FANUC Robot series

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

System Design Tool

OPERATOR'S MANUAL

MAROBDSGN07121E REV. B

FANUC America Corporation
3900 W. Hamlin Road
Rochester Hills, Michigan 48309–3253

B-83274EN/02

Copyrights and Trademarks

WARNING

This equipment generates, uses, and can radiate radiofrequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for Class A computing devices pursuant to subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of the equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measure may be required to correct the interference.

FANUC conducts courses on its systems and products on a regularly scheduled basis at the company's world headquarters in Rochester Hills, Michigan. For additional information contact

FANUC America Corporation
Training Department
3900 W. Hamlin Road
Rochester Hills, Michigan 48309-3253
www.fanucrobotics.com

For customer assistance, including Technical Support, Service, Parts & Part Repair, and Marketing Requests, contact the Customer Resource Center, 24 hours a day, at 1-800-47-ROBOT (1-800-477-6268). International customers should call 011-1-248-377-7159.

Send your comments and suggestions about this manual to:
product.documentation@fanucrobotics.com

Patents

One or more of the following U.S. patents might be related to the FANUC products described in this manual.

FANUC America Corporation Patent List

4,630,567 4,639,878 4,707,647 4,708,175 4,708,580 4,942,539 4,984,745
5,238,029 5,239,739 5,272,805 5,293,107 5,293,911 5,331,264 5,367,944
5,373,221 5,421,218 5,434,489 5,644,898 5,670,202 5,696,687 5,737,218
5,823,389 5,853,027 5,887,800 5,941,679 5,959,425 5,987,726 6,059,092
6,064,168 6,070,109 6,086,294 6,122,062 6,147,323 6,204,620 6,243,621
6,253,799 6,285,920 6,313,595 6,325,302 6,345,818 6,356,807 6,360,143
6,378,190 6,385,508 6,425,177 6,477,913 6,490,369 6,518,980 6,540,104
6,541,757 6,560,513 6,569,258 6,612,449 6,703,079 6,705,361 6,726,773
6,768,078 6,845,295 6,945,483 7,149,606 7,149,606 7,211,978 7,266,422
7,399,363

FANUC CORPORATION Patent List

4,571,694 4,626,756 4,700,118 4,706,001 4,728,872 4,732,526 4,742,207
4,835,362 4,894,596 4,899,095 4,920,248 4,931,617 4,934,504 4,956,594
4,967,125 4,969,109 4,970,370 4,970,448 4,979,127 5,004,968 5,006,035
5,008,834 5,063,281 5,066,847 5,066,902 5,093,552 5,107,716 5,111,019
5,130,515 5,136,223 5,151,608 5,170,109 5,189,351 5,267,483 5,274,360
5,292,066 5,300,868 5,304,906 5,313,563 5,319,443 5,325,467 5,327,057
5,329,469 5,333,242 5,337,148 5,371,452 5,375,480 5,418,441 5,432,316
5,440,213 5,442,155 5,444,612 5,449,875 5,451,850 5,461,478 5,463,297
5,467,003 5,471,312 5,479,078 5,485,389 5,485,552 5,486,679 5,489,758
5,493,192 5,504,766 5,511,007 5,520,062 5,528,013 5,532,924 5,548,194
5,552,687 5,558,196 5,561,742 5,570,187 5,570,190 5,572,103 5,581,167
5,582,750 5,587,635 5,600,759 5,608,299 5,608,618 5,624,588 5,630,955
5,637,969 5,639,204 5,641,415 5,650,078 5,658,121 5,668,628 5,687,295
5,691,615 5,698,121 5,708,342 5,715,375 5,719,479 5,727,132 5,742,138
5,742,144 5,748,854 5,749,058 5,760,560 5,773,950 5,783,922 5,799,135
5,812,408 5,841,257 5,845,053 5,872,894 5,887,122 5,911,892 5,912,540
5,920,678 5,937,143 5,980,082 5,983,744 5,987,591 5,988,850 6,023,044
6,032,086 6,040,554 6,059,169 6,088,628 6,097,169 6,114,824 6,124,693
6,140,788 6,141,863 6,157,155 6,160,324 6,163,124 6,177,650 6,180,898
6,181,096 6,188,194 6,208,105 6,212,444 6,219,583 6,226,181 6,236,011
6,236,896 6,250,174 6,278,902 6,279,413 6,285,921 6,298,283 6,321,139
6,324,443 6,328,523 6,330,493 6,340,875 6,356,671 6,377,869 6,382,012
6,384,371 6,396,030 6,414,711 6,424,883 6,431,018 6,434,448 6,445,979
6,459,958 6,463,358 6,484,067 6,486,629 6,507,165 6,654,666 6,665,588
6,680,461 6,696,810 6,728,417 6,763,284 6,772,493 6,845,296 6,853,881
6,888,089 6,898,486 6,917,837 6,928,337 6,965,091 6,970,802 7,038,165
7,069,808 7,084,900 7,092,791 7,133,747 7,143,100 7,149,602 7,131,848
7,161,321 7,171,041 7,174,234 7,173,213 7,177,722 7,177,439 7,181,294
7,181,313 7,280,687 7,283,661 7,291,806 7,299,713 7,315,650 7,324,873
7,328,083 7,330,777 7,333,879 7,355,725 7,359,817 7,373,220 7,376,488
7,386,367 7,464,623 7,447,615 7,445,260 7,474,939 7,486,816 7,495,192
7,501,778 7,502,504 7,508,155 7,512,459 7,525,273 7,526,121

Conventions

⚠**WARNING**

Information appearing under the "WARNING" caption concerns the protection of personnel. It is boxed and bolded to set it apart from the surrounding text.

⚠**CAUTION**

Information appearing under the "CAUTION" caption concerns the protection of equipment, software, and data. It is boxed and bolded to set it apart from the surrounding text.

Note Information appearing next to NOTE concerns related information or useful hints.

• Original Instructions

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

Safety

FANUC America Corporation is not and does not represent itself as an expert in safety systems, safety equipment, or the specific safety aspects of your company and/or its work force. It is the responsibility of the owner, employer, or user to take all necessary steps to guarantee the safety of all personnel in the workplace.

The appropriate level of safety for your application and installation can be best determined by safety system professionals. FANUC America Corporation therefore, recommends that each customer consult with such professionals in order to provide a workplace that allows for the safe application, use, and operation of FANUC America Corporation systems.

According to the industry standard ANSI/RIA R15-06, the owner or user is advised to consult the standards to ensure compliance with its requests for Robotics System design, usability, operation, maintenance, and service. Additionally, as the owner, employer, or user of a robotic system, it is your responsibility to arrange for the training of the operator of a robot system to recognize and respond to known hazards associated with your robotic system and to be aware of the recommended operating procedures for your particular application and robot installation.

Ensure that the robot being used is appropriate for the application. Robots used in classified (hazardous) locations must be certified for this use.

FANUC America Corporation therefore, recommends that all personnel who intend to operate, program, repair, or otherwise use the robotics system be trained in an approved FANUC America Corporation training course and become familiar with the proper operation of the system. Persons responsible for programming the system–including the design, implementation, and debugging of application programs–must be familiar with the recommended programming procedures for your application and robot installation.

The following guidelines are provided to emphasize the importance of safety in the workplace.

