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/01

- **Original Instructions**

Thank you very much for purchasing FANUC Robot.
Before using the Robot, be sure to read the "FANUC Robot SAFETY HANDBOOK (B-80687EN)" and understand the content.

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 must be read before using the robot.

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 “FANUC Robot SAFETY HANDBOOK (**B-80687EN)**”.

1 DEFINITION OF USER

The 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 safety fence

Maintenance technician:
- Operates the robot
- Teaches the robot inside the safety fence
- Performs maintenance (repair, adjustment, replacement)

- Operator is not allowed to work in the safety fence.
- Programmer/Teaching operator and maintenance technician is allowed to work in the safety fence. Works carried out in the safety fence include transportation, installation, teaching, adjustment, and maintenance.
- To work inside the safety fence, the person must be trained on proper robot operation.

Table 1 (a) lists the work outside the safety fence. In this table, the symbol "○" means the work allowed to be carried out by the worker.

Table 1 (a) List of work outside the fence

	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			○

In the robot operating, 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.

- Check this manual thoroughly, and keep it handy for the future reference.

TABLE OF CONTENTS

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.

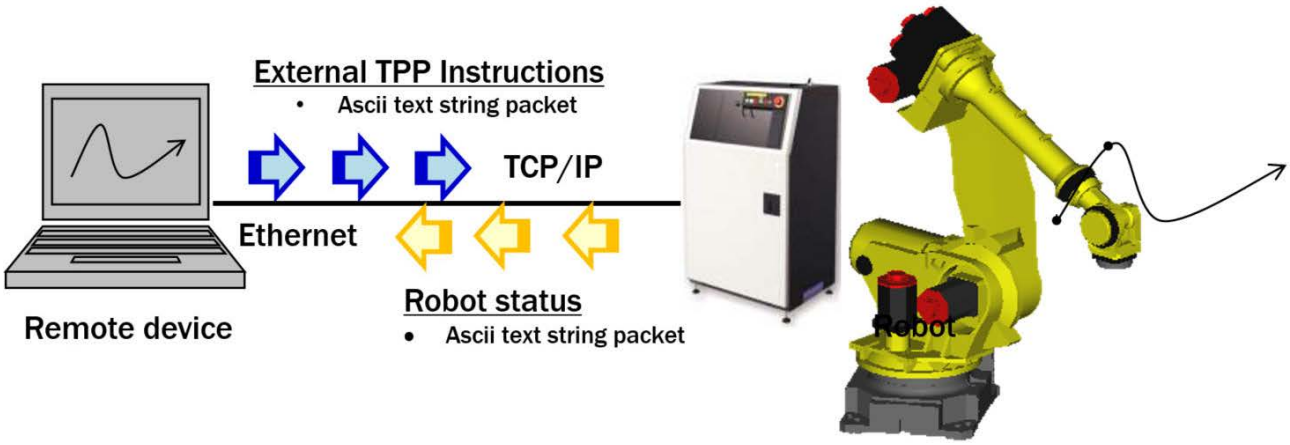

Fig. 1.1 (a) System Overview

Remote Motion Interface interacts with a remote device through TCP/IP socket messages.
The main target for this software is material handling with simple pick and place operations. This software does not support operations with very short execution times. Therefore, it cannot be used in applications that require tight motion using real-time sensor feedback.

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

RMI only supports a single 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 and posts an 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 sent to the robot controller at one time. When the capacity to accept instruction 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.
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 initialize packet, send several (up to 8) TP instruction to fill up the RMI instruction buffer.
2. After the capacity of instruction is full, only send the next TP instruction when a previously sent TP instruction is returned by the RMI.

1.4.4 Short Motion

The Remote Motion Interface does not support motion shorter than 40 ms. If a motion instruction is too short, the robot controller slows down the motion so that the motion is at least 40 ms in duration.

2 REMOTE MOTION APPLICATION PROGRAM INTERFACE

2.1 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 JSON string.

NOTE
Please termination of each JSON string use \r\n.

