....	21
2.4.7.1	Packet to add two groups linear motion instruction.....	25
2.4.8	Packet To Add Linear Incremental Motion Instruction.....	28
2.4.8.1	Packet to add two groups linear incremental motion instruction.....	29
2.4.9	Packet To Add Joint Motion Instruction	30
2.4.9.1	Packet to add two group joint motion instruction.....	31
2.4.10	Packet To Add Joint Incremental Motion Instruction	33
2.4.10.1	Packet to add two groups joint incremental motion instruction.....	34
2.4.11	Packet To Add Circular Motion Instruction.....	35
2.4.11.1	Packet to add two groups circular motion instruction	36
2.4.12	Packet To Add Circular Incremental MotionInstruction.....	39
2.4.12.1	Packet to add two groups circular incremental motion instruction.....	40
2.4.13	Packet To Add Joint Motion With Joint Representation.....	41
2.4.13.1	Packet to add two groups joint motion with joint representation instruction	42
2.4.14	Packet To Add Joint Incremental Motion With Joint Representation.....	43
2.4.14.1	Packet to add two groups joint incremental motion with joint representation instruction	44
2.4.15	Packet To Add Linear Motion With JointRepresentation.....	44
2.4.15.1	Packet to add two groups linear motion with joint representation instruction...	45
2.4.16	Packet To Add Linear Motion With JointRepresentation.....	46
2.4.16.1	Packet to add two groups circular incremental motion instruction.....	46
2.4.17	Unknown Packet Handling.....	47
2.5	POSITION RECORDING.....	47
2.5.1	Using Position Registers	47
2.5.2	Using RMI Position Record Menu.....	47

3	REMOTE DEVICE CONTROL FLOW	51
3.1	OVERVIEW OF REMOTE DEVICE CONTROLFLOW	51
3.2	INSTRUCTION BUFFER HANDLING	52
3.3	HOLD STATE ERROR HANDLING	52
3.4	JUMP AND SKIP INSTRUCTIONS	52
3.5	ERROR HANDLING	52
3.6	VISION PROCESS	53
4	DATA LOGGING	54
5	ASCII STRING INSTRUCTION PACKETS	55
5.1	ASCII STRING PACKET HANDSHAKING	55
5.2	SYSTEM VARIABLE SETTING	56
5.3	ASCII STRING INSTRUCTION PACKET	57
5.3.1	ASCII String Packet For Single Group Controller	57
5.3.2	ASCII String Packet For Two Group MotionInstruction	60
5.3.3	Communication Packet	60
5.3.4	Modified FRC_Initialize Command	61
5.4	LIST OF INVALID ASCII STRING INSTRUCTIONS	61
 APPENDIX		
A	RMI ERRORID REFERENCE TABLE	65

1 REMOTE MOTION INTERFACE OVERVIEW

1.1 OVERVIEW / INTRODUCTION

Remote Motion Interface (RMI) allows a remote device to control the robot motion by sending teach pendant program instructions to the controller. The controller creates a special teach pendant (TP) program, executes this program and appends these new instructions from the remote device to the TP program on the fly.

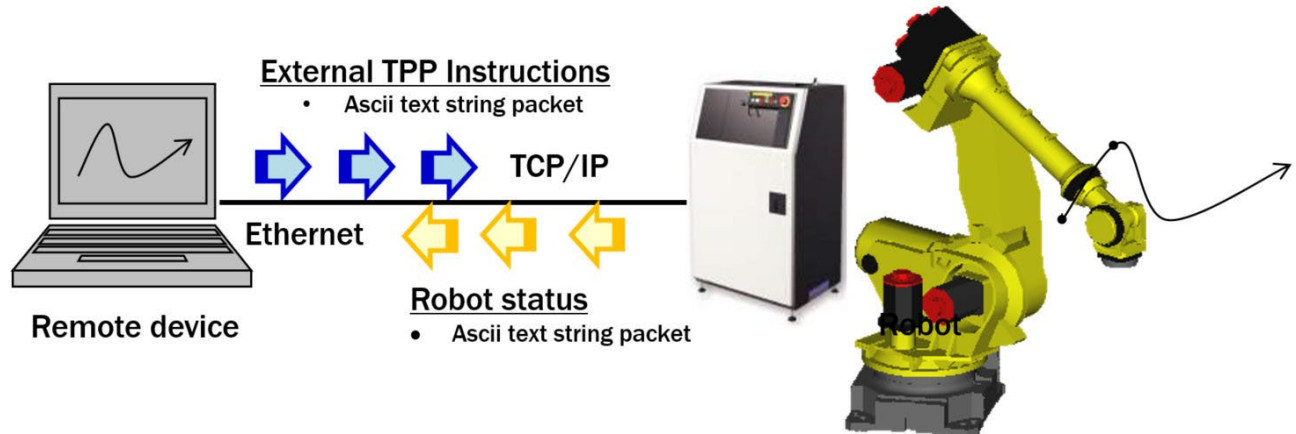


Fig. 1.1 (a) System Overview

This option is very similar to PLC Motion Interface. PLC Motion Interface interacts with a PLC through EDA, whereas the Remote Motion Interface interacts with a remote device through TCP/IP socket messages.

This option's main target is the material handling market with simple pick and place operations. Due to a limited number of instructions available for the Remote Motion Interface, it is not meant for complicated applications such as arc welding that reads real time sensor feedback and dynamically adjusts the robot path.

1.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware

The Remote Motion Interface option requires an R-30iB Plus controller.

Software

The controller needs to be loaded with the Remote Motion Interface (R912) option.

Communication

Ethernet connection.

1.3 COMPATIBILITY

The Remote Motion Interface (R912) option is not compatible with PLC Motion Interface (R892). You cannot load both options on the controller at the same time.

1.4 LIMITATIONS

1.4.1 Single Group Motion

Prior to the V9.30P/12 (7DF3/12) and V9.40P/06 (7DF5/06) software releases, RMI only supported single group systems. RMI now supports two group motion instructions in a multiple group system.

1.4.2 Hot Start Support

For a controller running the Remote Motion Interface software, the controller cannot perform a Hot Start. If Hot Start is enabled, the Remote Motion Interface software disables the Hot Start when it receives the FRC Initialize command and it posts an **RMIT-016** error to alert the user about the change in the system behavior.

1.4.3 Number Of Instruction Packets

The Remote Motion Interface supports a maximum number of eight TP instructions that the remote device can send to the robot controller at one time. When the buffer is full and the robot controller receives a new instruction packet, it returns the packet with an error code, **RMIT-028**. When the controller completes the first instruction, the remote device is allowed to send its next instruction.

1.4.4 Short Motion

The Remote Motion Interface does not support motion shorter than three ITP (12 milliseconds or 24 milliseconds based on robot model). If a motion instruction is too short, the robot controller slows down the motion so that the motion is at least three ITP in duration.

1.4.5 Non-Motion Interface Support

The Remote Motion Interface is intended for control of robot motion. Some basic non-motion commands are available, but systems will typically also require a separate communications option for the exchange of data and I/O. The HMI Device option (R553) supports the use of Modbus TCP and OPC-UA, both of which are commonly used with PC-based control systems. These protocols allow for the transfer of I/O and data values between the robot and remote device.

2 REMOTE MOTION APPLICATION PROGRAM INTERFACE

2.1 REMOTE MOTION APPLICATION PROGRAM INTERFACE OVERVIEW

There are three types of TCP/IP packets sent from the remote device to the robot controller:

1. *Communication packets*: These packets establish/terminate the TCP/IP communication session with the robot controller.
2. *Command packets*: These packets perform administrative work without adding a new TPP instruction to the TP program.
3. *Instruction packets*: These packets append TPP instructions to a running TP program.

The Remote Motion Interface uses TCP/IP socket to communicate with the remote device where the remote device is a client and the robot controller is the server. The Remote Motion Interface uses a specific logic port and it can only talk to one remote device at one time, hence it does not allow multiple remote sources to control the same robot. The TCP/IP protocol is chosen to ensure that an instruction or command packet is always received by the intended target in the same order as it is sent. The data (payload) exchanged between the controller and the remote device is ASCII text and it is similar to a JSON string.

NOTE

Each string is terminated by `\r\n` so the Remote Motion Interface parser knows that a packet is a complete packet.

NOTE

The first key in the text string has to be either “Command”, “Communication” or “Instruction”. RMI will return **RMIT-022 Invalid Text String** if this rule is not followed.

2.2 COMMUNICATION PACKETS

There are two connection packets that the remote device can send to the robot controller:

- **FRC_Connect**: Get controller permission to connect to the server. The remote device sends this packet to the controller at the beginning of a Remote Motion Interface session.
- **FRC_Disconnect**: Disconnect from the robot controller. This is the last packet the remote device sends to the robot controller to terminate its Remote Motion Interface(RMI) session.

2.2.1 Packet Exchange

Remote Device Command Packet	Robot Controller Return Packet
{“Communication”: “FRC_Connect”} \r\n	{“Communication”: “FRC_Connect”, “ErrorID”: integerValue, “PortNumber”: shortValue1, “MajorVersion”: shortValue2, “MinorVersion”: shortValue3} \r\n

The remote device sends this packet to a specific robot controller port (port number 16001) to start a Remote Motion Interface session. The robot controller sends the return packet back with the `integerValue = 0` to indicate that the connection is established. Otherwise, the `integerValue` is the robot controller's error code.

If the robot controller allows an RMI session to start (i.e., `ErrorID = 0`), the `PortNumber` returned by the robot controller is the port number that the remote device will connect to for the rest of the RMI session. The remote device should disconnect from port 16001 after it receives the return packet. Note that this is the only packet sent by the remote device that uses robot controller port 16001. Any other packet should use the returned `shortValue1` for its TCP/IP port connection.

The robot controller also returns the major and minor version number back to the remote device. The controller changes the major version number when the controller updates the protocols defined in this section, and the controller changes its minor version number when an update is applied with no change to the protocol. In general, the protocol is backward compatible. For example, if the remote device's interface program is implemented based on major version 1 and the major version on robot controller is 2, the remote device's code should still work. However, if the remote device's interface program is built on major version 2 and the controller returns that its major version is 1, the remote device interface may have issues since the controller will not provide some fields that are expected by the remote device. Therefore, it is recommended to update the robot controller to the latest software version before using the Remote Motion Interface.

Currently, the `MajorVersion` can have four different values:

MajorVersion = 1

RMI has the original specification. It does not support controller with multiple groups and it does not support ASCII string packets defined in Chapter 5, ASCII STRING INSTRUCTION PACKETS.

MajorVersion = 2

RMI has the original specification. It also supports controller with multiple groups and it does not support ASCII string packets defined in Chapter 5, ASCII STRING INSTRUCTION PACKETS.

MajorVersion = 3

This version adds support for ASCII string packets. This new version is in V9.30P/15 (7DF3P15), V9.40P/12(7DF5P12) and later software.

Major Version = 4

This version adds support for `FRC_Call` and `Tool_Offset`, `PR[]` motion modifier. The `FRC_Reset` command also works without running the `RMI_MOVE TP` program. These changes are in V9.30P/19 (7DF3P19) and V9.40P/18 (7DF5P18) and later software.

Major Version = 5

New motion modifier `ALIM`; `AlimReg` and `NoBlend` are added to RMI. These additional functions are available in V9.30P/24 (7DF3P24) and V9.40P/28(7DF5P28) and later software.

2.2.2 Controller Disconnect Packet

Remote Device Command Packet	Robot Controller Return Packet
{ "Communication": "FRC_Disconnect" } \r\n	{ "Communication": "FRC_Disonnect", "ErrorID": integerValue } \r\n

The remote device sends the `FRC_Disconnect` packet to terminate the communication session with the robot controller. The robot controller sends the return packet back with the

integerValue = 0 to indicate the communication session is completed. Otherwise, the integerValue is robot controller's error code.

2.2.3 Timeout Terminate Packet

Robot Controller Termination Packet	Remote Device Return Packet
{"Communication": "FRC_Terminate"} \r\n	None

When remote device connected to the controller does not send any command/instruction packet to the controller for a long period of time (default is 60 minutes), the controller will automatically terminate the connection between the controller and the remote device. Once the connection is terminated, the remote device needs to send an FRC_Connect packet to the controller to reestablish the connection.

2.2.4 System Fault Packet

Robot Controller System Fault Packet	Remote Device Return Packet
{"Communication": "FRC_SystemFault", "SequenceID": integerValue} \r\n	None

The robot controller sends this packet to the remote device when the controller encounters an error and the running TP program is paused. For example, when a motion instruction causes a limit error, the controller will send out this packet to inform the remote device about this error. The integerValue is the sequence ID of the instruction that causes the issue.

2.3 ROBOT COMMAND PACKETS

None of the packets listed in this section add an instruction to the teach pendant program. Instead, these packets ask the robot to perform functions outside of the TP program, and the effect is immediate. For example, if you send seven motion instructions to the controller and the robot is executing the third instruction, then you send a speed override command to change the program override from 100% to 50%, the override change takes effect during the execution of the third instruction and it does not wait until all seven previously sent instructions complete before changing the override.

When sending a command packet to the controller, make sure the previously sent command packet is returned. Some of the commands, such as FRC_Reset, take time to complete their operation. If you send a command packet before the controller has time to complete previous command packet, the controller will return the packet with an error code.

There are two classes of commands:

1. One type only requires that the remote device has made a connection to the robot controller. That is, once the FRC_Connect packet is accepted by the controller, the controller will respond to the following commands:
 - FRC_ReadPositionRegister
 - FRC_WritePositionRegister
 - FRC_SetOverride
 - FRC_GetStatus
 - FRC_GetUFrameUTool
 - FRC_ReadUToolData

- FRC_WriteUToolData
- FRC_ReadUFrameData
- FRC_WriteUFrameData
- FRC_ReadJointAngles
- FRC_ReadCartesianPosition
- FRC_Reset. This command is available after the V9.30P/19 (7DF3/19) and V9.40P/18 (7DF5/18) release

2. Other commands will work depending on the RMI current status. For example, the FRC_Continue command only works after the FRC_Initialize has been sent to the controller.