CONSIDERING SAFETY FOR YOUR ROBOT INSTALLATION

Safety is essential whenever robots are used. Keep in mind the following factors with regard to safety:

- The safety of people and equipment
- Use of safety enhancing devices
- Techniques for safe teaching and manual operation of the robot(s)
- Techniques for safe automatic operation of the robot(s)
- Regular scheduled inspection of the robot and workcell
- Proper maintenance of the robot

Keeping People Safe

The safety of people is always of primary importance in any situation. When applying safety measures to your robotic system, consider the following:

- External devices
- Robot(s)
- Tooling
- Workpiece

Using Safety Enhancing Devices

Always give appropriate attention to the work area that surrounds the robot. The safety of the work area can be enhanced by the installation of some or all of the following devices:

- Safety fences, barriers, or chains
- Light curtains
- Interlocks
- Pressure mats
- Floor markings
- Warning lights
- Mechanical stops
- EMERGENCY STOP buttons
- DEADMAN switches

Setting Up a Safe Workcell

A safe workcell is essential to protect people and equipment. Observe the following guidelines to ensure that the workcell is set up safely. These suggestions are intended to supplement and not replace existing federal, state, and local laws, regulations, and guidelines that pertain to safety.

- Sponsor your personnel for training in approved FANUC America Corporation training course(s) related to your application. Never permit untrained personnel to operate the robots.
- Install a lockout device that uses an access code to prevent unauthorized persons from operating the robot.
- Use anti–tie–down logic to prevent the operator from bypassing safety measures.
- Arrange the workcell so the operator faces the workcell and can see what is going on inside the cell.
- Clearly identify the work envelope of each robot in the system with floor markings, signs, and special barriers. The work envelope is the area defined by the maximum motion range of the robot, including any tooling attached to the wrist flange that extend this range.

- Position all controllers outside the robot work envelope.
- Never rely on software or firmware based controllers as the primary safety element unless they comply with applicable current robot safety standards.
- Mount an adequate number of EMERGENCY STOP buttons or switches within easy reach of the operator and at critical points inside and around the outside of the workcell.
- Install flashing lights and/or audible warning devices that activate whenever the robot is operating, that is, whenever power is applied to the servo drive system. Audible warning devices shall exceed the ambient noise level at the end–use application.
- Wherever possible, install safety fences to protect against unauthorized entry by personnel into the work envelope.
- Install special guarding that prevents the operator from reaching into restricted areas of the work envelope.
- Use interlocks.
- Use presence or proximity sensing devices such as light curtains, mats, and capacitance and vision systems to enhance safety.
- Periodically check the safety joints or safety clutches that can be optionally installed between the robot wrist flange and tooling. If the tooling strikes an object, these devices dislodge, remove power from the system, and help to minimize damage to the tooling and robot.
- Make sure all external devices are properly filtered, grounded, shielded, and suppressed to prevent hazardous motion due to the effects of electro–magnetic interference (EMI), radio frequency interference (RFI), and electro–static discharge (ESD).
- Make provisions for power lockout/tagout at the controller.
- Eliminate *pinch points*. Pinch points are areas where personnel could get trapped between a moving robot and other equipment.
- Provide enough room inside the workcell to permit personnel to teach the robot and perform maintenance safely.
- Program the robot to load and unload material safely.
- If high voltage electrostatics are present, be sure to provide appropriate interlocks, warning, and beacons.
- If materials are being applied at dangerously high pressure, provide electrical interlocks for lockout of material flow and pressure.

Staying Safe While Teaching or Manually Operating the Robot

Advise all personnel who must teach the robot or otherwise manually operate the robot to observe the following rules:

- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Know whether or not you are using an intrinsically safe teach pendant if you are working in a hazardous environment.

- Before teaching, visually inspect the robot and work envelope to make sure that no potentially hazardous conditions exist. The work envelope is the area defined by the maximum motion range of the robot. These include tooling attached to the wrist flange that extends this range.
- The area near the robot must be clean and free of oil, water, or debris. Immediately report unsafe working conditions to the supervisor or safety department.
- FANUC America Corporation recommends that no one enter the work envelope of a robot that is on, except for robot teaching operations. However, if you must enter the work envelope, be sure all safeguards are in place, check the teach pendant DEADMAN switch for proper operation, and place the robot in teach mode. Take the teach pendant with you, turn it on, and be prepared to release the DEADMAN switch. Only the person with the teach pendant should be in the work envelope.

⚠WARNING

Never bypass, strap, or otherwise deactivate a safety device, such as a limit switch, for any operational convenience. Deactivating a safety device is known to have resulted in serious injury and death.

- Know the path that can be used to escape from a moving robot; make sure the escape path is never blocked.
- Isolate the robot from all remote control signals that can cause motion while data is being taught.
- Test any program being run for the first time in the following manner:

⚠WARNING

Stay outside the robot work envelope whenever a program is being run. Failure to do so can result in injury.

  - Using a low motion speed, single step the program for at least one full cycle.
  - Using a low motion speed, test run the program continuously for at least one full cycle.
  - Using the programmed speed, test run the program continuously for at least one full cycle.
- Make sure all personnel are outside the work envelope before running production.

Staying Safe During Automatic Operation

Advise all personnel who operate the robot during production to observe the following rules:

- Make sure all safety provisions are present and active.

- Know the entire workcell area. The workcell includes the robot and its work envelope, plus the area occupied by all external devices and other equipment with which the robot interacts.
- Understand the complete task the robot is programmed to perform before initiating automatic operation.
- Make sure all personnel are outside the work envelope before operating the robot.
- Never enter or allow others to enter the work envelope during automatic operation of the robot.
- Know the location and status of all switches, sensors, and control signals that could cause the robot to move.
- Know where the EMERGENCY STOP buttons are located on both the robot control and external control devices. Be prepared to press these buttons in an emergency.
- Never assume that a program is complete if the robot is not moving. The robot could be waiting for an input signal that will permit it to continue its activity.
- If the robot is running in a pattern, do not assume it will continue to run in the same pattern.
- Never try to stop the robot, or break its motion, with your body. The only way to stop robot motion immediately is to press an EMERGENCY STOP button located on the controller panel, teach pendant, or emergency stop stations around the workcell.

Staying Safe During Inspection

When inspecting the robot, be sure to

- Turn off power at the controller.
- Lock out and tag out the power source at the controller according to the policies of your plant.
- Turn off the compressed air source and relieve the air pressure.
- If robot motion is not needed for inspecting the electrical circuits, press the EMERGENCY STOP button on the operator panel.
- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- If power is needed to check the robot motion or electrical circuits, be prepared to press the EMERGENCY STOP button, in an emergency.
- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.

Staying Safe During Maintenance

When performing maintenance on your robot system, observe the following rules:

- Never enter the work envelope while the robot or a program is in operation.
- Before entering the work envelope, visually inspect the workcell to make sure no potentially hazardous conditions exist.

- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Consider all or any overlapping work envelopes of adjoining robots when standing in a work envelope.
- Test the teach pendant for proper operation before entering the work envelope.
- If it is necessary for you to enter the robot work envelope while power is turned on, you must be sure that you are in control of the robot. Be sure to take the teach pendant with you, press the DEADMAN switch, and turn the teach pendant on. Be prepared to release the DEADMAN switch to turn off servo power to the robot immediately.
- Whenever possible, perform maintenance with the power turned off. Before you open the controller front panel or enter the work envelope, turn off and lock out the 3–phase power source at the controller.
- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.

⚠WARNING

Lethal voltage is present in the controller WHENEVER IT IS CONNECTED to a power source. Be extremely careful to avoid electrical shock. HIGH VOLTAGE IS PRESENT at the input side whenever the controller is connected to a power source. Turning the disconnect or circuit breaker to the OFF position removes power from the output side of the device only.

- Release or block all stored energy. Before working on the pneumatic system, shut off the system air supply and purge the air lines.
- Isolate the robot from all remote control signals. If maintenance must be done when the power is on, make sure the person inside the work envelope has sole control of the robot. The teach pendant must be held by this person.
- Make sure personnel cannot get trapped between the moving robot and other equipment. Know the path that can be used to escape from a moving robot. Make sure the escape route is never blocked.
- Use blocks, mechanical stops, and pins to prevent hazardous movement by the robot. Make sure that such devices do not create pinch points that could trap personnel.