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 controller sends to the robot controller to terminate its Remote Motion Interface session.

2.2.1 Packet Exchange

External 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 PortNumber 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.

2.2.2 Controller Disconnect Packet

Remote device command packet	Remote 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

Remote controller termination packet	Remote controller return packet
{"Communication": "FRC_Terminate"} \r\n	None

When an external 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 external device. Once the connection is terminated, the external device needs to send an FRC_Connect packet to the controller to reestablish the connection.

2.2.4 System Fault Packet

Remote controller system fault packet	Remote controller 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 limit error, the

controller will send out this packet to inform the extern 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

2. Other commands will work depend 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	Remote controller return packet
{“Command”: “FRC_Initialize”} \r\n	{“Command “ : “FRC_Initialize”, “ErrorID “: integerValue } \r\n

The remote device sends this packet to the controller when it wants to start sending motion commands. Before sending this command to 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 in ready to run; no servo or other errors.
3. The selected TP program is not RMI_MOVE.

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 TP program RMI_MOVE and the program is running. Otherwise, the integerValue of the return packet is the error code from the robot controller.

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

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 TP program RMI_MOVE. 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 TP program RMI_MOVE 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 Resume 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 controller to resume the TP program. If program is resumed, the retruned 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”} \r\n	{“Command” : “FRC_ReadError”, “ErrorID” : integerValue, “ErrorData”: Error Sting} \r\n

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.

2.3.6 Packet to Set the Current UFrame-Utool Number

Remote device command packet	Robot controller return packet
{“Command” : “FRC_SetUFrameUTool”, “UFrameNumber” : byteValue1, “UToolNumber” : byteValue2} \r\n	{“Command” : “FRC_SetUFrameUTool”, “ErrorID” : integerValue} \r\n

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.

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
{“Command”: “FRC_GetStatus”} \r\n	{“Command” : “FRC_GetStatus”, “ErrorID” : integerValue, “ServoReady” : byteValue1, “TPMode” : byteValue2, “RMIMotionStatus”: byteValue3, “ProgramStatus” : byteValue4, “SingleStepMode” : byteValue5, “NumberUTool”: byteValue6, “NextSequenceID” : integerValue1, “NumberUFrame”: byteValue7} \r\n

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

- integerValue : 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 : 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 : 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 : 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: The TP program RMI_MOVE's current status. If 1, the TP program RMI_MOVE is aborted.
- byteValue5: If 1, the robot controller is in single-step mode.
- byteValue6 : Number of user tools available in the robot controller.
- byteValue7 : Number of user frames available in the robot controller.
- integerValue1 : 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
{“Command”: “FRC_ReadUFrameData”, “FrameNumber” : byteValue} \r\n	{“Command” : “FRC_ReadUFrameData”, “ErrorID” : integerValue, “UFrameNumber” : byteValue, “Frame” : { “X” : floatX, “Y” : floatY, “Z” : floatZ, “W” : floatW, “P” : floatP, “R” : floatR } } \r\n

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.

2.3.9 Packet to Set User Frame Data

Remote device command packet	Robot controller return packet
{“Command”: “FRC_WriteUFrameData”, “FrameNumber”: byteValue, “Frame”: {“X”: floatX, “Y”: floatY, “Z”: floatZ, “W”: floatW, “P”: floatP, “R”: FloatR} } \r\n	{“Command”: “FRC_WriteUFrameData” , “ErrorID”: integerValue} \r\n

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.

2.3.10 Packet to Read User Tool Data

Remote device command packet	Robot controller return packet
{“Command”: “FRC_ReadUToolData”, “ToolNumber”: byteValue} \r\n	{“Command”: “FRC_ReadUToolData”, “ErrorID” : integerValue, “UToolNumber” : byteValue, “Frame”: {“ X”: floatX, “Y”: floatY, “Z”: floatZ, “W” : floatW, “P”: floatW, “R”: floatR} } \r\n

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.