2.3.1 Packet To Initialize Remote Motion

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_Initialize”, “GroupMask”: unsignedByteValue } \r\n</pre>	<pre>{“Command”: “FRC_Initialize”, “ErrorID”: integerValue, “GroupMask”: unsignedByteValue } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

The remote device sends this packet to the controller when it wants to start adding instructions to the TP program. Before sending this command to the controller, make sure that the robot controller is in the following state:

1. The teach pendant is disabled and the controller is in AUTO mode.
2. The controller is ready to run, i.e., there are no servo or other errors.
3. The selected TP program is not RMI_MOVE.

NOTE

Sometimes the RMI_MOVE program can be selected without the user being aware of it. If ErrorID = 7015 is received after sending FRC_Initialize, make sure the RMI_MOVE program is not selected by doing the following: Press the **SELECT** button on the TP, choose another program besides RMI_MOVE from the program list, then press the **ENTER** button on the TP. Next, resend the FRC_Initialize command.

If you are not sure about the current state of the controller, you can send the FRC_GetStatus packet to get the current robot controller's status.

If the initialization command is executed successfully, the return integerValue is 0. At this point, the controller has created the RMI_MOVE TP program and the program is running. Otherwise, the integerValue of the return packet is the error code from the robot controller.

If the FRC_Initialize command is executed successfully, you will have to send an FRC_Abort command to terminate the RMI program in order for another TP program to run. Otherwise, even if you manually abort the RMI_MOVE TP program through the teach pendant, the RMI still has program control and you will not be able to execute other TP programs.

It takes the robot controller some time to initialize the system to accept the TP program instructions. Please wait until you have received the return packet before sending the next packet to the controller.

The **GroupMask** key from a remote device is an option for a single group system. If the **GroupMask** key is not presented, RMI defaults the **GroupMask** to 1. The **GroupMask** is an unsigned one-byte bit- field where each bit corresponds to the group number. For example,

GroupMask = 0b00100010 (binary form)

would mean Groups 2 and 6. Since the maximum number of groups supported by a controller is eight (8), and the maximum valid value is 192 (0b11000000). However, RMI checks the value of the **GroupMask** against the groups that are actually used, and if the **GroupMask** specifies a group that does not exist in the controller's current configuration, RMI posts the error code **RMIT-040 Invalid Group Mask**.

RMI can be used to control up to two (2) robot groups, so **GroupMask** values consisting of up to two non-zero bits are allowed. **GroupMask** bits of more than two non-zero bits (e.g., 0b00000111) will result in the invalid group mask error. For **GroupMask** with two non-zero bits, RMI expects all the incoming motion instruction packets to have two sets of position data; one for the first group and one for the second group. If any motion instruction packet has only one group's position data, then RMI will return an error. On the other hand, if the **GroupMask** is set to run one group motion and the remote device sends motion instructions with more than one group's position data, RMI also posts an error.

For multiple group controllers, the **GroupMask** key is required. Otherwise, RMI will return **RMIT-040 Invalid Group Mask**.

NOTE

Please make sure the **GroupMask** is set correctly. If you set group mask to 1 and send RMI with group 2 position data, the controller will either post a run-time error due to invalid kinematics, or group 1 robot will go to the position intended for group 2 and cause unexpected motion.

2.3.2 Packet To Abort Remote Motion Program

Remote Device Command Packet	Robot Controller Return Packet
{“Command”: “FRC_Abort”} \r\n	{“Command”: “FRC_Abort”, “ErrorID”: integerValue} \r\n

Sending the abort packet to the robot controller allows you to abort the current running **RMI_MOVE TP** program. After this program is aborted, the robot controller sends a return packet back to the remote device. The **integerValue** in the return packet is 0 if the command is successfully executed. Otherwise, the **integerValue** is the controller's error code.

NOTE

Please always end your RMI session with either an **FRC_Abort** or **FRC_Disconnect** packet. This will ensure you can execute other TP programs after the RMI session.

2.3.3 Packet To Pause Remote Motion Program

Remote Device Command Packet	Robot Controller Return Packet
{“Command” : “FRC_Pause”} \r\n	{“Command”: “FRC_Pause”, “ErrorID”: integerValue} \r\n

To halt the RMI_MOVE TP program execution, send this packet to pause the program. If the integerValue in the return packet is 0, the TP Program is paused successfully. Otherwise, the integerValue is the error code from the robot controller.

2.3.4 Packet To Continue Remote Motion Program

Remote Device Command Packet	Robot Controller Return Packet
{“Command”: “FRC_Continue”} \r\n	{“Command”: “FRC_Continue”, “ErrorID”: integerValue} \r\n

To resume a halted TP program, send the continue packet to the controller to continue the TP program. If the program is continued, the returned integerValue is 0, otherwise, the integerValue contains the robot controller error code.

2.3.5 Packet To Read Controller Error

Remote Device Command Packet	Robot Controller Return Packet
{“Command”: “FRC_ReadError”, “Count”: byteValue1 } \r\n	{“Command”: “FRC_ReadError”, “ErrorID”: integerValue, “Count”: byteValue1, “ErrorData”: “Error Sting”, “ErrorData2”: “Error String2”, “ErrorData3”: “Error String3”, “ErrorData4”: “Error String4”, “ErrorData5”: “Error String5”, } \r\n

NOTE

The text with the gray background is optional and is only required for reading multiple error cases.

To get a controller error code, you can send the read error packet to the controller. The controller sends the most recent error text back in the return packet. The Error String has a length of 8 in a format of **XXXX-NNN**, where the **XXXX** is the sub-system error text, such as SRVO or MOTN. The **NNN** is the error code. Therefore, if the last error that the controller had was that the E-Stop button on the teach pendant had been pressed, the return error string will be **SRVO-002**.

The “Count” is an option key and its’ value has a valid range from 1 to 5. If the byteValue1 is less than 1 or greater than 5, RMI sends back a return packet with “ErrorID” = 2556949 (RMI_INV_VAL). The FRC_ReadError will return only the most recent error text if it does not have the optional key (compatible with the original specification).

If the controller has less errors than requested, RMI changes the return packet's byteValue1 to the number of error it can return.

If the error text has the optional key, the warning and not occurring error are not returned.

2.3.6 Packet To Set The Current UFrame-UTool Number

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_SetUFrameUTool”, “UFrameNumber”: byteValue1, “UToolNumber”: byteValue2, “Group”: byteValue3 } \r\n</pre>	<pre>{“Command”: “FRC_SetUFrameUTool”, “ErrorID”: integerValue, “Group”: byteValue3 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

You can set the controller's current user frame number and user tool number by sending this command packet to the controller, where byteValue1 is the user frame number and byteValue2 is the user tool number.

The Group key from remote device is optional. Without the Group key, RMI sets the User Frame number and User Tool number to group 1 by default.

The valid group number is from 1 to 8 since one controller can have up to 8 groups. RMI checks the input group number against the controller's robot configuration. If the input byteValue3 is not a valid group number, RMI returns error code **RMIT-039 Invalid Group Number**.

NOTE

You should not send this command packet while the robot is in motion since changing the frame number and tool number may cause a frame mismatch error and stop the TP program.

2.3.7 Packet To Get Controller Status

Remote Device command packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_GetStatus”} \r\n</pre>	<pre>{“Command”: “FRC_GetStatus”, “ErrorID”: integerValue, “ServoReady”: byteValue1, “TPMode”: byteValue2, “RMIMotionStatus”: byteValue3, “ProgramStatus”: byteValue4, “SingleStepMode”: byteValue5, “NumberUTool”: byteValue6, “NextSequenceID”: integerValue1, “NumberUFrame”: byteValue7} \r\n</pre>

The return packet of the Get Current Robot Status packet has the following information:

- **integerValue (ErrorID)** : if 0, the command is executed successfully. Otherwise, the **integerValue** contains the controller error code and all other values in the return packet are invalid.
- **byteValue1 (ServoReady)** : if it is 1, the robot controller is ready for motion. Otherwise, the robot controller is in an error condition. You can try an **FRC_Reset** packet to clear the error.
- **byteValue2 (TPMode)** : if it is 0, the teach pendant is disabled. If it is 1, the teach pendant is enabled. The Remote Motion interface only works when the teach pendant is disabled.
- **byteValue3 (RMIMotionStatus)** : if 1, the Remote Motion Interface is running. If 0, the Remote Motion interface is not running and you can send a **FRC_Initialize** packet to restart the motion interface.
- **byteValue4 (ProgramStatus)** : The **RMI_MOVE** TP program's current status. If 1, the **RMI_MOVE** TP program is aborted.
- **byteValue5 (SingleStepMode)** : If 1, the robot controller is in single-step mode.
- **byteValue6 (NumberUTool)** : Number of user tools available in the robot controller.
- **byteValue7 (NumberUFrame)** : Number of user frames available in the robot controller.
- **integerValue1 (NextSequenceID)** : The next valid sequence ID. This key is only valid if the system variable `$RMI_CFG.$Chk_seqID = TRUE`.

2.3.8 Packet To Read User Frame Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_ReadUFrameData”, “FrameNumber”: byteValue, “Group”: byteValue2 } \r\n</pre>	<pre>{“Command”: “FRC_ReadUFrameData”, “ErrorID”: integerValue, “UFrameNumber”: byteValue, “Frame”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR }, “Group”: byteValue2 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

You can read a user frame's data by sending a Read User Frame data packet where the **byteValue** is the user frame number. The return packet's **integerValue = 0** means that the position data is valid. Otherwise, the **integerValue** is the controller's error code.

The **Group** key from remote device is optional. Without the **Group** key, RMI returns the User Frame data for group 1 by default. If the input **byteValue2** is not a valid group number, RMI returns error code **RMIT-039**.

2.3.9 Packet To Set User Frame Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_WriteUFrameData”, “FrameNumber”: byteValue, “Frame”: {“X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR}, “Group”: byteValue2 } \r\n</pre>	<pre>{“Command”: “FRC_WriteUFrameData”, “ErrorID”: integerValue, “Group”: byteValue2 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

You can change a user frame's data by sending this packet. The **byteValue** is the user frame number. However, you should not change the user frame data when the robot is in motion since it may cause unexpected motion. The Remote Motion Interface rejects this packet if the robot is in motion when it receives this packet.

The **Group** key from the remote device is optional. Without the **Group** key, RMI sets the user frame data for group 1 by default. If the input **byteValue2** is not a valid group number, RMI returns error code **RMIT-039**.

2.3.10 Packet To Read User Tool Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_ReadUToolData”, “ToolNumber”: byteValue, “Group”: byteValue2 } \r\n</pre>	<pre>{“Command”: “FRC_ReadUToolData”, “ErrorID”: integerValue, “UToolNumber”: byteValue, “Frame”: {“ X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatW, “R”: floatR}, “Group”: byteValue2 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

You can read a user frame data by sending this packet where the **byteValue** is the tool frame number. The return packet's **floatX**, **floatY**, **floatZ**, **floatW**, **floatP**, **floatR** are the Cartesian representations of the tool frame **byteValue**.

The **Group** key from the remote device is optional. Without the **Group** key, RMI reads the User Tool data for group 1 by default. If the input **byteValue2** is not a valid group number, RMI returns error code **RMIT-039**.

2.3.11 Packet To Set User Tool Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_WriteUToolData”, “ToolNumber”: byteValue, “Frame”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR}, “Group”: byteValue2 } \r\n</pre>	<pre>{“Command”: “FRC_WriteUToolData”, “ErrorID”: integerValue, “Group”: byteValue2 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

Use this command packet to set a user tool frame, where the floatX, floatY, floatZ, floatW, floatP and floatR contain the Cartesian position data for the user tool byteValue.

The Group key from the remote device is optional. Without the Group key, RMI writes the user tool data to group 1 by default.

If the input byteValue2 is not a valid group number, RMI returns error code **RMIT-039**.

NOTE

You should not change the user tool frame data when the robot is in motion since it may cause unexpected motion. Please make sure that the robot is not in motion before sending this packet.

2.3.12 Packet To Read Input Port

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_ReadDIN”, “PortNumber”: shortValue} \r\n</pre>	<pre>{“Command”: “FRC_ReadDIN”, “ErrorID”: integerValue, “PortNumber”: shortValue, “PortValue”: byteValue} \r\n</pre>

You can read a digital input port value using this packet, where the shortValue indicates the port number. If the return packet's integerValue = 0, then the byteValue is the current value of DINport shortValue.

2.3.13 Packet To Write Digital Output Port

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_WriteDOUT”, “PortNumber”: shortValue, “PortValue”: StringValue} \r\n</pre>	<pre>{“Command”: “FRC_WriteDOUT”, “ErrorID”: integerValue} \r\n</pre>

Use this packet to set a digital output port where the shortValue is the port number and the stringValue is either ON or OFF. In the return packet, if the integerValue = 0, then the command is successfully executed.

2.3.14 Packet To Read Current Robot Cartesian Position

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_ReadCartesianPosition”, “Group”: byteValue } \r\n</pre>	<pre>{“Command”: “FRC_ReadCartesianPosition”, “ErrorID”: integerValue1, “TimeTag”: integerValue2, “Configuration”: { “UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9}, “Position”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR, “Ext1”: floatE1, “Ext2”: floatE2, “Ext3”: floatE3}, “Group”: byteValue10 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

Use this packet to get the current robot’s Cartesian position. If the return packet’s `integerValue1` = 0, the following position data are valid. Otherwise, the `integerValue1` is the error code from the controller. The `integerValue2` is the tick time from the robot controller.

The **Group** key from the remote device is optional. Without the **Group** key, RMI reads the robot position data from group 1 by default. If the input `byteValue` is not a valid group number, RMI returns error code **RMIT-039**.

2.3.15 Packet To Read Current Robot Joint Angles

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_ReadJointAngles”, “Group”: byteValue } \r\n</pre>	<pre>{“Command”: “FRC_ReadJointAngles”, “ErrorID”: integerValue1, “TimeTag”: integerValue2, “JointAngles”: {“J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “Group”: byteValue } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

Use this packet to get the current robot’s joint positions. If the return packet’s integerValue1 = 0, the following joint angle data are valid. Otherwise, the integerValue1 is the error code from the controller.

The Group key from the remote device is optional. Without the Group key, RMI reads the robot joint angles from group 1 by default. If the input byteValue is not a valid group number, RMI returns error code **RMIT-039**.

2.3.16 Packet To Set Speed Override

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_SetOverRide”, “Value”: byteValue} \r\n</pre>	<pre>{“Command”: “FRC_SetOverRide”, “ErrorID”: integerValue} \r\n</pre>

Send this packet to change the program override value. The allowable range for byteValue is from 1 to 100. The return packet’s integerValue = 0 means that the command is executed successfully.