⚠WARNING

Do not try to remove any mechanical component from the robot before thoroughly reading and understanding the procedures in the appropriate manual. Doing so can result in serious personal injury and component destruction.

- Be aware that when you remove a servomotor or brake, the associated robot arm will fall if it is not supported or resting on a hard stop. Support the arm on a solid support before you release the brake.
- When replacing or installing components, make sure dirt and debris do not enter the system.
- Use only specified parts for replacement. To avoid fires and damage to parts in the controller, never use nonspecified fuses.
- Before restarting a robot, make sure no one is inside the work envelope; be sure that the robot and all external devices are operating normally.

KEEPING MACHINE TOOLS AND EXTERNAL DEVICES SAFE

Certain programming and mechanical measures are useful in keeping the machine tools and other external devices safe. Some of these measures are outlined below. Make sure you know all associated measures for safe use of such devices.

Programming Safety Precautions

Implement the following programming safety measures to prevent damage to machine tools and other external devices.

- Back–check limit switches in the workcell to make sure they do not fail.
- Implement ‘‘failure routines” in programs that will provide appropriate robot actions if an external device or another robot in the workcell fails.
- Use *handshaking* protocol to synchronize robot and external device operations.
- Program the robot to check the condition of all external devices during an operating cycle.

Mechanical Safety Precautions

Implement the following mechanical safety measures to prevent damage to machine tools and other external devices.

- Make sure the workcell is clean and free of oil, water, and debris.
- Use DCS (Dual Check Safety), software limits, limit switches, and mechanical hardstops to prevent undesired movement of the robot into the work area of machine tools and external devices.

KEEPING THE ROBOT SAFE

Observe the following operating and programming guidelines to prevent damage to the robot.

Operating Safety Precautions

The following measures are designed to prevent damage to the robot during operation.

- Use a low override speed to increase your control over the robot when jogging the robot.
- Visualize the movement the robot will make before you press the jog keys on the teach pendant.
- Make sure the work envelope is clean and free of oil, water, or debris.
- Use circuit breakers to guard against electrical overload.

Programming Safety Precautions

The following safety measures are designed to prevent damage to the robot during programming:

- Establish *interference zones* to prevent collisions when two or more robots share a work area.
- Make sure that the program ends with the robot near or at the home position.
- Be aware of signals or other operations that could trigger operation of tooling resulting in personal injury or equipment damage.
- In dispensing applications, be aware of all safety guidelines with respect to the dispensing materials.

NOTE: Any deviation from the methods and safety practices described in this manual must conform to the approved standards of your company. If you have questions, see your supervisor.

ADDITIONAL SAFETY CONSIDERATIONS FOR PAINT ROBOT INSTALLATIONS

Process technicians are sometimes required to enter the paint booth, for example, during daily or routine calibration or while teaching new paths to a robot. Maintenance personnel also must work inside the paint booth periodically.

Whenever personnel are working inside the paint booth, ventilation equipment must be used. Instruction on the proper use of ventilating equipment usually is provided by the paint shop supervisor.

Although paint booth hazards have been minimized, potential dangers still exist. Therefore, today’s highly automated paint booth requires that process and maintenance personnel have full awareness of the system and its capabilities. They must understand the interaction that occurs between the vehicle moving along the conveyor and the robot(s), hood/deck and door opening devices, and high–voltage electrostatic tools.

CAUTION

Ensure that all ground cables remain connected. Never operate the paint robot with ground provisions disconnected. Otherwise, you could injure personnel or damage equipment.

Paint robots are operated in three modes:

- Teach or manual mode
- Automatic mode, including automatic and exercise operation
- Diagnostic mode

During both teach and automatic modes, the robots in the paint booth will follow a predetermined pattern of movements. In teach mode, the process technician teaches (programs) paint paths using the teach pendant.

In automatic mode, robot operation is initiated at the System Operator Console (SOC) or Manual Control Panel (MCP), if available, and can be monitored from outside the paint booth. All personnel must remain outside of the booth or in a designated safe area within the booth whenever automatic mode is initiated at the SOC or MCP.

In automatic mode, the robots will execute the path movements they were taught during teach mode, but generally at production speeds.

When process and maintenance personnel run diagnostic routines that require them to remain in the paint booth, they must stay in a designated safe area.

Paint System Safety Features

Process technicians and maintenance personnel must become totally familiar with the equipment and its capabilities. To minimize the risk of injury when working near robots and related equipment, personnel must comply strictly with the procedures in the manuals.

This section provides information about the safety features that are included in the paint system and also explains the way the robot interacts with other equipment in the system.

The paint system includes the following safety features:

- Most paint booths have red warning beacons that illuminate when the robots are armed and ready to paint. Your booth might have other kinds of indicators. Learn what these are.

- Some paint booths have a blue beacon that, when illuminated, indicates that the electrostatic devices are enabled. Your booth might have other kinds of indicators. Learn what these are.
- EMERGENCY STOP buttons are located on the robot controller and teach pendant. Become familiar with the locations of all E–STOP buttons.
- An intrinsically safe teach pendant is used when teaching in hazardous paint atmospheres.
- A DEADMAN switch is located on each teach pendant. When this switch is held in, and the teach pendant is on, power is applied to the robot servo system. If the engaged DEADMAN switch is released or pressed harder during robot operation, power is removed from the servo system, all axis brakes are applied, and the robot comes to an EMERGENCY STOP. Safety interlocks within the system might also E–STOP other robots.

WARNING

An EMERGENCY STOP will occur if the DEADMAN switch is released on a bypassed robot.

- Overtravel by robot axes is prevented by software limits. All of the major and minor axes are governed by software limits. DCS (Dual Check Safety), limit switches and hardstops also limit travel by the major axes.
- EMERGENCY STOP limit switches and photoelectric eyes might be part of your system. Limit switches, located on the entrance/exit doors of each booth, will EMERGENCY STOP all equipment in the booth if a door is opened while the system is operating in automatic or manual mode. For some systems, signals to these switches are inactive when the switch on the SOC is in teach mode.
- When present, photoelectric eyes are sometimes used to monitor unauthorized intrusion through the entrance/exit silhouette openings.
- System status is monitored by computer. Severe conditions result in automatic system shutdown.

Staying Safe While Operating the Paint Robot

When you work in or near the paint booth, observe the following rules, in addition to all rules for safe operation that apply to all robot systems.

WARNING

Observe all safety rules and guidelines to avoid injury.

WARNING

Never bypass, strap, or otherwise deactivate a safety device, such as a limit switch, for any operational convenience. Deactivating a safety device is known to have resulted in serious injury and death.

WARNING

Enclosures shall not be opened unless the area is known to be nonhazardous or all power has been removed from devices within the enclosure. Power shall not be restored after the enclosure has been opened until all combustible dusts have been removed from the interior of the enclosure and the enclosure purged. Refer to the Purge chapter for the required purge time.

- Know the work area of the entire paint station (workcell).
- Know the work envelope of the robot and hood/deck and door opening devices.
- Be aware of overlapping work envelopes of adjacent robots.
- Know where all red, mushroom–shaped EMERGENCY STOP buttons are located.
- Know the location and status of all switches, sensors, and/or control signals that might cause the robot, conveyor, and opening devices to move.
- Make sure that the work area near the robot is clean and free of water, oil, and debris. Report unsafe conditions to your supervisor.
- Become familiar with the complete task the robot will perform BEFORE starting automatic mode.
- Make sure all personnel are outside the paint booth before you turn on power to the robot servo system.
- Never enter the work envelope or paint booth before you turn off power to the robot servo system.
- Never enter the work envelope during automatic operation unless a safe area has been designated.
- Never wear watches, rings, neckties, scarves, or loose clothing that could get caught in moving machinery.
- Remove all metallic objects, such as rings, watches, and belts, before entering a booth when the electrostatic devices are enabled.
- Stay out of areas where you might get trapped between a moving robot, conveyor, or opening device and another object.
- Be aware of signals and/or operations that could result in the triggering of guns or bells.
- Be aware of all safety precautions when dispensing of paint is required.
- Follow the procedures described in this manual.