2.3.11 Packet to Set User tool Data

Remote device command packet	Robot controller return packet
{“Command”: “FRC_WriteUToolData”, “ToolNumber”: byteValue, “Frame”: { “X”: floatX, “Y”: floatY, “Z”: floatZ, “W”floatW, “P”: floatP, “R”: FloatR} } \r\n	{“Command”: “FRC_WriteUToolData”, “ErrorID”: integerValue} \r\n

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.

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
{“Command” : “FRC_ReadDIN”, “PortNumber”: shortValue} \r\n	{“Command” : “FRC_ReadDIN”, “ErrorID” : integerValue, “PortNumber” : shortValue, “PortValue”: byteValue} \r\n

You can read a digital I/O 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
{“Command” : “FRC_WriteDOUT”, “PortNumber” : byteValue, “PortValue”: StringValue} \r\n	{“Command” : “FRC_WriteDOUT”, “ErrorID” : integerValue} \r\n

Use this packet to set a digital output port where the byteValue 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
{“Command”: “FRC_ReadCartesianPosition”} \r \n	{“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} } \r\n

Use this packet to get the current robot’s Cartesian position. If 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.

2.3.15 Packet to Read Current Robot Joint Angles

Remote device command packet	Robot controller return packet
{“Command”: “FRC_ReadJointAngles”} \r\n	{“Command”: “FRC_ReadJointAngles”, “ErrorID” : integerValue1, “TimeTag”: integerValue2, “JointAngle”: {“J1”: floatJ1, “J2”: floatJ2, “J3”: floatJ3, “J4”: floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}} \r\n

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

2.3.16 Packet to Set Speed Override

Remote device command packet	Robot controller return packet
{"Command": "FRC_SetOverRide", "Value": byteValue} \r\n	{"Command" : "FRC_SetOverRide", "ErrorID" : integerValue} \r\n

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
{"Command": "FRC_GetUFrameUTool"} \r\n	{"Command": "FRC_GetUFrameUtool", "ErrorID" : integerValue, "UFrameNumber" : byteValue1, "UToolNumber": byteValue2} \r\n

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.

2.3.18 Packet to Read Position Register Data

Remote device command packet	Robot controller return packet
{Command: FRC_ReadPositionRegister, RegisterNumber: shortValue} \r\n	{"Command": "FRC_ReadePositionRegister", "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} } \r\n

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

2.3.19 Packet to Write Position Register Data

Remote device command packet	Robot controller return packet
{"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} } \r\n	{"Command": "FRC_WriteUFrameData", "ErrorID": integerValue} \r\n

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.

2.3.20 Packet to Reset Robot Controller

Remote device command packet	Robot controller return packet
{"Command": "FRC_Reset"} \r\n	{"Command" : "FRC_Reset" , "ErrorID": integerValue} \r\n

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
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.

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 TP Program RMI_MOVE 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 execute it when it completed 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 sent 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 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.

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 external device knows which instruction has been executed.

NOTE
Start your SeqeuenceID number from 1 after the FRC_Initialize packet. 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 "NextSeqeunceID" : 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 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_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
FRC_CircularRelative	C P[13] P[14] 100 mm/sec FINE INC
FRC_JointMotionJRep	J P[15] 10% CNT100
FRC_JointRelativeJRep	J P[16] 5% 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[]
- ACC
- VODDSET, VR[]
- Local Condition Block, TA/TB/DB

2.4.1 Packet to Wait for DIN Instruction

External device command 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 command 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 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 command 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 command 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 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 command packet	Robot controller return packet
{“Instruction” : “FRC_PayLoad”, “SequenceID” : integerValue, “ScheduleNumber”: byteValue} \r\n	{“Instruction” : “FRC_PayLoad”, “ErrorID” : integerValue1, “SequenceID” : integerValue2} \r\n

The FRC_PayLoad 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 Add Linear Motion Instruction

Remote device command packet	Robot controller return packet

{“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,
	{“Instruction”: “FRC_LinearMotion”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

The following items are optional. “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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” : inch/min
  - “Time” : 0.1seconds

- 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 to 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.