2.3.17 Packet To Get Current User/Tool Frame Number

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_GetUFrameUTool”, “Group”: byteValue } \r\n</pre>	<pre>{“Command”: “FRC_GetUFrameUtool”, “ErrorID”: integerValue, “UFrameNumber”: byteValue1, “UToolNumber”: byteValue2, “Group”: byteValue3 } \r\n</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

Use this packet to get the current user frame number and user tool number of the robot controller. When the return packet's `integerValue` = 0, then the `byteValue1` is the current user frame number, and the `byteValue2` is the current robot user tool number. If the `integerValue` > 0, then the `integerValue` is the robot's error code and the `byteValue1` and `byteValue2` are invalid.

The `Group` key from the remote device is optional. Without the `Group` key, RMI reads the current user tool number and user frame number from group 1 by default. If the input `byteValue` is not a valid group number, RMI returns error code **RMIT-039**.

2.3.18 Packet To Read Position Register Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{ "Command": FRC_ReadPositionRegister, "RegisterNumber": shortValue, "Group": byteValue }</pre>	<pre>{ "Command": "FRC_ReadPositionRegister", "ErrorID": integerValue, "RegisterNumber": shortValue, "Configuration": { "UtoolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "Group": byteValue10 }</pre>

NOTE

The text with the gray background is optional and only required for a multiple group system.

Use this packet to read the position register data from the robot controller where the `shortValue` is the register number. If the return packet's `integerValue` = 0, then the following position data is valid. Otherwise, the `integerValue` is the robot's error code.

The `Group` key from the remote device is optional. Without the `Group` key, RMI returns the group 1 position from the position register by default. If the input `byteValue` is not a valid group number, RMI returns error code **RMIT-039**.

2.3.19 Packet To Write Position Register Data

Remote Device Command Packet	Robot Controller Return Packet
<pre>{ "Command": "FRC_WritePositionRegister", "RegisterNumber": shortValue, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "Group": byteValue10 } }</pre>	<pre>{ "Command": "FRC_WriteUFrameData", "ErrorID": integerValue }</pre>

Use this packet to write data into a position register number (**shortValue**). When the **integerValue** in the return packet = 0 it means the position register is set. Otherwise, the **integerValue** is the error code from the robot.

The **Group** key from the remote device is optional. Without the **Group** key, RMI returns the group 1 position from the position register by default. If the input **byteValue10** is not a valid group number, RMI returns error code **RMIT-039**.

2.3.20 Packet to Reset Robot Controller

Remote Device Command Packet	Robot Controller Return Packet
<pre>{ "Command": "FRC_Reset" }</pre>	<pre>{ "Command": "FRC_Reset", "ErrorID": integerValue }</pre>

Use this packet to reset errors in the controller and make the robot ready to receive motion commands. When the return packet's **integerValue** = 0 it indicates that the reset is successful. Otherwise, the **integerValue** contains the controller's error code.

NOTE

Before V9.30P/19 (7DF3/19) and V9.40P/18 (7DF5/18), **FRC_Reset** only works after RMI received **FRC_Initialize** command. After V9.30P/19 (7DF3/19) and V9.40P/18 (7DF5/18), **FRC_Reset** can work without user sending **FRC_Initialize** command.

NOTE

It takes some time to get the system reset, please wait till you receive the return packet before sending another command packet to the robot controller.

2.3.21 Packet To Read Current Tool Center Point Speed

Remote Device Command Packet	Robot Controller Return Packet
{“Command”: “FRC_ReadTCPSpeed”} \r\n	{“Command”: “FRC_ReadTCPSpeed”, “ErrorID”: integerValue1, “TimeTag”: integerValue2, “Speed”: floatValue} \r\n

Use this packet to get the current tool center point (TCP) speed. The controller sends back a packet where the floatValue is the current TCP speed value in mm/sec.

2.4 TEACH PENDANT PROGRAM INSTRUCTION PACKETS

When the controller receives an Initialization Packet (FRC_Initialize), it creates a TP program, RMI_MOVE, and starts the execution of this program. Please wait for the FRC_Initialize packet to be returned before sending TP instruction packets to the controller. Otherwise, these instruction packets will be returned with an error code. You can use the packets listed in this section to add a new TP instruction to the end of this running program. The controller will execute this instruction after it completes all other instructions sent previously.

The RMI_MOVE TP Program is 200 instructions in size and acts like a ring buffer. When the controller receives the 201st instruction, it puts this instruction at the beginning of the TP program and executes it when it completes the 200th instruction.

All the packets listed in this section add a new line of TP instruction to the RMI_MOVE TP program. You can send eight instructions to the controller at a time. However, the controller will not accept the next instruction until the first instruction is executed and the tenth instruction has to wait for the second instruction to complete.

Note that a motion instruction with continuous termination type (CNT1-100) will not execute until the next motion line comes in, to make sure the two motions can blend together correctly. Therefore, make sure that the last motion instruction you send to the controller is with FINE termination type. Otherwise, the last motion instruction with the CNT termination type will not be executed.

NOTE

After version 5, RMI allows the last motion instruction to be executed if the motion has the NoBlend flag set in that instruction, regardless of the termination type.

Each instruction packet has an ID number (SequenceID), and its value is assigned by the remote device to uniquely identify this instruction packet. This ID number should be consecutive and monotonically increasing. When this instruction is completed, the robot controller sends the return packet with the same key SequenceID and value back to the remote device so that the remote device knows which instruction has been executed.

NOTE

Start your SequenceID number from 1 after the FRC_Initialize packet. (Note that if the SequenceID should ever reach its maximum value of $2^{31}-1$, it will wrap back to 1.) By default, RMI checks the SequenceID to make sure there are no missing packets. If RMI detects a non-consecutive sequence ID, RMI sends a **RMIT-029 Invalid sequence ID number** error ID back to the sender. At this point, RMI goes into a HOLD state. While in a HOLD state, RMI continues to execute the TP instructions that are already in the TP program but will not accept new TP instructions until RMI receives the FRC_Reset command. You can get the correct sequence ID by sending an FRC_GetStatus packet and getting "NextSequenceID" : nnnn where the nnnn is the next valid sequence ID. You can also turn off the sequence ID check by setting the system variable *\$RMI_CFG.chk_seqID* = FALSE.

The Remote Motion interface supports a limited set of TP instructions and they are listed here.

Table 2.4 (a) Specification of TP PROGRAM INSTRUCTION PACKETS

Packet Name	TP Instruction Example
FRC_WaitDIN	WAIT DI[1] = ON
FRC_SetUFrame	UFRAME_NUM = 1
FRC_SetUTool	UTOOL_NUM = 2
FRC_WaitTime	WAIT 0.5 (sec)
FRC_SetPayLoad	PAYLOAD[2]
FRC_Call	Call Do_IO_Check
FRC_LinearMotion	L P[10] 100 mm/sec CNT100
FRC_LinearRelative	L P[11] 150 mm/sec FINE INC
FRC_JointMotion	J P[12] 5% FINE
FRC_JointRelative	J P[12] 10% CNT100 INC
FRC_CircularMotion	C P[13] P[14] 100 mm/sec FINE INC
FRC_CircularRelative	C P[13] P[14] 100 mm/sec FINE INC
FRC_JointMotionJRep	J P[16] 5% FINE INC
FRC_JointRelativeJRep	L P[10] 100 mm/sec CNT100
FRC_LinearMotionJRep	L P[11] 150 mm/sec FINE INC

Any instruction that is not in this list is not supported.

For motion instructions, the Remote Motion only supports the following motion options:

- MORT
- WJnt
- Offset, PR[]
- COORD
- ACC
- VODDSET, VR[]
- Local Condition Block, TA/TB/DB

- ALIM

NOTE

For V9.30P/13 (7DF3/13) and later software and V9.40P/08 (7DF5/08) and later software, RMI supports ASCII instruction strings that covers more TP instructions. Please see Chapter 5, ASCII STRING INSTRUCTION PACKETS for more details.

2.4.1 Packet To Wait For DIN Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction”: “FRC_WaitDIN”, “SequenceID”: integerValue, “PortNumber”: shortValue, “portValue”: “ON” or “OFF”} \r\n	{“Instruction”: “FRC_WaitDIN”, “ErrorID”: integerValue1, “SequenceID”: integerValue2 } \r\n

The FRC_WaitDIN packet adds a TP instruction WAIT DI[shortValue] = ON/OFF to the TP program.

The controller sends the return packet back after the TP instruction is executed. The integerValue1 is 0 when this instruction is successfully executed. Otherwise, the integerValue1 returns an error code.

2.4.2 Packet To Set User Frame Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction”: “FRC_SetUFrame”, “SequenceID”: integerValue, “FrameNumber”: byteValue} \r\n	{“Instruction”: “FRC_SetUFrame”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n

The FRC_SetUFrame packet adds a TP instruction UFRAME_NUM = byteValue to the running TP program. The controller sends the return packet back after this instruction is executed and the RMI_Move TP program can advance to the next time. The integerValue1 is set to 0 if this instruction is successfully executed. Otherwise, the integerValue1 is the error code.

2.4.3 Packet To Set User Tool Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction”: “FRC_SetUTool”, “SequenceID”: integerValue, “ToolNumber”: byteValue} \r\n	{“Instruction”: “FRC_SetUtool”, “ErrorID”: integerValue1, “SequenceID”: integerValue} \r\n

The FRC_SetUTool packet adds a TP instruction UTOOL_NUM = byteValue to the running TP program. The controller sends the return packet back after this instruction is executed and the return integerValue1 is set to 0 if this instruction is successfully executed. Otherwise, the integerValue1 is the error code.

2.4.4 Packet To Add Wait Time Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction”: FRC_WaitTime”, “SequenceID”: integerValue, “Time” : floatValue} \r\n	{“Instruction”: “FRC_WaitTime”, “ErrorID”: integerValue1, “SequenceID : integerValue2} \r\n

The FRC_WaitTime packet adds a TP instruction WAIT floatValue(sec) to the running TP program. The controller sends the return packet back after this instruction is executed and the RMI_MOVE TP program can advance to the next line. The integerValue1 is set to 0 if this instruction is successfully executed. Otherwise, the integerValue1 is the error code.

2.4.5 Packet To Set Payload Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction”: “FRC_SetPayLoad”, “SequenceID”: integerValue, “ScheduleNumber”: byteValue} \r\n	{“Instruction”: “FRC_SetPayLoad”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n

The FRC_SetPayLoad packet adds a PAYLOAD[byteValue] instruction to the running TP program. The controller sends the return packet back after this instruction is executed and the integerValue1 is set to 0 if this instruction is successfully executed. Otherwise, the integerValue1 is the error code.

2.4.6 Packet To Call A Program

Remote Device Instruction Packet	Robot Controller Return Packet
{“Instruction” : “FRC_Call”, “SequenceID” : integerValue, “ProgramName” : String} \r\n	{“Instruction” : “FRC_Call”, “ErrorID” : integerValue1, “SequenceID” : integerValue2} \r\n

The “ProgramName” String can be up to 36 bytes in length. RMI returns an error if the String is more than 36 bytes long.

RMI sends the return packet back when the called program is executed and the RMI_MOVE TP program can advance to the next line. Therefore, if the CALL instruction is the last line in the RMI_MOVE, RMI will not return the FRC_CALL packet back until RMI receives a new instruction.

2.4.7 Packet To Add Linear Motion Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_LinearMotion", "SequenceID": integerValue, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue10, </pre>	<pre>{ "Instruction": "FRC_LinearMotion", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n</pre>
<p>The following items are optional.</p> <pre> "ACC": byteValue11, "OffsetPRNumber": shortValue2, "VisionPRNumber": shortValue3, "WristJoint": "ON", "MROT": "ON", "LCBType": stringValue3, "LCBValue": shortValue4, "PortType": byteValue12, "PortNumber": ShortValue5, "PortValue": stringValue4, "ToolOffsetPRNumber": shortValue6, "ALIM": integerValue1, "ALIMREG": shortValue7, "NoBlend" : "ON"} \r\n </pre>	

The FRC_LinearMotion packet adds a linear motion instruction to the running TP program.

The basic motion command has the following items that you can change:

- SpeedType: the stringValue1 can be
 - mmSec: mm/sec

- InchMin: 0.1 inch/min
 - Time: 0.1 seconds
 - mSec : milliseconds
- Speed: shortValue1 is the speed value.
 - TermType: stringValue2 can be either FINE, CNT or CR. If the TermType is FINE, the byteValue10 is ignored.
 - TermValue: byteValue10 is the continuous termination type; value from 1-100.

NOTE

The CR TermType is only valid when the Advance Constant Path option is loaded. If the robot does not have this option, the controller returns an error. CR is similar to the CNT termination type. It requires the next motion instruction to plan its motion. Therefore, please do not use the CR termination type for the last motion you send to the controller, since this motion will not be executed.

For example, assuming the stringValue1 = mmSec, shortValue1 = 150 with CNT termination type and byteValue10 = 100; then the following instruction is added to the TP program:

L P[10] 150mm/sec CNT100

As previously noted, the optional items above can be added to the motion instruction. The following specifications apply to these options:

- ACC: byteValue11. You can add an ACC option to the motion instruction, where the valid byteValue11 is from 1 to 100. Assuming byteValue11 is 80, then the added instruction is L P[10] 150 mm/sec CNT100 ACC80.
- OffsetPRNumber: shortValue2. You can also add a position register offset to the motion instruction. The shortValue2 is the register number. Assuming the shortValue2 = 2, then the instruction becomes L P[10] 150 mm/sec CNT100 Offset, PR[2].
- ToolOffsetPRNumber: shortValue6. You can also add a tool position register offset to the motion instruction. The shortValue6 is the register number. Assuming the shortValue6 = 2, then the instruction becomes L P[10] 100 mm/sec CNT100 Tool_Offset, PR[2].
- VisionPRNumber: shortValue3 adds a vision offset to the linear motion. The shortValue3 is the vision offset position register number. Assuming the shortValue3 = 5 then the linear motion instruction becomes L P[10] 100 mm/sec CNT100 VOFFSET, VR[5].
- WristJoint: ON adds a Wrist Joint motion option to the linear motion instruction.

NOTE

This option only applies to Linear and Circular motion and it is ignored when used in a joint motion instruction.

With this option, the linear motion instruction becomes L P[10] 100 mm/sec CNT100 Wjnt .