Special Precautions for Combustible Dusts (Powder Paint)

When the robot is used in a location where combustible dusts are found, such as the application of powder paint, the following special precautions are required to insure that there are no combustible dusts inside the robot.

- Purge maintenance air should be maintained at all times, even when the robot power is off. This will insure that dust can not enter the robot.
- A purge cycle will not remove accumulated dusts. Therefore, if the robot is exposed to dust when maintenance air is not present, it will be necessary to remove the covers and clean out any accumulated dust. Do not energize the robot until you have performed the following steps.

1. Before covers are removed, the exterior of the robot should be cleaned to remove accumulated dust.
2. When cleaning and removing accumulated dust, either on the outside or inside of the robot, be sure to use methods appropriate for the type of dust that exists. Usually lint free rags dampened with water are acceptable. Do not use a vacuum cleaner to remove dust as it can generate static electricity and cause an explosion unless special precautions are taken.
3. Thoroughly clean the interior of the robot with a lint free rag to remove any accumulated dust.
4. When the dust has been removed, the covers must be replaced immediately.
5. Immediately after the covers are replaced, run a complete purge cycle. The robot can now be energized.

Staying Safe While Operating Paint Application Equipment

When you work with paint application equipment, observe the following rules, in addition to all rules for safe operation that apply to all robot systems.

WARNING

When working with electrostatic paint equipment, follow all national and local codes as well as all safety guidelines within your organization. Also reference the following standards: NFPA 33 Standards for Spray Application Using Flammable or Combustible Materials, and NFPA 70 National Electrical Code.

- **Grounding**: All electrically conductive objects in the spray area must be grounded. This includes the spray booth, robots, conveyors, workstations, part carriers, hooks, paint pressure pots, as well as solvent containers. Grounding is defined as the object or objects shall be electrically connected to ground with a resistance of not more than 1 megohms.
- **High Voltage**: High voltage should only be on during actual spray operations. Voltage should be off when the painting process is completed. Never leave high voltage on during a cap cleaning process.
- Avoid any accumulation of combustible vapors or coating matter.
- Follow all manufacturer recommended cleaning procedures.
- Make sure all interlocks are operational.

- No smoking.
- Post all warning signs regarding the electrostatic equipment and operation of electrostatic equipment according to NFPA 33 Standard for Spray Application Using Flammable or Combustible Material.
- Disable all air and paint pressure to bell.
- Verify that the lines are not under pressure.

Staying Safe During Maintenance

When you perform maintenance on the painter system, observe the following rules, and all other maintenance safety rules that apply to all robot installations. Only qualified, trained service or maintenance personnel should perform repair work on a robot.

- Paint robots operate in a potentially explosive environment. Use caution when working with electric tools.
- When a maintenance technician is repairing or adjusting a robot, the work area is under the control of that technician. All personnel not participating in the maintenance must stay out of the area.
- For some maintenance procedures, station a second person at the control panel within reach of the EMERGENCY STOP button. This person must understand the robot and associated potential hazards.
- Be sure all covers and inspection plates are in good repair and in place.
- Always return the robot to the ‘‘home’’ position before you disarm it.
- Never use machine power to aid in removing any component from the robot.
- During robot operations, be aware of the robot’s movements. Excess vibration, unusual sounds, and so forth, can alert you to potential problems.
- Whenever possible, turn off the main electrical disconnect before you clean the robot.
- When using vinyl resin observe the following:
  - Wear eye protection and protective gloves during application and removal.
  - Adequate ventilation is required. Overexposure could cause drowsiness or skin and eye irritation.
  - If there is contact with the skin, wash with water.
  - Follow the Original Equipment Manufacturer’s Material Safety Data Sheets.
- When using paint remover observe the following:
  - Eye protection, protective rubber gloves, boots, and apron are required during booth cleaning.
  - Adequate ventilation is required. Overexposure could cause drowsiness.
  - If there is contact with the skin or eyes, rinse with water for at least 15 minutes. Then seek medical attention as soon as possible.
  - Follow the Original Equipment Manufacturer’s Material Safety Data Sheets.

SAFETY PRECAUTIONS

Thank you for purchasing FANUC Robot.
This chapter describes the precautions which must be observed to ensure the safe use of the robot.
Before attempting to use the robot, be sure to read this chapter thoroughly.

Before using the functions related to robot operation, read the relevant operator's manual to become familiar with those functions.

If any description in this chapter differs from that in the other part of this manual, the description given in this chapter shall take precedence.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral devices installed in a work cell.
In addition, refer to the “FANUC Robot SAFETY HANDBOOK (B-80687EN)”.

1 WORKING PERSON

The personnel can be classified as follows.

Operator:
- Turns robot controller power ON/OFF
- Starts robot program from operator’s panel

Programmer or teaching operator:
- Operates the robot
- Teaches robot inside the safety fence

Maintenance engineer:
- Operates the robot
- Teaches robot inside the safety fence
- Maintenance (adjustment, replacement)

- An operator cannot work inside the safety fence.
- A programmer, teaching operator, and maintenance engineer can work inside the safety fence. The working activities inside the safety fence include lifting, setting, teaching, adjusting, maintenance, etc.
- To work inside the fence, the person must be trained on proper robot operation.

During the operation, programming, and maintenance of your robotic system, the programmer, teaching operator, and maintenance engineer should take additional care of their safety by using the following safety precautions.

- Use adequate clothing or uniforms during system operation
- Wear safety shoes
- Use helmet

2 DEFINITION OF WARNING, CAUTION AND NOTE

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 attempting to use the robots.

> ⚠**WARNING**
> Applied when there is a danger of the user being injured or when there is a danger of both the user being injured and the equipment being damaged if the approved procedure is not observed.

> ⚠**CAUTION**
> Applied when there is a danger of the equipment being damaged, if the approved procedure is not observed.

> **NOTE**
> Notes are used to indicate supplementary information other than Warnings and Cautions.

- Read this manual carefully, and store it in a sales place.

3 WORKING PERSON SAFETY

Working person safety is the primary safety consideration. Because it is very dangerous to enter the operating space of the robot during automatic operation, adequate safety precautions must be observed.
The following lists the general safety precautions. Careful consideration must be made to ensure working person safety.

(1) Have the robot system working persons attend the training courses held by FANUC.

> FANUC provides various training courses. Contact our sales office for details.

(2) Even when the robot is stationary, it is possible that the robot is still in a ready to move state, and is waiting for a signal. In this state, the robot is regarded as still in motion. To ensure working person safety, provide the system with an alarm to indicate visually or aurally that the robot is in motion.

(3) Install a safety fence with a gate so that no working person can enter the work area without passing through the gate. Install an interlocking device, a safety plug, and so forth in the safety gate so that the robot is stopped as the safety gate is opened.

> The controller is designed to receive this interlocking signal of the door switch. When the gate is opened and this signal received, the controller stops the robot (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type). For connection, see Fig.3 (a) and Fig.3 (b).

(4) Provide the peripheral devices with appropriate grounding (Class A, Class B, Class C, and Class D).

(5) Try to install the peripheral devices outside the work area.
(6) Draw an outline on the floor, clearly indicating the range of the robot motion, including the tools such as a hand.
(7) Install a mat switch or photoelectric switch on the floor with an interlock to a visual or aural alarm that stops the robot when a working person enters the work area.
(8) If necessary, install a safety lock so that no one except the working person in charge can turn on the power of the robot.