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 : shortValue1. You can also add a position register offset to the motion instruction. The shortValue1 is the register number. Assuming the shortValue1 = 2, then the instruction becomes “L P[10] 150 mm/sec CNT100 Offset, PR[2]”.
- VisionPRNumber : shortValue2 adds a vision offset to the linear motion. The shortValue2 is the vision offset position register number. Assuming the shortValue2 = 5 then the linear motion instruction becomes “L P[10] 150 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] 150 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] 150 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: shortValue3. This shortValue3 indicates the “TB”/”TA”/”DB” value. For “DB”, the unit is 0.01 mm. For “TA” and “TB” , the unit is ms.

- PortType : byteValue11. The byteValue11 can be either
  - ○ 1: DOUT
  - ○ 2: ROUT

NOTE
GO and AO are not supported

- portNumber : shortValue4: The shortValue4 is the DOUT/ROUT port number.
- PortType : 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 executed 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.

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 Packet to Add Linear Incremental Motion Instruction

Remote device command packet	Robot controller return packet

{“Instruction”: “FRC_LinearRelative”,
“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,
	{“Instruction”: “FRC_LinearRelative”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

The following items are optional. “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “WristJoint”: “ON”, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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 Packet to Add Joint Motion Instruction

Remote device command packet	Robot controller return packet

{“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,
	{“Instruction” : “FRC_JointMotion”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

The following items are optional. “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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” : msec

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 Packet to Add Joint Incremental Motion Instruction

Remote device command packet	Robot controller return packet

{“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,
	{“Instruction” : “FRC_JointRelative”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

The following items are optional. “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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 Packet to Add Circular Motion Instruction

Remote device command packet	Robot controller return packet

{“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,
	{“Instruction”: “FRC_CircularMotion”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

Remote device command packet	Robot controller return packet
The following items are optional. "ACC" : byteValue20, "OffsetPRNumber": shortValue1, "VisionPRNumber": shortValue2, "WristJoint": "ON", "MROT": "ON", "LCBType" : stringValue3, "LCBValue" : shortValue3, "PortType" : byteValue21, "PortNumber" : ShortValue4, "PortValue": stringValue4} \r\n	

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 and the joint motion. Please refer to the linear motion instruction for a detailed description.

After the controller executed 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 Packet to Add Circular Incremental Motion Instruction

Remote device command packet	Robot controller return packet

{“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,
	{“Instruction”: “FRC_CircularRelative”,
	“ErrorID” : integerValue1,
	“SequenceID”: integerValue2} \r\n

Remote device command packet	Robot controller return packet
The following items are optional. "ACC" : byteValue20, "OffsetPRNumber": shortValue1, "VisionPRNumber": shortValue2, "WristJoint": "ON", "MROT": "ON", "LCBType" : stringValue3, "LCBValue" : shortValue3, "PortType" : byteValue21, "PortNumber" : ShortValue4, "PortValue": stringValue4} \r\n	

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 Packet to Add Joint Motion with Joint Representation

Remote device command packet	Robot controller return packet
{“Instruction”: “FRC_JointMotionJRep”, “SequenceID” : integerValue, “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”: byteValue9,	{“Instruction”: FRC_JointMotionJRep”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n
The following items are optional: “ACC” : byteValue10, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue11, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

This packet adds a joint motion instruction to the TP program RMI_MOVE, 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 Packet to Add Joint Incremental Motion with Joint Representation

Remote device command packet	Robot controller return packet
{“Instruction”: “FRC_JointRelativeJRep”, “SequenceID” : integerValue, “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”: byteValue9,	{“Instruction” : “FRC_JointRelativeJRep”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n
The following items are optional: “ACC” : byteValue10, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue11, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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 Packet to Add Linear Motion with Joint Representation

Remote device command packet	Robot controller return packet
{“Instruction”: “FRC_LinearMotionJRep”, “SequenceID” : integerValue, “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,	{“Instruction”: FRC_LinearMotionJRep”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n
The following items are optional: “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “WristJoint” : “ON”, “MROT”: “ON”, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

This packet adds a linear motion instruction to the TP program RMI_MOVE, the only difference between this packet and the FRC_LinearMotion 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] 150mm/sec CNT100

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.