- MROT: ON adds a MROT motion option to the linear motion instruction. With this option, the linear motion instruction becomes:

L P[10] 100 mm/sec CNT100 MROT

NOTE

The MROT is available only when the R640 option is loaded on the controller. Otherwise, the controller will return an error for this instruction.

- **LCBType: stringValue3** = TB, DB, or TA: There are three LCB (Local Condition Block) types:
 - *TB*: Time before the robot reaches the destination point. The unit for TB is seconds.
 - *TA*: Time after the robot reaches the destination point. The unit for TA is seconds.
 - *DB*: Distance before the robot reaches the destination point. The unit for DB is mm.

The LCB is used to trigger an output port when the robot reaches a certain position with respect to the destination position.

 - **LCBValue: shortValue4**. This shortValue4 indicates the TB/TA/DB value. For DB, the unit is 0.01 mm. For TA and TB, the unit is ms.
 - **PortType: byteValue12**. The byteValue12 can be either
 - 1: DOUT
 - 2: ROUT

NOTE

GO and AO are not supported

- **PortNumber : shortValue5**: The shortValue5 is the DOUT/ROUT port number.
- **PortValue : stringValue4**. The string can either be ON or OFF.

For example, given the LCBType: TA and LCBValue: 100, PortType: 1, PortNumber: 3 and PortValue: ON, the motion instruction becomes:

L P[10] 150mm/sec CNT10 TA 0.10 sec DO[3] = ON

After the controller executes this linear motion instruction, the controller sends the return packet back. If the integerValue1 = 0 it means that the motion instruction has been successfully executed. If integerValue1 > 0 it means that there was an error while executing this motion and the integerValue1 is the error code.

NOTE

The MROT, Offset PR, Offset VR options change the robot's trajectory and therefore, if these setting are invalid, the controller will not insert these motion instruction in the TP program and will return the motion instruction right away. At this point, the robot will not accept any new TP instruction until the controller receives an FRC_Reset command.

- **ALIM:Integer1** adds an Acceleration Limit motion modifier to the motion instruction. The unit is in mm/sec² and the maximum allowed value is 10000.
- **ALIMREG : ShortValue7** adds an Acceleration Limit with Register motion modifier to the motion instruction. If the ShortValue7 is less than 1 and bigger than existing register number, the program execution will post an error and stop program execution. If the register is not initialized, program execution stops and an error is posted.
- **NoBlend:ON** allows a motion with CNT termination type to be executed without waiting for next motion. By default, NoBlend is always OFF.

NOTE

This is a special motion modifier for RMI only, It does not add any text to the motion instruction.

RMI provides abbreviations for some of the key strings. These abbreviations are listed below:

- “SequenceID”(“SID”)
- “Turn4”(“T4”)
- “Turn5”(“T5”)
- “Turn6”(“T6”)
- “UToolNumber”(“UTNum”)
- “UFrameNumber”(“UFNum”)

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.7.1 Packet to add two groups linear motion instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_LinearMotion", "SequenceID": integerValue, "G1": { "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}}, "G2": { "Configuration": { "UToolNumber": byteValue10, "UFrameNumber": byteValue11, "Front": byteValue12, "Up": byteValue13, "Left": byteValue14, "Flip": byteValue15, "Turn4": byteValue16, "Turn5": byteValue17, "Turn6": byteValue18}, "Position": { "X": floatX2, "Y": floatY2, "Z": floatZ2, "W": floatW2, "P": floatP2, "R": floatR2, "Ext1": floatE4, "Ext2": floatE5, "Ext3": floatE6}}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue19,</pre>	<pre>{ "Instruction": "FRC_LinearMotion", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n</pre>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>“ACC”: byteValue20,</p> <p>“OffsetPRNumber”: shortValue2,</p> <p>“VisionPRNumber”: shortValue3,</p> <p>“WristJoint”: stringValue3: “ON” or “OFF”,</p> <p>“MROT”: stringValue4: “ON” or “OFF”,</p> <p>“Coord”: stringValue5: “ON” or “OFF”,</p> <p>“LCBType”: stringValue6,</p> <p>“LCBValue”: shortValue4,</p> <p>“PortType”: byteValue21,</p> <p>“PortNumber”: ShortValue5,</p> <p>“PortValue”: stringValue7,</p> <p>“ToolOffsetPRNumber”: shortValue6,</p> <p>“NoBlend” : “ON”}\r\n</p>	

NOTE

G1 indicates group 1 position and G2 is for group 2. For controllers with more than two groups, G3, G4, G5 and G6 are valid as long as the group mask is set accordingly. For example, if the FRC_Initialize packet has GroupMask = 5, RMI expects to see both G1 and G3 in all RMI motion instructions sent to the controller. All other combinations of Gx are invalid.

The first group position should have the same representation as required by the packet. For example, G1 has to have Cartesian position representation when it is the first position data in the FRC_LinearMotion packet. The second position can be either Cartesian representation or joint representation. RMI does not check the second position’s representation type. If you have position data such that G1 has joint angles and G2 has Cartesian position, you can still use FRC_LinearMotion with G2 be the first position data in the packet.

The Coord option is valid only if the RMI_MOVE TP program has 2 groups. Otherwise, an error is returned to the remote device. RMI does not check the coordinated motion setup when it receives the COORD option key, it is user’s responsibility to set up the coordinate motion pair before running the RMI. Otherwise, you will get run time error posted by the motion system.

If the second group is a positioner, the FRC_LinearMotion Instruction should be as follows:

Remote Device Instruction Packet	Robot Controller Return Packet
<pre> {“Instruction”: “FRC_LinearMotion”, “SequenceID”: integerValue, “G1”: { “Configuration”: { “UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9}, “Position”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR, “Ext1”: floatE1, “Ext2”: floatE2, “Ext3”: floatE3}}, “G2”: { “JointAngle”: { “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue10, </pre>	<pre> {“Instruction”: “FRC_LinearMotion”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n </pre>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>“ACC”: byteValue11,</p> <p>“OffsetPRNumber”: shortValue2,</p> <p>“VisionPRNumber”: shortValue3,</p> <p>“WristJoint”: stringValue3: “ON” or “OFF”,</p> <p>“MROT”: stringValue4: “ON” or “OFF”,</p> <p>“Coord”: stringValue5: “ON” or “OFF”,</p> <p>“LCBType”: stringValue6,</p> <p>“LCBValue”: shortValue4,</p> <p>“PortType”: byteValue12,</p> <p>“PortNumber”: ShortValue5,</p> <p>“PortValue”: stringValue7,</p> <p>“ToolOffsetPRNumber”: shortValue6,</p> <p>“NoBlend” : “ON”}\r\n</p>	

NOTE

JointAngle does not have to have all nine axes defined. If the positioner has only two axes, the group 2 can be simplified as “G2” : { “JointAngle” : { “J1” : floatJ1, “J2” : floatJ2 } }, However, if the positioner has two axes and the remote device only provides one joint angle, RMI assumes J2 is at 0.

2.4.8 Packet To Add Linear Incremental Motion Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<p>{“Instruction”: “FRC_LinearRelative”,</p> <p>“SequenceID”: integerValue,</p> <p>“Configuration”: {</p> <p>“UToolNumber”: byteValue1,</p> <p>“UFrameNumber”: byteValue2,</p> <p>“Front”: byteValue3, “Up”: byteValue4,</p> <p>“Left”: byteValue5, “Flip”: byteValue6,</p> <p>“Turn4”: byteValue7, “Turn5”: byteValue8,</p> <p>“Turn6”: byteValue9},</p> <p>“Position”: {</p> <p>“X”: floatX, “Y”: floatY, “Z”: floatZ,</p> <p>“W”: floatW, “P”: floatP, “R”: floatR,</p> <p>“Ext1”: floatE1, “Ext2”: floatE2,</p> <p>“Ext3”: floatE3},</p> <p>“SpeedType”: stringValue1,</p> <p>“Speed”: shortValue1,</p> <p>“TermType”: stringValue2,</p> <p>“TermValue”: byteValue10,</p>	<p>{“Instruction”: “FRC_LinearRelative”,</p> <p>“ErrorID”: integerValue1,</p> <p>“SequenceID”: integerValue2} \r\n</p>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>“ACC” : byteValue11, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue4, “PortType” : byteValue12, “PortNumber” : shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “ALIM”: integerValue1, “ALIMREG”: shortValue7, “NoBlend”:“ON”} \r\n</p>	

The FRC_LinearRelative packet is the same as the FRC_LinearMotion, except the position data is a relative position. For a simple FRC_LinearRelative packet, the controller creates a TP line as:

L P[10] 150mm/sec CNT100 INC

After the controller executes this incremental linear motion instruction, the controller sends the return packet back. The return value integerValue1 = 0 means that the motion instruction has been successfully executed. If integerValue1 > 0, there was an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.8.1 Packet to add two groups linear incremental motion instruction

The two groups FRC_LinearRelative instruction shares the same protocol as the two groups FRC_LinearMotion instruction defined in Section 2.4.7.1, Packet to add two groups linear motion instruction. Please use the protocol defined in Section 2.4.7.1, Packet to add two groups linear motion instruction, change the value of the Instruction key from FRC_LinearMotion to FRC_LinearRelative, replace the absolute position data with incremental position and you have a two groups linear incremental motion packet for RMI.

2.4.9 Packet To Add Joint Motion Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_JointMotion", "SequenceID": integerValue, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue10, } }</pre>	<pre>{ "Instruction": "FRC_JointMotion", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n</pre>
<p>The following items are optional.</p> <pre>"ACC": byteValue11, "OffsetPRNumber": shortValue2, "VisionPRNumber": shortValue3, "MROT": "ON", "LCBType": stringValue3, "LCBValue": shortValue4, "PortType": byteValue12, "PortNumber": shortValue5, "PortValue": stringValue4, "ToolOffsetPRNumber": shortValue6, "NoBlend": "ON"} \r\n</pre>	

The FRC_JointMotion packet is the same as the FRC_LinearMotion, except that the motion type is joint instead of linear.

In addition, the joint motion only supports the following two speed types:

- Percent: %
- Time: 0.1 sec
- mSec : milliseconds

For a simple FRC_JointMotion packet, the controller creates a TP line such as:

J P[10] 10% CNT100

After the controller executes this joint motion instruction, the controller sends the return packet back. When the return value `integerValue1 = 0` it means that the motion instruction has been successfully executed. If `integerValue1 > 0`, there is an error in executing the motion and the `integerValue1` is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.9.1 Packet to add two group joint motion instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_JointMotion”, “SequenceID”: integerValue, “G1”: {“Configuration”: {“UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9}, “Position”: {“X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR, “Ext1”: floatE1, “Ext2”: floatE2, “Ext3”: floatE3}}, “G2”: {“Configuration”: { “UToolNumber”: byteValue10, “UFrameNumber”: byteValue11, “Front”: byteValue12, “Up”: byteValue13, “Left”: byteValue14, “Flip”: byteValue15, “Turn4”: byteValue16, “Turn5”: byteValue17, “Turn6”: byteValue18}, “Position”: { “X”: floatX2, “Y”: floatY2, “Z”: floatZ2, “W”: floatW2, “P”: floatP2, “R”: floatR2, “Ext1”: floatE4, “Ext2”: floatE5, “Ext3”: floatE6}}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue19,</pre>	<pre>{“Instruction”: “FRC_JointMotion”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n</pre>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following are optional.</p> <p>“ACC”: byteValue20,</p> <p>“OffsetPRNumber”: shortValue2,</p> <p>“VisionPRNumber”: shortValue3,</p> <p>“MROT”: stringValue3: “ON” or “OFF”,</p> <p>“LCBType”: stringValue4,</p> <p>“LCBValue”: shortValue4,</p> <p>“PortType”: byteValue21,</p> <p>“PortNumber”: ShortValue5,</p> <p>“PortValue”: stringValue5,</p> <p>“ToolOffsetPRNumber”: shortValue6,</p> <p>“NoBlend” : “ON”}\r\n</p>	

NOTE

G1 indicates group 1 position and G2 is for group 2. For controller with more than two groups, G3, G4 are valid as long as the group mask is set accordingly by the FRC_Initialize command.

NOTE

Joint motion does not support the COORD motion option. If you specify COORD in a joint motion instruction, RMI will return an error code.

2.4.10 Packet To Add Joint Incremental Motion Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_JointRelative”, “SequenceID”: integerValue, “Configuration”: { “UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9}, “Position”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR, “Ext1”: floatE1, “Ext2”: floatE2, “Ext3”: floatE3}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue10,</pre>	<pre>{“Instruction” : “FRC_JointRelative”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n</pre>
<p>The following items are optional.</p> <pre>“ACC” : byteValue11, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue4, “PortType” : byteValue12, “PortNumber” : shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”} \r\n</pre>	

The FRC_JointRelative packet is the same as FRC_JointMotion, except the position is a relative position. For a simple FRC_JointRelative packet, the controller creates a TP line as:

J P[10] 10% CNT100 INC

After the controller executes this incremental joint motion instruction, the controller sends the return packet back. The return value integerValue1 = 0 means that the motion instruction was successfully executed. If integerValue1 > 0, there is an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.10.1 Packet to add two groups joint incremental motion instruction

The two groups the `FRC_JointRelative` instruction shares the same protocol as the two groups `FRC_JointMotion` instruction defined in Section 2.4.9.1, Packet to add two group joint motion instruction. Please use the same protocol defined in Section 2.4.9.1, Packet to add two group joint motion instruction, change the value of the `Instruction` key from `FRC_JointrMotion` to `FRC_JointRelative`, replace the absolute joint angles with incremental joint angles and you have the two groups joint incremental motion packet for RMI.

2.4.11 Packet To Add Circular Motion Instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_CircularMotion", "SequenceID": integerValue, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "ViaConfiguration": { "UtoolNumber": byteValue10, "UFrameNumber": byteValue11, "Front": byteValue12, "Up": byteValue13, "Left": byteValue14, "Flip": byteValue15, "Turn4": byteValue16, "Turn5": byteValue17, "Turn6": byteValue18}, "ViaPosition": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue19, </pre>	<pre>{ "Instruction": "FRC_CircularMotion", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n </pre>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>“ACC”: byteValue20, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType”: stringValue3, “LCBValue”: shortValue4, “PortType”: byteValue21, “PortNumber”: shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”} \r\n</p>	

The FRC_CircularMotion packet has two sets of position data: Destination position data and via position data, as shown in the packet. This packet adds a circular motion instruction to the running TP program:

C P[10]

P[11] 15mm/sec CNT100

The circular motion supports the same motion options as the linear motion. Please refer to the linear motion instruction for a detailed description.