The circuit breaker installed in the controller is designed to disable anyone from turning it on when it is locked with a padlock.

(9) When adjusting each peripheral device independently, be sure to turn off the power of the robot
(10) Operators should be ungloved while manipulating the operator's panel or teach pendant. Operation with gloved fingers could cause an operation error.
(11) Programs, system variables, and other information can be saved on memory card or USB memories. Be sure to save the data periodically in case the data is lost in an accident.
(12) The robot should be transported and installed by accurately following the procedures recommended by FANUC. Wrong transportation or installation may cause the robot to fall, resulting in severe injury to workers.
(13) In the first operation of the robot after installation, the operation should be restricted to low speeds. Then, the speed should be gradually increased to check the operation of the robot.
(14) Before the robot is started, it should be checked that no one is in the area of the safety fence. At the same time, a check must be made to ensure that there is no risk of hazardous situations. If detected, such a situation should be eliminated before the operation.
(15) When the robot is used, the following precautions should be taken. Otherwise, the robot and peripheral equipment can be adversely affected, or workers can be severely injured.
- Avoid using the robot in a flammable environment.
- Avoid using the robot in an explosive environment.
- Avoid using the robot in an environment full of radiation.
- Avoid using the robot under water or at high humidity.
- Avoid using the robot to carry a person or animal.
- Avoid using the robot as a stepladder. (Never climb up on or hang from the robot.)
(16) When connecting the peripheral devices related to stop(safety fence etc.) and each signal (external emergency , fence etc.) of robot. be sure to confirm the stop movement and do not take the wrong connection.
(17) When preparing trestle, please consider security for installation and maintenance work in high place according to Fig.3 (c). Please consider footstep and safety bolt mounting position.

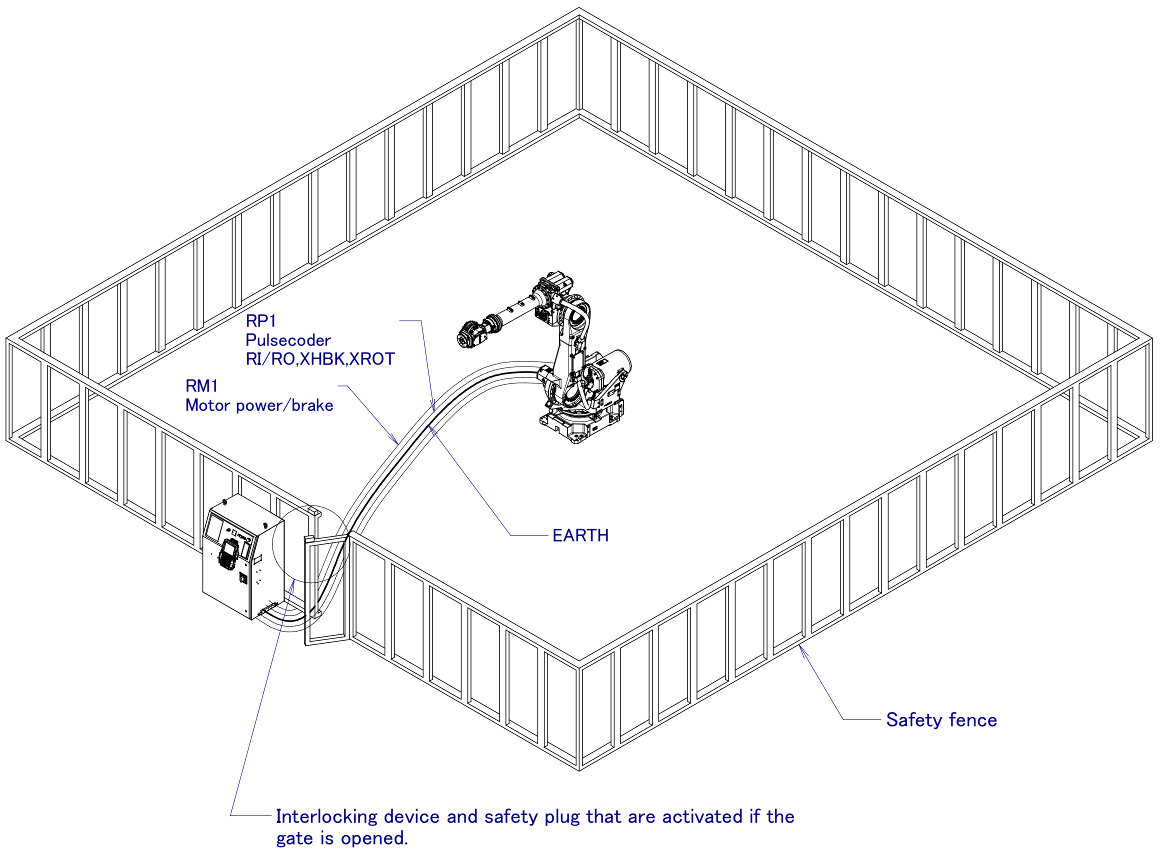
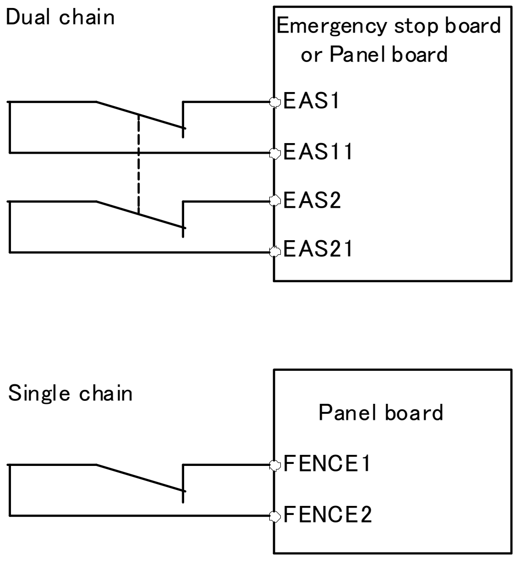

RP1
Pulsecoder
RI/RO,XHBK,XROT
RM1
Motor power/brake
EARTH
Safety fence
Interlocking device and safety plug that are activated if the gate is opened.

Fig. 3 (a) Safety fence and safety gate



(Note)

In case of R-30*i*B, R-30*i*B Mate
Terminals EAS1,EAS11,EAS2,EAS21 are provided on the emergency stop board.

Refer to the ELECTRICAL CONNCETIONS Chapter of CONNECTION of
R-30*i*B controller maintenance manual (B-83195EN) or
R-30*i*B Mate controller maintenance manual (B-83525EN)
for details.