2.4.15 Packet to Add Linear Incremental Motion with Joint Representation

Remote device command packet	Robot controller return packet
{“Instruction”: “FRC_LinearRelativeJRep”, “SequenceID” : integerValue, “JointAngle d ” : { “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,	{“Instruction” : “FRC_LinearRelativeJRep”, “ErrorID” : integerValue1, “SequenceID”: integerValue2} \r\n
The following items are optional: “ACC” : byteValue11, “OffsetPRNumber”: shortValue1, “VisionPRNumber”: shortValue2, “WristJoint” : “ON”, “MROT”: ON, “LCBType” : stringValue3, “LCBValue” : shortValue3, “PortType” : byteValue12, “PortNumber” : ShortValue4, “PortValue”: stringValue4} \r\n	

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 integeValue1 is the error code.

2.4.16 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 JSON string error:

Remote device command packet	Robot controller return packet
RMI cannot interpret the JSON string	{“Command”: “Unknown”, “ErrorID”: “integerValue”} \r\n

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

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 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.

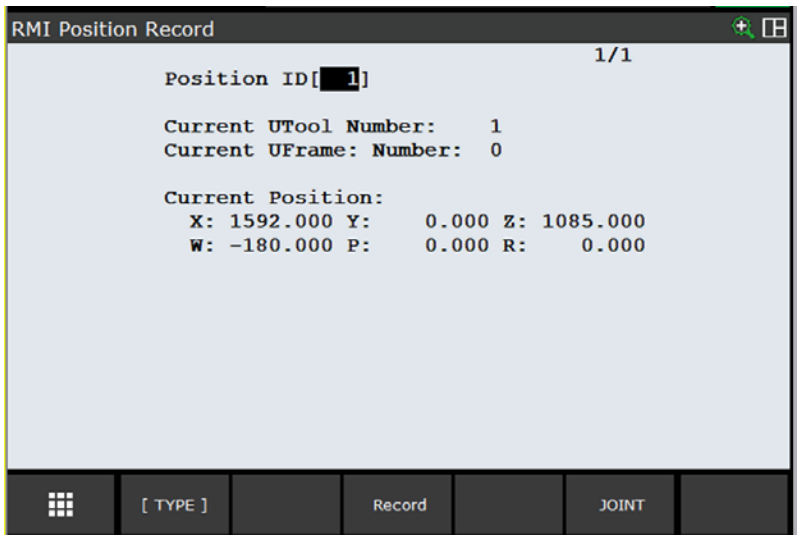

Fig. 2.5.2 (a) RMI Position Record

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

Robot controller sent packet	Extermal device return packet

{ “Command” : “TouchUp”,
“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
	None

Robot controller sent packet	Extermal device return packet
{“Command” : “TouchUp”, “PositionID”: UShortValue, “Representation” : “Joint”, “JointAngles” : { “J1”: floatJ1, “J2”: floatJ2, “J3” : floatJ3, “J4” : floatJ4, “J5”: floatJ5, “J6”: floatJ6, “J7”: floatJ7, “J8”: floatJ8, “J9”: floatJ9}} \r\n	None

3 REMOTE DEVICE CONTROL FLOW

3.1 OVERVIEW

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.