After the controller executes this circular motion instruction, the controller sends the return packet back. The return value integerValue1 = 0 means that the motion instruction was successfully executed. If integerValue1 > 0, there is an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.11.1 Packet to add two groups circular motion instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<p>{“Instruction”: “FRC_CircularMotion”, “SequenceID”: integerValue, “G1”: {“Configuration”: {“UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9}, “Position”: {“X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR,</p>	<p>{“Instruction”: “FRC_JointMotion”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n</p>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>“Ext1”: floatE1, “Ext2”: floatE2, “Ext3”: floatE3},</p> <p>“ViaConfiguration”: {</p> <p>“UtoolNumber”: byteValue10,</p> <p>“UFrameNumber”: byteValue11,</p> <p>“Front”: byteValue12,</p> <p>“Up”: byteValue13,</p> <p>“Left”: byteValue14,</p> <p>“Flip”: byteValue15,</p> <p>“Turn4”: byteValue16,</p> <p>“Turn5”: byteValue17,</p> <p>“Turn6”: byteValue18},</p> <p>“ViaPosition”: {</p> <p>“X”: floatX2, “Y”: floatY2, “Z”: floatZ2,</p> <p>“W”: floatW2, “P”: floatP2, “R”: floatR2,</p> <p>“Ext1”: floatE4, “Ext2”: floatE5,</p> <p>“Ext3”: floatE6}},</p> <p>“G2”: {“Configuration”: {</p> <p>“UToolNumber”: byteValue19,</p> <p>“UFrameNumber”: byteValue20,</p> <p>“Front”: byteValue21, “Up”: byteValue22,</p> <p>“Left”: byteValue23, “Flip”: byteValue24,</p> <p>“Turn4”: byteValue25, “Turn5”: byteValue26,</p> <p>“Turn6”: byteValue27},</p> <p>“Position”: {</p> <p>“X”: floatX3, “Y”: floatY3, “Z”: floatZ3,</p> <p>“W”: floatW3, “P”: floatP3, “R”: floatR3,</p> <p>“Ext1”: floatE7, “Ext2”: floatE8,</p> <p>“Ext3”: floatE9},</p> <p>“ViaConfiguration”: {</p> <p>“UtoolNumber”: byteValue28,</p> <p>“UFrameNumber”: byteValue29,</p> <p>“Front”: byteValue30,</p> <p>“Up”: byteValue31,</p> <p>“Left”: byteValue32,</p> <p>“Flip”: byteValue33,</p> <p>“Turn4”: byteValue34,</p> <p>“Turn5”: byteValue35,</p> <p>“Turn6”: byteValue36},</p> <p>“ViaPosition”: {</p> <p>“X”: floatX4, “Y”: floatY4, “Z”: floatZ4,</p> <p>“W”: floatW4, “P”: floatP4, “R”: floatR4,</p> <p>“Ext1”: floatE10, “Ext2”: floatE11,</p>	

Remote Device Instruction Packet	Robot Controller Return Packet
“Ext3”: floatE12}}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue37,	
The following items are optional. “ACC”: byteValue38, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “WristJoint”: stringValue3: “ON” or “OFF”, “MROT”: stringValue4: “ON” or “OFF”, “COORD”: stringValue5: “ON” or “OFF”, “LCBType”: stringValue6, “LCBValue”: shortValue4, “PortType”: byteValue39, “PortNumber”: shortValue5, “PortValue”: stringValue7, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”}\r\n	

2.4.12 Packet To Add Circular Incremental MotionInstruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_CircularRelative", "SequenceID": integerValue, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "ViaConfiguration": { "UToolNumber": byteValue10, "UFrameNumber": byteValue11, "Front": byteValue12, "Up": byteValue13, "Left": byteValue14, "Flip": byteValue15, "Turn4": byteValue16, "Turn5": byteValue17, "Turn6": byteValue18}, "ViaPosition": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue19, </pre>	<pre>{ "Instruction": "FRC_CircularRelative", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n </pre>

Remote Device Instruction Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>“ACC”: byteValue20, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType”: stringValue3, “LCBValue”: shortValue4, “PortType”: byteValue21, “PortNumber”: ShortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”} \r\n</p>	

The FRC_CircularRelative packet is the same as the FRC_CircularMotion, except the destination and via position are relative. This packet adds the following line to the running TP program:

C P[10]
P[11] 15mm/sec CNT100 INC

After the controller executes this incremental circular motion instruction, the controller sends the return packet back. If the return value integerValue1 = 0 it means that the motion instruction was successfully executed. If integerValue1 > 0, there was an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “EXT1”, “EXT2” and “EXT3” keys in the Position.

2.4.12.1 Packet to add two groups circular incremental motion instruction

The two groups FRC_CircularRelative instruction shares the same protocol as the two groups FRC_CircularMotion instruction defined in Section 2.4.11.1, Packet to add two groups circular motion instruction. Please use the same protocol defined in Section 2.4.11.1, Packet to add two groups circular motion instruction, change the value of the Instruction key from FRC_CircularMotion to FRC_CircularRelative, replace the absolute Cartesian position with incremental Cartesian position and you have the two groups circular incremental motion packet for RMI.

2.4.13 Packet To Add Joint Motion With Joint Representation

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_JointMotionJRep”, “SequenceID”: integerValue, “JointAngles”: { “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue1,</pre>	<pre>{“Instruction”: FRC_JointMotionJRep”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n</pre>
<p>The following items are optional:</p> <pre>“ACC” : byteValue2, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue4, “PortType” : byteValue3, “PortNumber” : shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”} \r\n</pre>	

This packet adds a joint motion instruction to the RMI_MOVE TP program. The only difference between this packet and the FRC_JointMotion packet is the new instruction has joint angle representation, instead of Cartesian representation. Without any motion option, a basic FRC_JointMotionJRep packet creates the following TP motion line:

J P[10] 2% FINE

After the controller executes this joint motion instruction, the controller sends the return packet back. The return value integerValue1 = 0 means that the motion instruction successfully executed. If integerValue1 > 0, there is an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “J7”, “J8” and “J9” keys in the JointAngle.

2.4.13.1 Packet to add two groups joint motion with joint representation instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_JointMotionJRep”, “SequenceID”: integerValue, “G1”: {“JointAngle”: “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “G2”: {“JointAngle”: “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue1,</pre>	<pre>{“Instruction”: “FRC_JointMotion”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n</pre>
<p>The following items are optional.</p> <pre>“ACC”: byteValue2, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “MROT”: stringValue3: “ON” or “OFF”, “LCBType”: stringValue4, “LCBValue”: shortValue5, “PortType”: byteValue3, “PortNumber”: shortValue5, “PortValue”: stringValue5, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”}\r\n</pre>	

This protocol is the same as FRC_JointMotion and only the difference is that the first position must be in joint angles. Please refer to section Section 2.4.9, Packet To Add Joint Motion Instruction FRC_JointMotion for detail description. Note that joint motion does not support the COORD motion option. If you specified COORD in a joint motion instruction, RMI will return an **RMIT-041** error.

2.4.14 Packet To Add Joint Incremental Motion With Joint Representation

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_JointRelativeJRep”, “SequenceID”: integerValue, “JointAngles”: { “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “SpeedType”: stringValue1, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue1,</pre>	<pre>{“Instruction”: “FRC_JointRelativeJRep”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n</pre>
<p>The following items are optional:</p> <pre>“ACC” : byteValue2, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue4, “PortType” : byteValue3, “PortNumber” : shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “NoBlend” : “ON”} \r\n</pre>	

This packet adds an incremental joint motion instruction to the RMI_MOVE TP program. The only difference between this packet and the FRC_JointRelative packet is that the new instruction has joint angle representation, instead of Cartesian representation. Without any motion option, a basic FRC_JointRelativeJRep packet creates the following TP motion line:

J P[10] 2% FINE INC

After the controller executes this incremental joint motion instruction, the controller sends the return packet back. If the return value integerValue1 = 0 it means that the motion instruction was successfully executed. If integerValue1 > 0, there was an error in executing the motion and the integerValue1 is the error code.

NOTE

If your controller does not have extended axes, you can ignore the “J7”, “J8” and “J9” keys in the JointAngle.

2.4.14.1 Packet to add two groups joint incremental motion with joint representation instruction

The two groups FRC_JointRelativeJRep instruction shares the same protocol as the two groups FRC_JointMotionJRep instruction defined in Section 2.4.13.1, Packet to add two groups joint motion with joint representation instruction. Please use the same protocol defined in Section 2.4.13.1, Packet to add two groups joint motion with joint representation instruction, change the value of the Instruction key from FRC_JointMotionJRep to FRC_JointRelativeJRep , replace the absolute joint angles with incremental joint angles and you have the two groups joint incremental motion with joint representation packet for RMI.

2.4.15 Packet To Add Linear Motion With JointRepresentation

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{“Instruction”: “FRC_LinearMotionJRep”, “SequenceID”: integerValue, “JointAngles”: { “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}, “Speed”: shortValue1, “TermType”: stringValue2, “TermValue”: byteValue1,</pre>	<pre>{“Instruction”: “FRC_LinearMotionJRep”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n</pre>
<p>The following items are optional:</p> <pre>“ACC” : byteValue2, “OffsetPRNumber”: shortValue2, “VisionPRNumber”: shortValue3, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue4, “PortType” : byteValue3, “PortNumber” : shortValue5, “PortValue”: stringValue4, “ToolOffsetPRNumber”: shortValue6, “ALIM”:integerValue1, “ALIMREG”:shortValue7, “NoBlend” : “ON”}\r\n</pre>	

This packet adds a linear motion instruction to the RMI_MOVE TP program. The only difference between this packet and the FRC_LinearMotion packet is that the new instruction has joint angle representation instead of Cartesian representation. Without any motion option, a basic FRC_LinearMotionJRep packet creates a TP motion line as shown below. Assuming the stringValue1 = “mmsec”, shortValue1 = 150 with “CNT” termination type and byteValue1 = 100, then the following instruction is added to the TP program:

L P[10] 150mm/sec CNT100

After the controller executes this instruction, the controller sends the return packet back. If integerValue1 = 0 it means that the motion instruction was successfully executed. If integerValue1 > 0, it means that there was an error in executing the motion and the integerValue1 is the error code.

2.4.15.1 Packet to add two groups linear motion with joint representation instruction

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_JointMotionJRep", "SID": integerValue, "G1": { "JointAngles": { "J1": floatJ1, "J2": floatJ2, "J3": floatJ3, "J4": floatJ4, "J5": floatJ5, "J6": floatJ6, "J7": floatJ7, "J8": floatJ8, "J9": floatJ9 } }, "G2": { "JointAngles": { "J1": floatJ1, "J2": floatJ2, "J3": floatJ3, "J4": floatJ4, "J5": floatJ5, "J6": floatJ6, "J7": floatJ7, "J8": floatJ8, "J9": floatJ9 } }, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue1, </pre>	<pre>{ "Instruction": "FRC_JointMotionJRep", "ErrorID": integerValue1, "SequenceID": integerValue2 }</pre>
<p>The following items are optional.</p> <pre> "ACC" : byteValue2, "OffsetPRNumber": shortValue2, "VisionPRNumber": shortValue3, "WristJoint": stringValue3: "ON" or "OFF", "MROT": stringValue4: "ON" or "OFF", "COORD": stringValue5: "ON" or "OFF", "LCBType" : stringValue6, "LCBValue" : shortValue4, "PortType" : byteValue3, "PortNumber" : shortValue5, "PortValue": stringValue7, "ToolOffsetPRNumber": shortValue6, "NoBlend": "ON" </pre>	

2.4.16 Packet To Add Linear Motion With JointRepresentation

Remote Device Instruction Packet	Robot Controller Return Packet
<pre>{ "Instruction": "FRC_LinearRelativeJRep", "SequenceID": integerValue, "JointAngles": { "J1": floatJ1, "J2": floatJ2, "J3": floatJ3, "J4": floatJ4, "J5": floatJ5, "J6": floatJ6, "J7": floatJ7, "J8": floatJ8, "J9": floatJ9}, "SpeedType": stringValue1, "Speed": shortValue1, "TermType": stringValue2, "TermValue": byteValue1, </pre>	<pre>{ "Instruction": "FRC_LinearRelativeJRep", "ErrorID": integerValue1, "SequenceID": integerValue2} \r\n</pre>
<p>The following items are optional:</p> <pre> "ACC" : byteValue2, "OffsetPRNumber": shortValue2, "VisionPRNumber": shortValue3, "WristJoint": "ON", "MROT": "ON", "LCBType" : stringValue3, "LCBValue" : shortValue4, "PortType" : byteValue3, "PortNumber" : shortValue5, "PortValue": stringValue4, "ToolOffsetPRNumber": shortValue6, "ALIM": integerValue1, "ALIMREG": shortValue7, "NoBlend" : "ON"} \r\n </pre>	

The FRC_LinearRelativeJRep packet is the same as FRC_LinearRelative, except that the position data is in joint representation. For a simple FRC_LinearRelativeJRep packet, the controller creates a TP line as follows:

L P[10] 150mm/sec CNT100 INC

After the controller executes this incremental linear motion instruction, the controller sends the return packet back. If integerValue1 = 0 it means that the motion instruction successfully executed. If integerValue1 > 0, there was an error in executing the motion and the integerValue1 is the error code.

2.4.16.1 Packet to add two groups circular incremental motion instruction

The two groups FRC_LinearRelativeJRep instruction shares the same protocol as the two groups FRC_LinearMotionJRep instruction defined in Section 2.4.15.1, Packet to add two groups linear motion with joint representation instruction. Please use the same protocol defined in Section 2.4.15.1, Packet to

add two groups linear motion with joint representation instruction, change the value of the **Instruction** key from **FRC_LinearMotionJRep** to **FRC_LinearRelativeJRep**, replace the absolute Cartesian position with incremental position and you have the two groups linear incremental motion with joint representation packet for RMI.