Fig. 3 (b) Limit switch circuit diagram of the safety fence

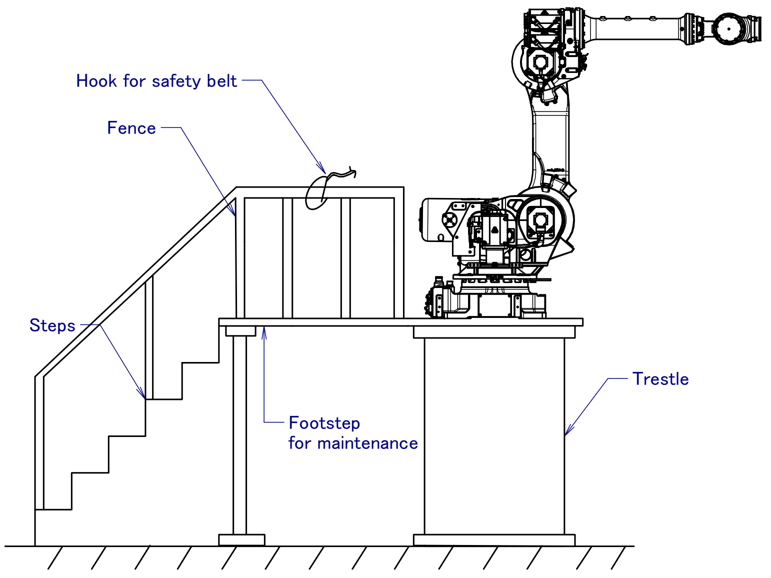


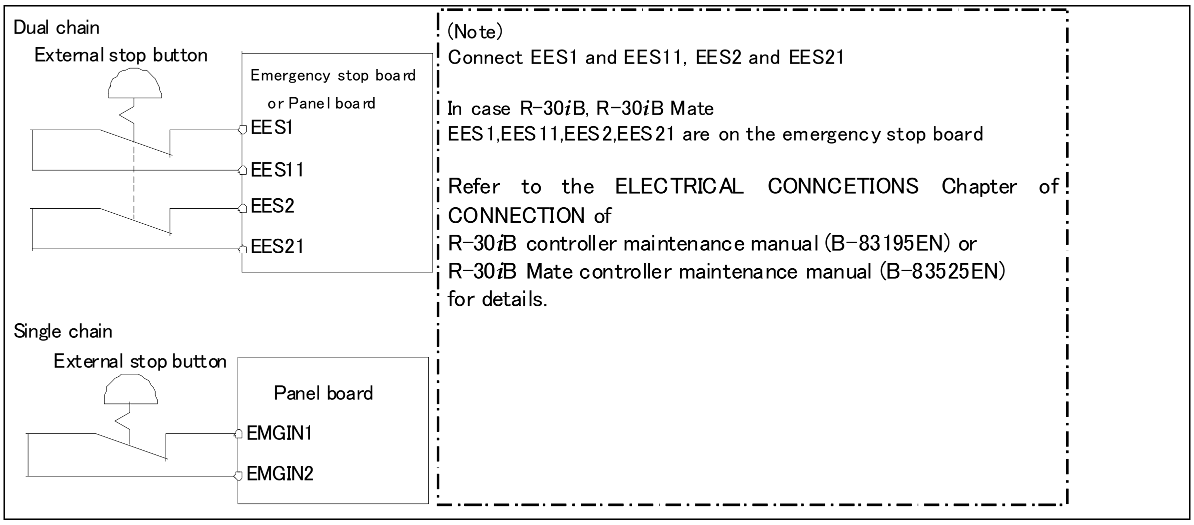
Fig.3 (c) Footstep for maintenance

3.1 OPERATOR SAFETY

The operator is a person who operates the robot system. In this sense, a worker who operates the teach pendant is also an operator. However, this section does not apply to teach pendant operators.

(1) If you do not have to operate the robot, turn off the power of the robot controller or press the EMERGENCY STOP button, and then proceed with necessary work.
(2) Operate the robot system at a location outside of the safety fence
(3) Install a safety fence with a safety gate to prevent any worker other than the operator from entering the work area unexpectedly and to prevent the worker from entering a dangerous area.
(4) Install an EMERGENCY STOP button within the operator's reach.

The robot controller is designed to be connected to an external EMERGENCY STOP button. With this connection, the controller stops the robot operation (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type), when the external EMERGENCY STOP button is pressed. See the diagram below for connection.



(Note)
Connect EES1 and EES11, EES2 and EES21

In case R-30*i*B, R-30*i*B Mate
EES1,EES11,EES2,EES21 are on the emergency stop board

Refer to the ELECTRICAL CONNCETIONS Chapter of CONNECTION of
R-30*i*B controller maintenance manual (B-83195EN) or
R-30*i*B Mate controller maintenance manual (B-83525EN)
for details.

Fig.3.1 Connection diagram for external emergency stop button

3.2 SAFETY OF THE PROGRAMMER

While teaching the robot, the operator must enter the work area of the robot. The operator must ensure the safety of the teach pendant operator especially.

(1) Unless it is specifically necessary to enter the robot work area, carry out all tasks outside the area.
(2) Before teaching the robot, check that the robot and its peripheral devices are all in the normal operating condition.
(3) If it is inevitable to enter the robot work area to teach the robot, check the locations, settings, and other conditions of the safety devices (such as the EMERGENCY STOP button, the DEADMAN switch on the teach pendant) before entering the area.
(4) The programmer must be extremely careful not to let anyone else enter the robot work area.
(5) Programming should be done outside the area of the safety fence as far as possible. If programming needs to be done in the area of the safety fence, the programmer should take the following precautions:
   - Before entering the area of the safety fence, ensure that there is no risk of dangerous situations in the area.
   - Be prepared to press the emergency stop button whenever necessary.
   - Robot motions should be made at low speeds.
   - Before starting programming, check the entire system status to ensure that no remote instruction to the peripheral equipment or motion would be dangerous to the user.

Our operator panel is provided with an emergency stop button and a key switch (mode switch) for selecting the automatic operation mode (AUTO) and the teach modes (T1 and T2). Before entering the inside of the safety fence for the purpose of teaching, set the switch to a teach mode, remove the key from the mode switch to prevent other people from changing the operation mode carelessly, then open the safety gate. If the safety gate is opened with the automatic operation mode set, the robot stops (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type). After the switch is set to a teach mode, the safety gate is disabled. The programmer should understand that the safety gate is disabled and is responsible for keeping other people from entering the inside of the safety fence.

Our teach pendant is provided with a DEADMAN switch as well as an emergency stop button. These button and switch function as follows:
(1) Emergency stop button: Causes an emergency stop (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type) when pressed.
(2) DEADMAN switch: Functions differently depending on the teach pendant enable/disable switch setting status.
   (a) Disable: The DEADMAN switch is disabled.
   (b) Enable: Servo power is turned off when the operator releases the DEADMAN switch or when the operator presses the switch strongly.
   Note) The DEADMAN switch is provided to stop the robot when the operator releases the teach pendant or presses the pendant strongly in case of emergency. The R-30*i*B/R-30*i*B Mate employs a 3-position DEADMAN switch, which allows the robot to operate when the 3-position DEADMAN switch is pressed to its intermediate point. When the operator releases the DEADMAN switch or presses the switch strongly, the robot stops immediately.

The operator's intention of starting teaching is determined by the controller through the dual operation of setting the teach pendant enable/disable switch to the enable position and pressing the DEADMAN switch. The operator should make sure that the robot could operate in such conditions and be responsible in carrying out tasks safely.

Based on the risk assessment by FANUC, number of operation of DEADMAN SW should not exceed about 10000 times per year.

The teach pendant, operator panel, and peripheral device interface send each robot start signal. However the validity of each signal changes as follows depending on the mode switch and the DEADMAN switch of the operator panel, the teach pendant enable switch and the remote condition on the software.

In case of R-30*i*B Controller

Mode	Teach pendant enable switch	Software remote condition	Teach pendant	Operator panel	Peripheral device
AUTO mode	On	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed
	Off	Local	Not allowed	Allowed to start	Not allowed
		Remote	Not allowed	Not allowed	Allowed to start
T1, T2 mode	On	Local	Allowed to start	Not allowed	Not allowed
		Remote	Allowed to start	Not allowed	Not allowed
	Off	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed

T1,T2 mode:DEADMAN switch is effective.

(6) To start the system using the operator's panel, make certain that nobody is the robot work area and that there are no abnormal conditions in the robot work area.
(7) When a program is completed, be sure to carry out a test operation according to the procedure below.
  (a) Run the program for at least one operation cycle in the single step mode at low speed.
  (b) Run the program for at least one operation cycle in the continuous operation mode at low speed.
  (c) Run the program for one operation cycle in the continuous operation mode at the intermediate speed and check that no abnormalities occur due to a delay in timing.
  (d) Run the program for one operation cycle in the continuous operation mode at the normal operating speed and check that the system operates automatically without trouble.
  (e) After checking the completeness of the program through the test operation above, execute it in the automatic operation mode.
(8) While operating the system in the automatic operation mode, the teach pendant operator should leave the robot work area.