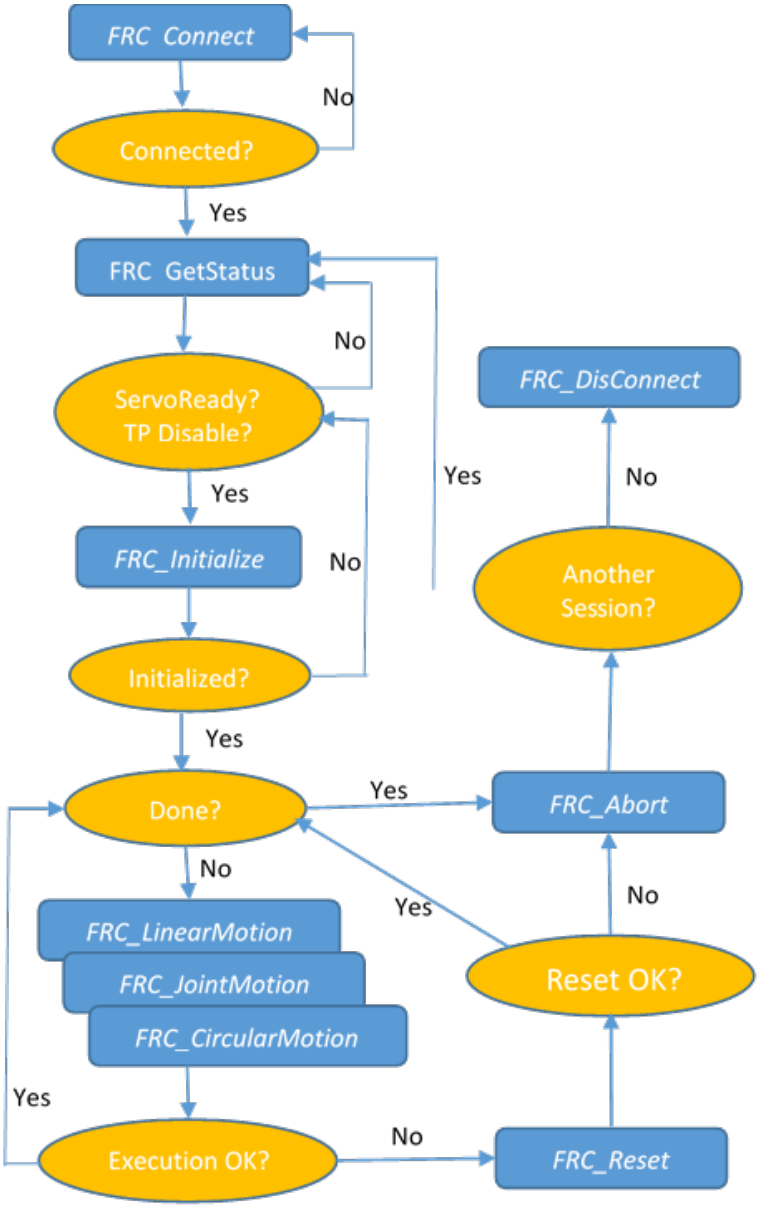

Fig. 3.1 (a) Remote Device Control Flow Chart

3.2 HOLD STATE ERROR HANDLING

The controller goes to the HOLD state when the RMI receives a motion instruction with an invalid motion option. For example, if the controller does not have R640 (MROT) software loaded and it receives a motion instruction with 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.

3.3 JUMP AND SKIP INSTRUCTIONS

RMI don't have a JUMP or SKIP instruction since this TP program is dynamically updated during its execution in the running TP program RMI_MOVE. Therefore, by calling FRC_Abort to terminate the current RMI_MOVE TP program, and use FRC_Initalize to create the program again, processing equivalent to the SKIP or JUMP instruction can be performed. When the new TP program is created, you can add the required TP instructions back to the new RMI_MOVE TP program.

3.4 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 none-warning error, such as a limit error, you have to do the following to recover:

  – Abort the current TP program RMI_MOVE 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 TP program RMI_MOVE again following a new set of instructions to continue your process.

3.5 VISION PROCESS

In order to use FANUC vision, you can run your vision process as a background task. The Remote Motion Interface can sent 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.

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.

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) Exsample Log File

From Fig. 4(a), you can see the controller receives an FRC_Initialize packet from the external device at the controller's tick time 55567 (each tick is 2 ms). The controller returns the FRC_Intialize 8 ms later and the TP program is initialized successfully.

The external device immediately sends the FRC_SetUFrame instruction and the controller executes this instruction 8 ms later.

INDEX

<U>

<V>

REVISION RECORD

Edition	Date	Contents
01	Mar., 2020	

B-84184EN/01