2.4.17 Unknown Packet Handling

When the RMI software receives a packet that it cannot interpret, that is, this packet does not have the format specified in the above sections, the RMI software will return a packet with an unknown string error:

Remote Device Instruction Packet	Robot Controller Return Packet
RMI cannot interpret the string	{“Command”: “Unknown”, “ErrorID”: “integerValue”} \r\n

Since RMI cannot interpret the incoming string, it returns a special Unknown command packet back to the remote device.

If RMI can partially recognize the incoming instruction packet, it will return the sequence ID as well. The format is shown below.

Remote Device Instruction Packet	Robot Controller Return Packet
RMI cannot interpret the JSON string	{“Command”: “Unknown”, “SequenceID”: “integerValue1”, “ErrorID”: “integerValue2”} \r\n

You can check the ErrorID and find out which field is incorrect or missing for this instruction. For example, if the ErrorID = 2556959 (**RMIT-031**), it means the motion speed value is invalid.

NOTE

Please see the *Error Code Manual* for a description of each of the Remote Motion Interface error codes.

2.5 POSITION RECORDING

There are two methods of recording position data on the robot controller to use this data in your application. The first method uses position registers and the second method uses the **RMI Position Record** menu.

2.5.1 Using Position Registers

You can record the robot position to position registers without connecting the remote device to the robot controller. After you have taught all of the points in the position registers, you can connect the remote device to the robot controller (**FRC_Connect**) and use the **FRC_ReadPositionRegister** packet to transfer all of the recorded position register data to the remote device.

2.5.2 Using RMI Position Record Menu

You can also choose **RMI Position Record** under **UTILITIES**. To use this menu, you should have the remote device already connected to the robot controller.

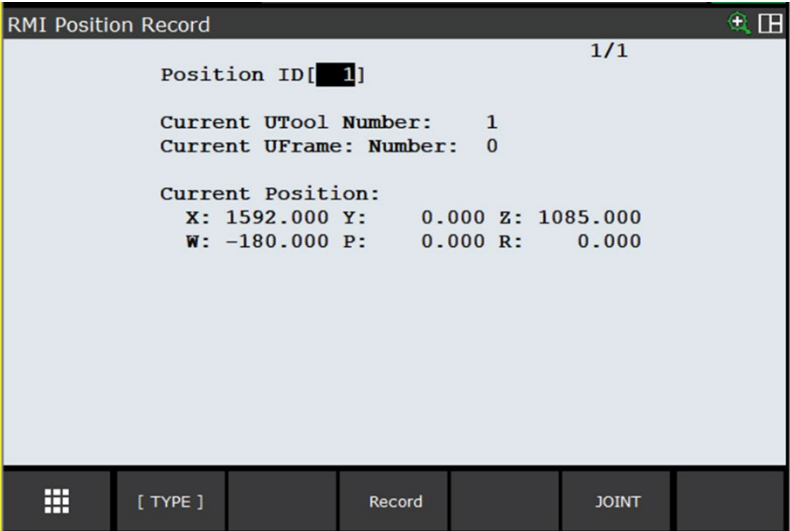


Fig. 2.5.2 (a) RMI Position Record

Pressing the **Record** key, the robot controller sends either one of the following packets to the remote device:

Robot Controller Sent Packet	Remote Device Return Packet
<pre>{ "Command": "FRC_Record", "PositionID": UShortValue, "Representation": Cartesian, "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX, "Y": floatY, "Z": floatZ, "W": floatW, "P": floatP, "R": floatR, "Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}} \r\n</pre>	None

Robot Controller Sent Packet	Remote Device Return Packet
<pre>{"Command": "FRC_Record", "PositionID": UShortValue, "Representation": "Joint", "JointAngle": { "J1": floatJ1, "J2": floatJ2, "J3": floatJ3, "J4": floatJ4, "J5": floatJ5, "J6": floatJ6, "J7": floatJ7, "J8": floatJ8, "J9": floatJ9}} \r\n</pre>	None

For a multiple group controller, the menu changes as shown in Figure 2.5.2 (b).

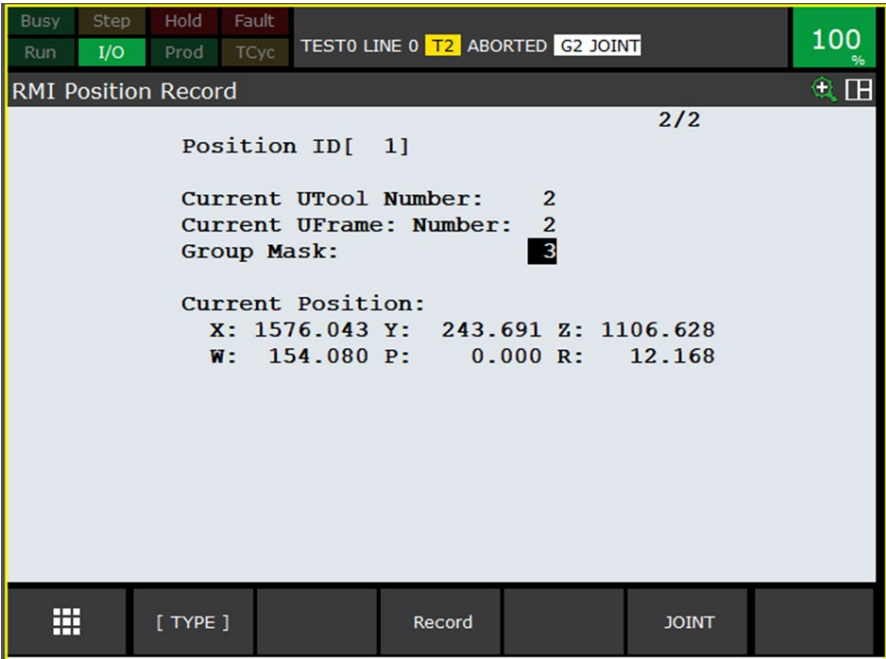


Fig. 2.5.2 (b) RMI Position Record for Multiple Groups

When the **Record** button is pressed, RMI will send the selected group position data back to remote device based on the group mask field. Please see section Section 2.3.1, Packet To Initialize Remote Motion for detail description of the Group Mask.

The Current position display is based on the current jogging group selection. For the example above, the Current Position displays group 2 current position. You can use the **FCTN** key **CHANGE GROUP** to toggle the display between different groups.

The Group Mask field is not available on a single group system.

For the above example, pressing the **Record** key, the robot controller sends the one of the following packets to the remote device:

Robot Controller Sent Packet	Remotel Device Return Packet
{ "Command": "FRC_Record", "PositionID": UShortValue, "Representation": Cartesian, "G1": { "Configuration": { "UToolNumber": byteValue1, "UFrameNumber": byteValue2, "Front": byteValue3, "Up": byteValue4, "Left": byteValue5, "Flip": byteValue6, "Turn4": byteValue7, "Turn5": byteValue8, "Turn6": byteValue9}, "Position": { "X": floatX1, "Y": floatY1, "Z": floatZ1, "W": floatW1, "P": floatP, "R1": floatR1,	None

Robot Controller Sent Packet	Remotel Device Return Packet
"Ext1": floatE1, "Ext2": floatE2, "Ext3": floatE3}}, "G2": { "Configuration": { "UToolNumber": byteValue10, "UFrameNumber": byteValue11, "Front": byteValue12, "Up": byteValue13, "Left": byteValue14, "Flip": byteValue15, "Turn4": byteValue16, "Turn5": byteValue17, "Turn6": byteValue18}, "Position": { "X": floatX2, "Y": floatY2, "Z": floatZ2, "W": floatW2, "P": floatP2, "R": floatR2, "Ext1": floatE4, "Ext2": floatE5, "Ext3": floatE6}}}\r\n	

Robot Controller Sent Packet	Remote Device Return Packet
{"Command": "FRC_Record", "PositionID": UShortValue, "Representation": "Joint", "G1": { "JointAngles": { "J1": floatJ1, "J2": floatJ2, "J3": floatJ3, "J4": floatJ4, "J5": floatJ5, "J6": floatJ6, "J7": floatJ7, "J8": floatJ8, "J9": floatJ9}}, "G2": { "JointAngles": { "J1": floatJ10, "J2": floatJ11, "J3": floatJ12, "J4": floatJ13, "J5": floatJ14, "J6": floatJ15, "J7": floatJ16, "J8": floatJ17, "J9": floatJ18}}}\r\n	None

3 REMOTE DEVICE CONTROL FLOW

3.1 OVERVIEW OF REMOTE DEVICE CONTROLFLOW

The following flow chart shows a typical Remote Motion session between a remote device and the robot controller.

NOTE

When the remote device sends the `FRC_Connect` packet, the packet should be sent to a fixed port number (16001) on the controller. When the robot controller sends the returned packet back and the connection is accepted, the return packet will include a new port number. All packets after the `FRC_Connect` should use the new port number to communicate with the robot's Remote Motion Interface.

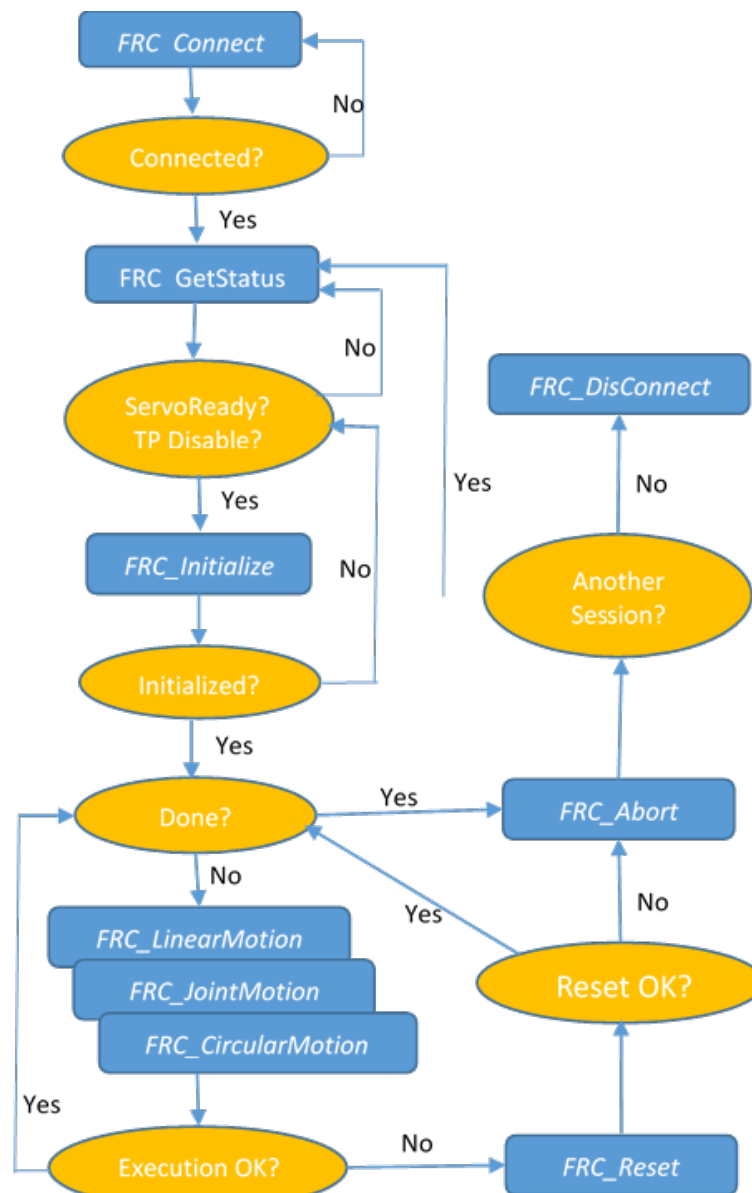


Fig. 3.1 (a) Remote Device Control Flow Chart

3.2 INSTRUCTION BUFFER HANDLING

The Remote Motion Interface has a limited buffer for the incoming Teach Pendant instructions. Currently, the limit is set to 8. When the buffer is full, the RMI returns an error if the remote device sends an extra TP instruction. Therefore, when designing the remote device interface with the RMI, it is advised to have a hand shaking mechanism that does the following:

1. After the `FRC_Initialize` packet, send several (up to 8) TP instructions to fill up the RMI instruction buffer.

NOTE

For each of the 8 instructions, please wait at least 2 milliseconds before sending the next instruction. This is due to TCP/IP packs several RMI packets together in one TCP/IP packet if these RMI packets arrive around the same time. It is possible that during the packing, an RMI packet could be broken into two parts and carried by two TCP/IP packets. RMI will return an error in this case.

2. After the buffer is full, only send the next TP instruction when a previously sent TP instruction is returned by the RMI.

3.3 HOLD STATE ERROR HANDLING

The controller goes to the HOLD state when the RMI receives an invalid motion instruction. For example, if the controller does not have R640 (MROT) software loaded and it receives a motion instruction with the MROT option, the controller will return an error packet back to the remote device and the RMI goes into the HOLD state. In the HOLD state, the controller is still executing the RMI TP program, but it blocks any new instructions to the TP program until it receives a `FRC_Reset` command. This allows the remote device to react to the error without stopping the execution of the TP program. Once the remote device corrects the error, it can set `FRC_Reset` and then continue to send the correct motion instructions. All instructions sent during the HOLD state are ignored and returned back to the sender with an error code.

3.4 JUMP AND SKIP INSTRUCTIONS

The running `RMI_MOVE` TP program cannot execute a JUMP or SKIP instruction since this TP program is dynamically updated during its execution. Therefore, you have to manage the SKIP or JUMP instruction in your application program by calling `FRC_Abort` to terminate the current `RMI_MOVE` TP program, and use `FRC_Initialize` to create the program again. When the new TP program is created, you can add the required TP instructions back to the new `RMI_MOVE` TP program.

3.5 ERROR HANDLING

There are two type of errors that can happen during the remote motion execution:

- *Recoverable*: For some warning errors, you may able reset the error by sending the `FRC_Reset` packet. If the return packet shows that the reset is successful, use the `FRC_Continue` packet to continue the current teach pendent program execution. Note that after sending the `FRC_Reset` command, it takes time for the robot controller to reset its error. Please wait for the return of the `FRC_Reset` packet before sending the `FRC_Continue` packet. Otherwise, the `FRC_Continue` command may fail.
- *Non-Recoverable*: For any non-warning error, such as a limit error, you have to do the following to recover:

- Abort the current RMI_MOVE TP program to flush out all of the instructions that are already sent to the controller using the FRC_Abort packet.
- Go to the controller to clear the error.
- Send a FRC_Initialize packet to start the RMI_MOVE TP program again following a new set of instructions to continue your process.