3.3 SAFETY OF THE MAINTENANCE ENGINEER

For the safety of maintenance engineer personnel, pay utmost attention to the following.

(1) During operation, never enter the robot work area.
(2) A hazardous situation may arise when the robot or the system, are kept with their power-on during maintenance operations. Therefore, for any maintenance operation, the robot and the system should be put into the power-off state. If necessary, a lock should be in place in order to prevent any other person from turning on the robot and/or the system. In case maintenance needs to be executed in the power-on state, the emergency stop button must be pressed.
(3) If it becomes necessary to enter the robot operation range while the power is on, press the emergency stop button on the operator panel, or the teach pendant before entering the range. The maintenance personnel must indicate that maintenance work is in progress and be careful not to allow other people to operate the robot carelessly.
(4) When entering the area enclosed by the safety fence, the maintenance worker must check the entire system in order to make sure no dangerous situations exist. In case the worker needs to enter the safety area whilst a dangerous situation exists, extreme care must be taken, and entire system status must be carefully monitored.
(5) Before the maintenance of the pneumatic system is started, the supply pressure should be shut off and the pressure in the piping should be reduced to zero.

(6) Before the start of teaching, check that the robot and its peripheral devices are all in the normal operating condition.
(7) Do not operate the robot in the automatic mode while anybody is in the robot work area.
(8) When you maintain the robot alongside a wall or instrument, or when multiple workers are working nearby, make certain that their escape path is not obstructed.
(9) When a tool is mounted on the robot, or when any moving device other than the robot is installed, such as belt conveyor, pay careful attention to its motion.
(10) If necessary, have a worker who is familiar with the robot system stand beside the operator panel and observe the work being performed. If any danger arises, the worker should be ready to press the EMERGENCY STOP button at any time.
(11) When replacing a part, please contact FANUC service center. If a wrong procedure is followed, an accident may occur, causing damage to the robot and injury to the worker.
(12) When replacing or reinstalling components, take care to prevent foreign material from entering the system.
(13) When handling each unit or printed circuit board in the controller during inspection, turn off the circuit breaker to protect against electric shock.
If there are two cabinets, turn off the both circuit breaker.
(14) A part should be replaced with a part recommended by FANUC. If other parts are used, malfunction or damage would occur. Especially, a fuse that is not recommended by FANUC should not be used. Such a fuse may cause a fire.
(15) When restarting the robot system after completing maintenance work, make sure in advance that there is no person in the work area and that the robot and the peripheral devices are not abnormal.
(16) When a motor or brake is removed, the robot arm should be supported with a crane or other equipment beforehand so that the arm would not fall during the removal.
(17) Whenever grease is spilled on the floor, it should be removed as quickly as possible to prevent dangerous falls.
(18) The following parts are heated. If a maintenance worker needs to touch such a part in the heated state, the worker should wear heat-resistant gloves or use other protective tools.
- Servo motor
- Inside the controller
- Reducer
- Gearbox
- Wrist unit

(19) Maintenance should be done under suitable light. Care must be taken that the light would not cause any danger.
(20) When a motor, reducer, or other heavy load is handled, a crane or other equipment should be used to protect maintenance workers from excessive load. Otherwise, the maintenance workers would be severely injured.
(21) The robot should not be stepped on or climbed up during maintenance. If it is attempted, the robot would be adversely affected. In addition, a misstep can cause injury to the worker.
(22)When performing maintenance work in high place, secure a footstep and wear safety belt.
(23) After the maintenance is completed, spilled oil or water and metal chips should be removed from the floor around the robot and within the safety fence.
(24) When a part is replaced, all bolts and other related components should put back into their original places. A careful check must be given to ensure that no components are missing or left not mounted.
(25) In case robot motion is required during maintenance, the following precautions should be taken :
- Foresee an escape route. And during the maintenance motion itself, monitor continuously the whole system so that your escape route will not become blocked by the robot, or by peripheral equipment.
- Always pay attention to potentially dangerous situations, and be prepared to press the emergency stop button whenever necessary.

(26) The robot should be periodically inspected. (Refer to the robot mechanical manual and controller maintenance manual.) A failure to do the periodical inspection can adversely affect the performance or service life of the robot and may cause an accident

(27) After a part is replaced, a test operation should be given for the robot according to a predetermined method. (See TESTING section of “R-30iB/R-30iB Mate Controller operator’s manual (Basic Operation)”.) During the test operation, the maintenance staff should work outside the safety fence.

4 SAFETY OF THE TOOLS AND PERIPHERAL DEVICES

4.1 PRECAUTIONS IN PROGRAMMING

(1) Use a limit switch or other sensor to detect a dangerous condition and, if necessary, design the program to stop the robot when the sensor signal is received.
(2) Design the program to stop the robot when an abnormal condition occurs in any other robots or peripheral devices, even though the robot itself is normal.
(3) For a system in which the robot and its peripheral devices are in synchronous motion, particular care must be taken in programming so that they do not interfere with each other.
(4) Provide a suitable interface between the robot and its peripheral devices so that the robot can detect the states of all devices in the system and can be stopped according to the states.

4.2 PRECAUTIONS FOR MECHANISM

(1) Keep the component cells of the robot system clean, and operate the robot in an environment free of grease, water, and dust.
(2) Don’t use unconfirmed liquid for cutting fluid and cleaning fluid.
(3) Employ a limit switch or mechanical stopper to limit the robot motion so that the robot or cable does not strike against its peripheral devices or tools.
(4) Observe the following precautions about the mechanical unit cables. Failure to follow precautions may cause mechanical troubles.
   - Use mechanical unit cable that have required user interface.
   - Don’t add user cable or hose to inside of mechanical unit.
   - Please do not obstruct the movement of the mechanical unit cable when cables are added to outside of mechanical unit.
   - In the case of the model that a cable is exposed, Please do not perform remodeling (Adding a protective cover and fix an outside cable more) obstructing the behavior of the outcrop of the cable.
   - When installing user peripheral equipment on the robot mechanical unit, please pay attention that equipment does not interfere with the robot itself.

(5) The frequent power-off stop for the robot during operation causes the trouble of the robot. Please avoid the system construction that power-off stop would be operated routinely. (Refer to bad case example.) Please execute power-off stop after reducing the speed of the robot and stopping it by hold stop or cycle stop when it is not urgent. (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type.)
(Bad case example)
   - Whenever poor product is generated, a line stops by emergency stop.
   - When alteration was necessary, safety switch is operated by opening safety fence and power-off stop is executed for the robot during operation.
   - An operator pushes the emergency stop button frequently, and a line stops.
   - An area sensor or a mat switch connected to safety signal operate routinely and power-off stop is executed for the robot.

(6) Robot stops urgently when collision detection alarm (SRVO-050) etc. occurs. Please try to avoid unnecessary power-off stops. It may cause the trouble of the robot, too. So remove the causes of the alarm.

5 SAFETY OF THE ROBOT MECHANISM

5.1 PRECAUTIONS IN OPERATION

(1) When operating the robot in the jog mode, set it at an appropriate speed so that the operator can manage the robot in any eventuality.
(2) Before pressing the jog key, be sure you know in advance what motion the robot will perform in the jog mode.

5.2 PRECAUTIONS IN PROGRAMMING

(1) When the work areas of robots overlap, make certain that the motions of the robots do not interfere with each other.
(2) Be sure to specify the predetermined work origin in a motion program for the robot and program the motion so that it starts from the origin and terminates at the origin.
Make it possible for the operator to easily distinguish at a glance that the robot motion has terminated.

5.3 PRECAUTIONS FOR MECHANISMS

(1) Keep the work areas of the robot clean, and operate the robot in an environment free of grease, water, and dust.