3.6 VISION PROCESS

In order to use FANUC vision, you can run your vision process as a background task. The Remote Motion Interface can send a DOUT (FRC_WriteDOUT) to start the vision process and use FRC_WaitDIN to wait for the vision process to complete before the TP program executes its next instruction. If the vision process is synchronous with the RMI_MOVE program, you can also use FRC_CALL to call the vision process.

4 DATA LOGGING

In order to facilitate debugging an RMI session, a data log is created to log all the interaction between the controller and the remote device. This log can be viewed on the TP by pressing **MENU > NEXT > BROWSER > RMI_LOG**.

Like the PLC Motion Interface, the Remote Motion Interface creates a data log for all incoming instruction/command packets with a time stamp and some description of the packets.

```
**** RMI Execution Log ****
05-DEC-19 10:25 :12 FRC_Initialize @ 55567
05-DEC-19 10:25 :12 FRC_Initialize OK return @ 55675
05-DEC-19 10:25 :12 FRC_SetUFrame ID: 1 @ 55681
05-DEC-19 10:25 :12 Line 3 FRC_SetUFrame ID: 1 @ 55681
05-DEC-19 10:25 :12 Line 3 is executing at 55689
05-DEC-19 10:25 :12 Line 3 ID: 1, return @ 55689
05-DEC-19 10:25 :13 FRC_SetUTool ID: 2 @ 55691
05-DEC-19 10:25 :13 Line 4 FRC_SetUTool ID: 2 @ 55691
05-DEC-19 10:25 :13 Line 4 is executing at 55693
05-DEC-19 10:25 :13 Line 4 ID: 2, return @ 55693
05-DEC-19 10:25 :13 FRC_JointMotion ID: 3 @ 55693
05-DEC-19 10:25 :13 Line 5 FRC_JointMotion ID: 3 @ 55693
05-DEC-19 10:25 :13 Line 5 is executing at 55693
05-DEC-19 10:25 :13 FRC_LinearMotion ID: 4 @ 55693
05-DEC-19 10:25 :13 Line 6 FRC_LinearMotion ID: 4 @ 55693
05-DEC-19 10:25 :13 FRC_LinearRelative ID: 5 @ 55693
05-DEC-19 10:25 :13 Line 7 FRC_LinearRelative ID: 5 @ 55693
05-DEC-19 10:25 :13 FRC_JointRelative ID: 6 @ 55693
05-DEC-19 10:25 :13 Line 8 FRC_JointRelative ID: 6 @ 55693
05-DEC-19 10:25 :13 FRC_JointRelativeJRep ID: 7 @ 55693
-----
```

Fig. 4 (a) Example Log File

From Figure 4, you can see the controller receives an `FRC_Initialize` packet from the remote device at the controller's tick count 55567. The controller returns the `FRC_Initialize` 8 ticks later and the TP program is initialized successfully.

The remote device immediately sends the `FRC_SetUFrame` instruction and the controller executes this instruction 8 tick later.

5 ASCII STRING INSTRUCTION PACKETS

A Teach Pendant program can be printed out as a LS file using the PRINT function key in the SELECT menu. An LS file can have five sections:

1. /PROG (program name)
2. ATTR (program attributes)
3. /APPL(Application)
4. /MN (program instructions)
5. /POS (position data)

For V9.30P/13 (7DF3/13) and later software and V9.40P/08 (7DF5/08) and later software, the remote application can send ASCII strings from the LS file's /MN section to RMI and RMI will add the equivalent binary TP instruction to the running RMI_MOVE TP program. In the following sections, we will use ASCII string interchangeably as the *ASCII string in the /MN section of an LS file*.

The new ASCII string support adds lots of new TP instructions to the RMI. However, not all TP instruction are supported due to the sequential execution nature of the RMI TP program. That is, all TP instructions in the RMI_MOVE program are executed sequentially, and any instruction that changes the order of the execution is not allowed. Some of the invalid TP instructions are listed in a later section.

Another issue with the ASCII string support is that some motion instructions, such as ARC motion instruction, require multiple motion lines in the TP Program when the first motion instruction is executed. If there are insufficient ARC motion lines in the TP program when the first ARC motion is executed, the controller stops the program execution with an **INTP-609** error. Therefore, when sending ARC motion instructions, users have to make sure the program will not execute the first ARC motion until at least three ARC motion instructions are written into the TP program.

5.1 ASCII STRING PACKET HANDSHAKING

When loading an LS file into the controller, the LS file goes through an ASCII to binary conversion that converts the ASCII file into a binary Teach Pendant program. RMI uses the same process for processing the ASCII string sent through the RMI connection. Since the conversion process runs outside of the RMI, RMI will have to wait for the conversion process to complete before it can process the next ASCII string packet. To make sure the remote application will wait for the ASCII to binary conversion, the handshaking between the RMI and the RMI application is changed from Figure 5.1 (a) to Figure 5.1 (b).

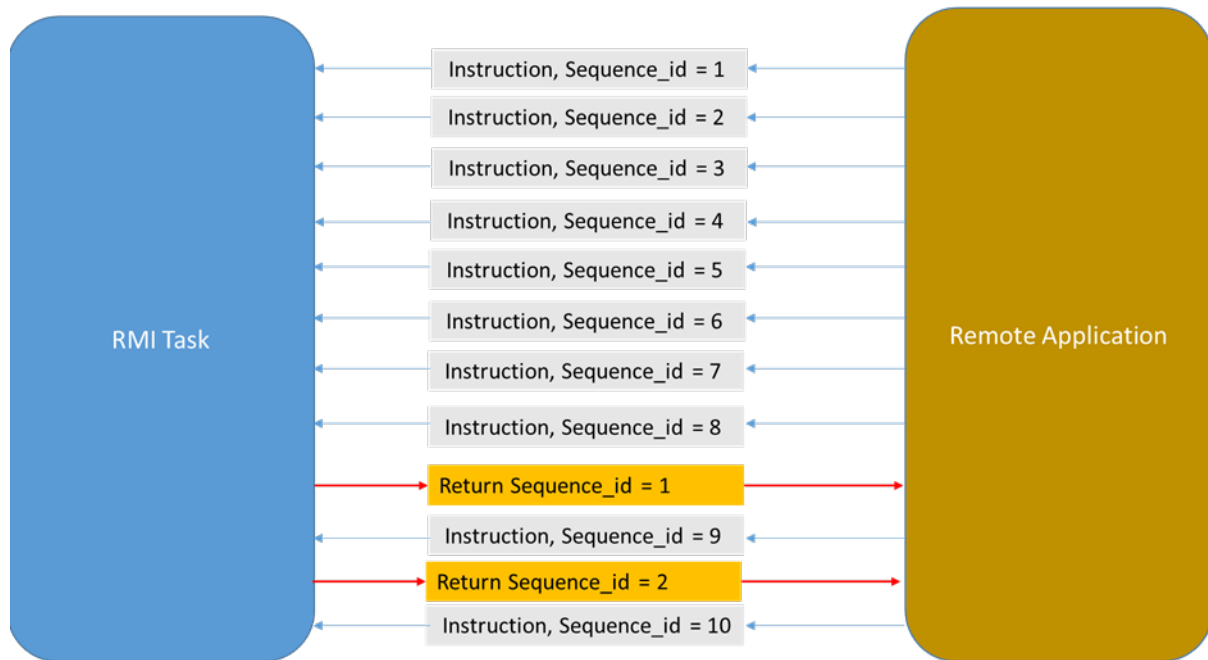


Figure 5.1 (a) Current RMI and RMI Application Handshaking Sequence

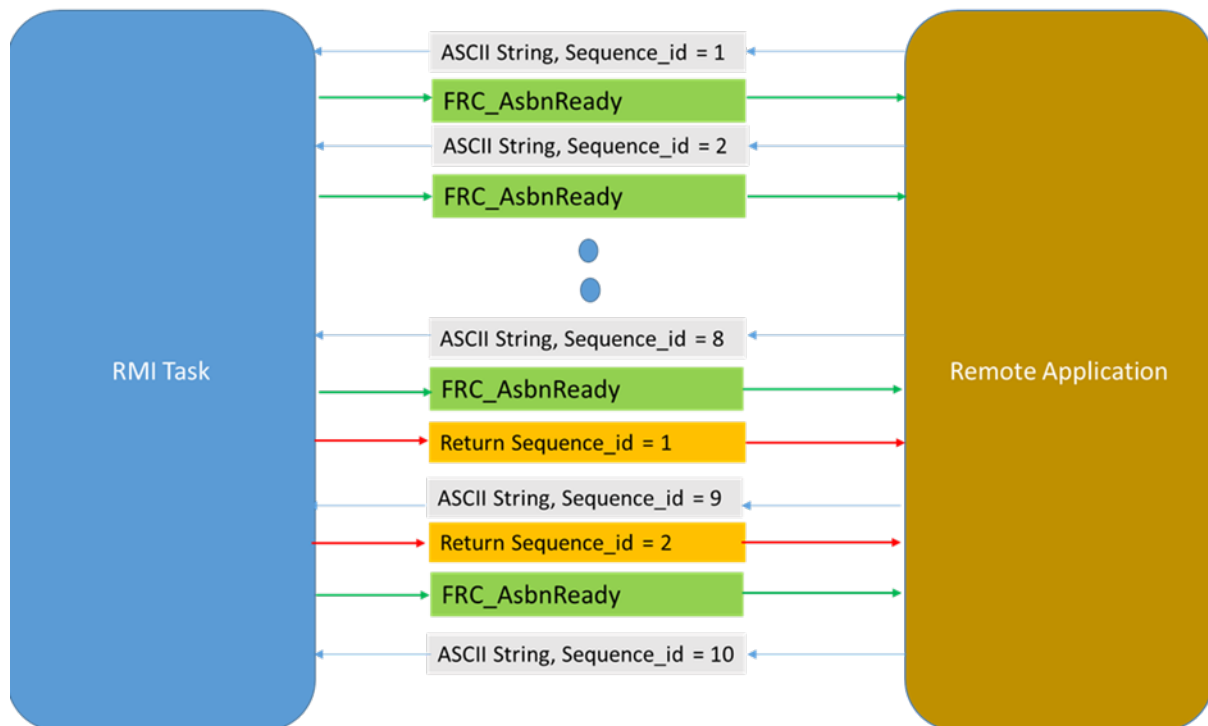


Figure 5.1 (b) RMI with ASCII String and RMI Application Handshaking

5.2 SYSTEM VARIABLE SETTING

To enable the ASCII string support in RMI, set the system variable `$rmi_cfg.$asbn_enb` to TRUE and recycle power. Since the ASCII string packet and the JSON based instruction packet have different handshaking, you cannot mix the ASCII string packet with the JSON based instruction packet. By default, this system variable is set to FALSE so RMI is compatible with the current version of the RMI application that uses JSON based instruction packets. If a user sends an ASCII string packet to RMI when the system variable is set to FALSE, RMI returns **RMIT-050 Invalid ASCII String Packet**. If

the *\$rmi_cfg.\$asbn_enb* is set to TRUE and the user sends a JSON instruction packet, RMI returns **RMIT-049 Invalid Instruction Packet**.

The ASCII string is very flexible. However, the ASCII string packet goes through the ASCII to binary conversion process so it takes a longer time to convert the ASCII string to a TP instruction. Whereas the RMI converts the JSON instruction to a TP instruction directly, the ASCII conversion requires an additional 4 to 16 milliseconds, depending on the complexity of the ASCII string. If your RMI application is a time critical one, please use the JSON instruction packet instead of the ASCII string packet by setting *\$rmi_cfg.\$asbn_enb* = FALSE.

When using the ASCII string packet, the RMI application needs to consider that it is possible that both the **FRC_AsbnReady** packet and a return packet will be sent out by RMI around the same time. In this case, the TCP/IP will combine both packets into one TCP/IP packet and the application will only receive one TCP/IP packet instead of two. The RMI Application needs to handle the TCP/IP packet with multiple RMI packets in order to process these two RMI packets individually. It does this by using “\r\n” as a delimiter to separate these RMI packets.

5.3 ASCII STRING INSTRUCTION PACKET

5.3.1 ASCII String Packet For Single Group Controller

Remote Device ASCII Packet	Robot Controller Return Packet
{“ASCII”: “ASCII String”, “SequenceID”: integerValue,	{“ASCII”: “ASCII String”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n
See note ¹ below explaining Option 1; no position data.	
The following items are optional. See note ² below explaining Option 2. “Configuration”: { “UToolNumber”: byteValue1, “UFrameNumber”: byteValue2, “Front”: byteValue3, “Up”: byteValue4, “Left”: byteValue5, “Flip”: byteValue6, “Turn4”: byteValue7, “Turn5”: byteValue8, “Turn6”: byteValue9 }, “Position”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: floatR, “Ext1”: floatE1, “Ext2”: floatE2,	

Remote Device ASCII Packet	Robot Controller Return Packet
<p>“Ext3”: floatE3}</p> <p>The following items are optional.</p> <p>See note ³ below explaining Option 3.</p> <p>“Configuration”: {</p> <p>“UToolNumber”: byteValue1,</p> <p>“UFrameNumber”: byteValue2,</p> <p>“Front”: byteValue3,</p> <p>“Up”: byteValue4,</p> <p>“Left”: byteValue5,</p> <p>“Flip”: byteValue6,</p> <p>“Turn4”: byteValue7,</p> <p>“Turn5”: byteValue8,</p> <p>“Turn6”: byteValue9},</p> <p>“Position”: {</p> <p>“X”: floatX, “Y”: floatY,</p> <p>“Z”: floatZ, “W”: floatW,</p> <p>“P”: floatP, “R”: floatR ,</p> <p>“Ext1”: floatE1, “Ext2”: floatE2,</p> <p>“Ext3”: floatE3},</p> <p>“ViaConfiguration”: {</p> <p>“UtoolNumber”: byteValue10,</p> <p>“UFrameNumber”: byteValue11,</p> <p>“Front”: byteValue12,</p> <p>“Up”: “byteValue13” ,</p> <p>“Left”: byteValue14,</p> <p>“Flip”: byteValue15,</p> <p>“Turn4”: byteValue16,</p> <p>“Turn5”: byteValue17,</p> <p>“Turn6”: byteValue18 },</p> <p>“ViaPosition”: {</p> <p>“X”: floatX, “Y”: floatY,</p> <p>“Z”: floatZ, “W”: floatW,</p> <p>“P”: floatP, “R”: floatR,</p> <p>“Ext1”: floatE1, “Ext2”: floatE2,</p> <p>“Ext3”: floatE3}</p>	

Remote Device ASCII Packet	Robot Controller Return Packet
<p>The following items are optional.</p> <p>See note 4 below explaining Option 4.</p> <pre> “JointAngle”: { “J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9} } \r\n </pre>	

NOTE

For a single group controller, the ASCII key can have 4 options where option 2, 3 and 4 are position data used in a motion instruction:

- ¹ Option 1: The ASCII instruction does not have a position data associated with this instruction. For example, the packet from the remote device can be {“ASCII” : “PAYLOAD[2] ; “} \r\n
- ² Option 2: The ASCII instruction is a motion instruction that includes Cartesian position data.
- ³ Option 3: the ASCII instruction is a motion instruction that includes a Cartesian via position and destination position data.
- ⁴ Option 4: The ASBN instruction is a motion instruction that includes position data in joint representation.

The “ASCII String” has to be a valid LS instruction line in the \MN section of an LS file, including the ‘;’ at the end of the line. The maximum length of the “ASCII String” is 127 bytes.

There are several limitations on the “ASCII string” value:

- The RMI does not preserve the position ID. For example, if the remote device sends the following ASCII string: {“ASCII” : “L P[5] 200 mm/sec FINE ;” } \r\n, the RMI will not use 5 in the position ID in the RMI_MOVE TP program since RMI assigns an internal position ID number for each new motion instruction.
- All motion instructions having a position ID should provide position data within the ASCII packet using option 2, 3 or 4. If there is no position data, an error will be posted.
- If the user sends multiple position data with a motion instruction, RMI will return an error.
- If user sends option 3 with linear or joint motion, the via position is ignored.
- If a motion instruction uses a position register, RMI preserves the position register number. However, if the position register is not initialized, a run time error will occur since RMI does not check the validity of the position register before writing the TP instruction to the RMI_MOVE program.
- RMI does not check the validity of motion options. For example, if the “ASCII String” is a motion instruction with the MROT option, RMI does not check whether the MROT motion option is loaded or not before writing the TP binary code to the RMI_MOVE program. If the MROT motion option is not loaded, the execution of MROT will cause a run time program execution error.
- RMI does not check motion option conflicts. For example, an RMI application can sent a multiple group system with coordinated motion option with {“ASCII” : “J P[10] 10% CNT100 COORD ;”} \r\n. RMI will write the TP instruction in the RMI_MOVE program, however since joint motion is illegal for coordinated motion option, this packet will cause a run time program execution error.

- RMI cannot provide any error information for an illegal “ASCII String” string. Instead, the RMI returns ErrorID = 1441794 (ASBN-002). Users have to go to the controller’s alarm page to find out the issue with the ASCII string.

5.3.2 ASCII String Packet For Two Group MotionInstruction

RMI can support a TP motion instruction for two groups in a multiple group controller.

Remote Device ASBN Packet	Robot Controller Return Packet
{“ASCII”: “ASCII String”, “SequenceID”: integerValue,	{“ASCII”: “ASCII String”, “ErrorID”: integerValue1, “SequenceID”: integerValue2} \r\n
The following items are Optional: “G1” : {Option 2 or Option 3 or Option 4}, “G2” : {Option 2 or Option 3 or Option 4}	
} \r\n	

NOTE

“G1” and “G2” should be replaced by other valid group numbers as specified in the FRC_Initialize packet.

RMI does not check whether the ASCII instruction has valid position data or not. If there is not enough position data provided by the remote application, it will cause a run time program execution error.

5.3.3 Communication Packet

Robot Controller ASCII to Binary Ready Packet	Remote Device Return Packet
{“Communication”: “FRC_AsbnReady” } \r\n	None

When *\$rmi_cfg.sasbn_enb* = TRUE, RMI will send this packet to the remote device to indicate the ASCII to binary conversion has completed successfully and the TP instruction is written to the RMI_MOVE TP program. Otherwise, RMI will return a packet with a non-zero errorID. For the first eight ASCII String instruction packets, the remote application can send the next ASCII packet to the RMI after it received the FRC_AsbnReady packet. After the instruction buffer is filled, the remote application has to wait for both the FRC_AsbnReady packet and a returned instruction packet before it can send the next ASCII packet.

5.3.4 Modified FRC_Initialize Command

To support multiple tools, the FRC_Initialize packet adds two new optional fields as shown in the table below.

Remote Device Command Packet	Robot Controller Return Packet
<pre>{“Command”: “FRC_Initialize”, The following items are optional: “GroupMask”: unsignedByteValue, “Application”: StringValue , “Equipment”: ByteValue} \r\n</pre>	<pre>{“Command”: “FRC_Initialize”, “ErrorID”: integerValue , “GroupMask”: unsignedByteValue} \r\n</pre>

The StringValue has the following valid options:

- “Handling”
- “Arc”
- “Spot”
- “Dispense”

If the “Application” key is not included in the FRC_Initialize packet, RMI creates the RMI_MOVE program without the application tool specification in its program header. Otherwise, RMI creates the RMI_MOVE TP program header accordingly. However, if the controller’s application tool setting is not compatible with StringValue, RMI will return an error.

RMI also support multiple equipment in the RMI_MOVE program header for ARC and Spot application. The ByteValue is the equipment number for RMI to write to the RMI_MOVE program header. RMI does not check the validity of the equipment number.

NOTE

If the controller is loaded with the handling tool or dispense tool, the equipment number is ignored.

For controller loaded with singularity avoidance option, RMI will automatically disable the singularity avoidance in RMI_MOVE program header. Since the remote RMI application issues the motion instructions, the remote RMI application should plan motion around singularity by itself.

For controller loaded with tracking option, RMI will automatically disable the tracking option in RMI_MOVE program header since RMI does not support tracking.

For HandlingTool application, by default, the RMI_MOVE program is created as JOB program type. For Spot, Arc and Dispense, the RMI_MOVE program is created as NONE program type.

When *\$rmi_cfg.\$asbn_enb* = FALSE, the “Application” and “Equipment” keys are ignored.

5.4 LIST OF INVALID ASCII STRING INSTRUCTIONS

RMI does not allow instructions that change the order of execution of the RMI_MOVE program. The following is a list of known instructions that RMI does not support and will return an ErrorID = 2556975 (**RMIT-047**).

- Jump/Label instructions.
- Run instruction.
- IF/Select instruction.
- Skip instruction.
- FOR/ENDFOR instruction.

APPENDIX

A RMI ERRORID REFERENCE TABLE

The following table provides a quick reference for the return packet's ErrorID.

ErrorID	RMI Error	Error Message
2556929	RMIT-001	Internal System Error.
2556930	RMIT-002	Invalid UTool Number.
2556931	RMIT-003	Invalid UFrame Number.
2556932	RMIT-004	Invalid Position Register.
2556933	RMIT-005	Invalid Speed Override.
2556934	RMIT-006	Cannot Execute TP program.
2556935	RMIT-007	Controller Servo is Off.
2556936	RMIT-008	Cannot Execute TP program.
2556937	RMIT-009	RMI is Not Running.
2556938	RMIT-010	TP Program is Not Paused.
2556939	RMIT-011	Cannot Resume TP Program.
2556940	RMIT-012	Cannot Reset Controller.
2556941	RMIT-013	Invalid RMI Command.
2556942	RMIT-014	RMI Command Fail.
2556943	RMIT-015	Invalid Controller State.
2556944	RMIT-016	Please Cycle Power.
2556945	RMIT-017	Invalid Payload Schedule.
2556946	RMIT-018	Invalid Motion Option.
2556947	RMIT-019	Invalid Vision Register.
2556948	RMIT-020	Invalid RMI Instruction.
2556949	RMIT-021	Invalid Value.
2556950	RMIT-022	Invalid Text String
2556951	RMIT-023	Invalid Position Data
2556952	RMIT-024	RMI is In HOLD State
2556953	RMIT-025	Remote Device Disconnected.
2556954	RMIT-026	Robot is Already Connected.
2556955	RMIT-027	Wait for Command Done.
2556956	RMIT-028	Wait for Instruction Done.
2556957	RMIT-029	Invalid sequence ID number.
2556958	RMIT-030	Invalid Speed Type.
2556959	RMIT-031	Invalid Speed Value.
2556960	RMIT-032	Invalid Term Type.
2556961	RMIT-033	Invalid Term Value.
2556962	RMIT-034	Invalid LCB Port Type.
2556963	RMIT-035	Invalid ACC Value.
2556964	RMIT-036	Invalid Destination Position
2556965	RMIT-037	Invalid VIA Position.
2556966	RMIT-038	Invalid Port Number.
2556967	RMIT-039	Invalid Group Number

ErrorID	RMI Error	Error Message
2556968	RMIT-040	Invalid Group Mask
2556969	RMIT-041	Joint motion with COORD
2556970	RMIT-042	Incremental motn with COORD
2556971	RMIT-043	Robot in Single Step Mode
2556972	RMIT-044	Invalid Position Data Type
2556973	RMIT-045	Ready for ASCII Packet
2556974	RMIT-046	ASCII Conversion Failed
2556975	RMIT-047	Invalid ASCII Instruction
2556976	RMIT-048	Invalid Number of Groups
2556977	RMIT-049	Invalid Instruction packet
2556978	RMIT-050	Invalid ASCII String packet
2556979	RMIT-051	Invalid ASCII string size
2556980	RMIT-052	Invalid Application Tool
2556981	RMIT-053	Invalid Call Program Name

INDEX

<A>

ASCII STRING INSTRUCTION PACKET	57
ASCII STRING INSTRUCTION PACKETS	55
ASCII String Packet For Single Group Controller	57
ASCII String Packet For Two Group MotionInstruction	60
ASCII STRING PACKET HANDSHAKING.....	55

<C>

Communication Packet.....	60
COMMUNICATION PACKETS.....	3
COMPATIBILITY.....	1
Controller Disconnect Packet.....	4

<D>

DATA LOGGING.....	54
-------------------	----

<E>

ERROR HANDLING	52
----------------------	----

<H>

HARDWARE AND SOFTWARE REQUIREMENTS	1
HOLD STATE ERROR HANDLING	52
Hot Start Support.....	2

<I>

INSTRUCTION BUFFER HANDLING	52
-----------------------------------	----

<J>

JUMP AND SKIP INSTRUCTIONS.....	52
---------------------------------	----

<L>

LIMITATIONS.....	2
LIST OF INVALID ASCII STRING INSTRUCTIONS.....	61

<M>

Modified FRC_Initialize Command.....	61
--------------------------------------	----

<N>

Non-Motion Interface Support	2
Number Of Instruction Packets	2

<O>

OVERVIEW / INTRODUCTION.....	1
OVERVIEW OF REMOTE DEVICE CONTROL	
FLOW.....	51

<P>

Packet Exchange	3
Packet To Abort Remote Motion Program.....	7
Packet To Add Circular Incremental MotionInstruction.....	39
Packet To Add Circular Motion Instruction.....	35
Packet To Add Joint Incremental Motion Instruction	33
Packet To Add Joint Incremental Motion With Joint Representation.....	43
Packet To Add Joint Motion Instruction	30
Packet To Add Joint Motion With Joint Representation	41
Packet To Add Linear Incremental Motion Instruction.....	28
Packet To Add Linear Motion Instruction	21
Packet To Add Linear Motion With JointRepresentation	44,46
Packet to add two group joint motion instruction	31
Packet to add two groups circular incremental motion instruction.....	40,46
Packet to add two groups circular motion instruction	36
Packet to add two groups joint incremental motion instruction.....	34
Packet to add two groups joint incremental motion with joint representation instruction	44
Packet to add two groups joint motion with joint representation instruction	42
Packet to add two groups linear incremental motion instruction.....	29
Packet to add two groups linear motion instruction	25
Packet to add two groups linear motion with joint representation instruction	45
Packet To Add Wait Time Instruction	20
Packet To Call A Program	20
Packet To Continue Remote Motion Program	8
Packet To Get Controller Status.....	9
Packet To Get Current User/Tool Frame Number.....	14
Packet To Initialize Remote Motion	6
Packet To Pause Remote Motion Program.....	8
Packet To Read Controller Error.....	8
Packet To Read Current Robot Cartesian Position.....	13
Packet To Read Current Robot Joint Angles.....	14
Packet To Read Current Tool Center Point Speed	17
Packet To Read Input Port	12
Packet To Read Position Register Data.....	15

Packet To Read User Frame Data	10
Packet To Read User Tool Data	11
Packet to Reset Robot Controller	16
Packet To Set Payload Instruction.....	20
Packet To Set Speed Override.....	14
Packet To Set The Current UFrame-UTool Number.....	9
Packet To Set User Frame Data	11
Packet To Set User Frame Instruction.....	19
Packet To Set User Tool Data	12
Packet To Set User Tool Instruction	19
Packet To Wait For DIN Instruction	19
Packet To Write Digital Output Port.....	12
Packet To Write Position Register Data.....	16
POSITION RECORDING	47

<R>

REMOTE DEVICE CONTROL FLOW	51
REMOTE MOTION APPLICATION PROGRAM	
INTERFACE	3
REMOTE MOTION APPLICATION PROGRAM	
INTERFACE OVERVIEW	3
REMOTE MOTION INTERFACE OVERVIEW.....	1
RMI ERRORID REFERENCE TABLE	65
ROBOT COMMAND PACKETS.....	5

<S>

SAFETY PRECAUTIONS	s-1
Short Motion	2
Single Group Motion.....	2
System Fault Packet	5
SYSTEM VARIABLE SETTING	56

<T>

TEACH PENDANT PROGRAM INSTRUCTION	
PACKETS	17
Timeout Terminate Packet	5

<U>

Unknown Packet Handling.....	47
Using Position Registers	47
Using RMI Position Record Menu	47

<V>

VISION PROCESS	53
----------------------	----

REVISION RECORD

Edition	Date	Contents
02	Oct., 2022	<ul style="list-style-type: none">• Support ASCII string packets• New feature description for FRC_Call instruction and Tool_offset, PR[] motion modifier• Multiple Groups• Addition of the ALIM (Acceleration Limit) motion modifier function.
01	Mar., 2020	

B-84184EN/02