5.4 PROCEDURE TO MOVE ARM WITHOUT DRIVE POWER IN EMERGENCY OR ABNORMAL SITUATIONS

For emergency or abnormal situations (e.g. persons trapped in or by the robot), brake release unit can be used to move the robot axes without drive power.
Please refer to controller maintenance manual and mechanical unit operator’s manual for using method of brake release unit and method of supporting robot.

6 SAFETY OF THE END EFFECTOR

6.1 PRECAUTIONS IN PROGRAMMING

(1) To control the pneumatic, hydraulic and electric actuators, carefully consider the necessary time delay after issuing each control command up to actual motion and ensure safe control.
(2) Provide the end effector with a limit switch, and control the robot system by monitoring the state of the end effector.

7 STOP TYPE OF ROBOT

The following three robot stop types exist:

Power-Off Stop (Category 0 following IEC 60204-1)

Servo power is turned off and the robot stops immediately. Servo power is turned off when the robot is moving, and the motion path of the deceleration is uncontrolled.
The following processing is performed at Power-Off stop.
- An alarm is generated and servo power is turned off.
- The robot operation is stopped immediately. Execution of the program is paused.

Controlled stop (Category 1 following IEC 60204-1)

The robot is decelerated until it stops, and servo power is turned off.
The following processing is performed at Controlled stop.
- The alarm "SRVO-199 Controlled stop" occurs along with a decelerated stop. Execution of the program is paused.
- An alarm is generated and servo power is turned off.

Hold (Category 2 following IEC 60204-1)

The robot is decelerated until it stops, and servo power remains on.
The following processing is performed at Hold.
- The robot operation is decelerated until it stops. Execution of the program is paused.

⚠ **WARNING**
The stopping distance and stopping time of Controlled stop are longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the increased stopping distance and stopping time, is necessary when Controlled stop is used.

When the emergency stop button is pressed or the FENCE is open, the stop type of robot is Power-Off stop or Controlled stop. The configuration of stop type for each situation is called *stop pattern*. The stop pattern is different according to the controller type or option configuration.

There are the following 3 Stop patterns.

Stop pattern	Mode	Emergency stop button	External Emergency stop	FENCE open	SVOFF input	Servo disconnect
A	AUTO	P-Stop	P-Stop	C-Stop	C-Stop	P-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop
B	AUTO	P-Stop	P-Stop	P-Stop	P-Stop	P-Stop
	T1	P-Stop	P-Stop	-	P-Stop	P-Stop
	T2	P-Stop	P-Stop	-	P-Stop	P-Stop
C	AUTO	C-Stop	C-Stop	C-Stop	C-Stop	C-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop

P-Stop:	Power-Off stop
C-Stop:	Controlled stop
-:	Disable

The following table indicates the Stop pattern according to the controller type or option configuration.

Option	R-30*i*B/ R-30*i*B Mate
Standard	A (*)
Controlled stop by E-Stop (A05B-2600-J570)	C (*)

(*) R-30*i*B / R-30*i*B Mate does not have servo disconnect. / R-30*i*B Mate does not have SVOFF input.

The stop pattern of the controller is displayed in "Stop pattern" line in software version screen. Please refer to "Software version" in operator's manual of controller for the detail of software version screen.

"Controlled stop by E-Stop" option

When "Controlled stop by E-Stop" (A05B-2600-J570) option is specified, the stop type of the following alarms becomes
Controlled stop but only in AUTO mode. In T1 or T2 mode, the stop type is Power-Off stop which is the normal operation of the system.

Alarm	Condition
SRVO-001 Operator panel E-stop	Operator panel emergency stop is pressed.
SRVO-002 Teach pendant E-stop	Teach pendant emergency stop is pressed.
SRVO-007 External emergency stops	External emergency stop input (EES1-EES11, EES2-EES21) is open.
SRVO-408 DCS SSO Ext Emergency Stop	In DCS Safe I/O connect function, SSO[3] is OFF.
SRVO-409 DCS SSO Servo Disconnect	In DCS Safe I/O connect function, SSO[4] is OFF.

Controlled stop is different from Power-Off stop as follows:

- In Controlled stop, the robot is stopped on the program path. This function is effective for a system where the robot can interfere with other devices if it deviates from the program path.
- In Controlled stop, physical impact is less than Power-Off stop. This function is effective for systems where the physical impact to the mechanical unit or EOAT (End Of Arm Tool) should be minimized.
- The stopping distance and stopping time of Controlled stop is longer than the stopping distance and stopping time of Power-Off stop, depending on the robot model and axis. Please refer to the operator's manual of a particular robot model for the data of stopping distance and stopping time.

When this option is loaded, this function cannot be disabled.

The stop type of DCS Position and Speed Check functions is not affected by the loading of this option.

⚠ **WARNING**

The stopping distance and stopping time of Controlled stop are longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the increased stopping distance and stopping time, is necessary when this option is loaded.

130828

TABLE OF CONTENTS

1 OVERVIEW

System Design Tool is the software option (J738). It is package of useful functions to construct production system that uses robot. KAREL programs mainly realize these functions. You can select to load KAREL programs which you need to the robot controller. It has following functions.

Refer to the section "Overview of KAREL" of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN).

There are 6 categories in System Design Tool.

Category	Description
Trigonometric Function	Calculating trigonometric function by TP programs.
Robot Setup Assistance	Copying the configuration of virtual robot set up on ROBOGUIDE to actual robot. Easily setting the comment of I/O, register and so on with PC.
TP Program Memory Assist	Loading from or saving to external device or deleting TP programs according to specified file list. It is helpful in case of lack of memory in the robot controller.
Sim Check/Unsim I/O	Checking I/O simulated status. Unsimulating I/O by TP programs.
Status Check/Set Function	Checking various status of the robot such as single step, group motion (machine lock) and so on. Setting various status of the robot such as single step.
Others	Useful functions are included for articulation of the robot system.

“Others” includes following.

Function	Description
User Specified Alarm Post	Displaying alarms with message depending on each system by TP programs.
Get Date, Time	Getting date or time and output them to registers.
Set Language	Switching the language, English or Japanese.
Set Program	Selecting TP program set in the string register.

KAREL is the robot language for the robot system architecture. User’s original function can be created after creating KAREL program on PC, loading it to the robot controller and executing it. Refer to KAREL Function, B-83144, to know the detail about KAREL programs.

NOTE

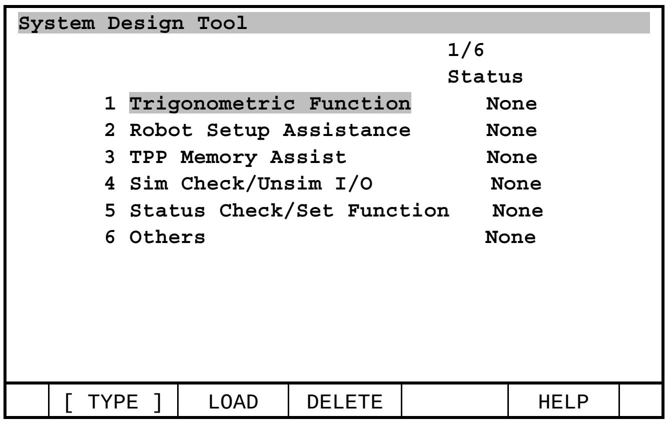
System Design Tool software option must be installed to the robot controller to use KAREL programs of its option. Otherwise, one of following warning is displayed.
SYST - 298 Trigonometric Function not installed
SYST - 299 Robot Setup Assistance not installed
SYST - 300 System Design Tool not installed

2 HOW TO SET UP

This chapter describes setting up of System Design Tool.

2.1 SYSTEM DESIGN TOOL SETUP SCREEN

Select [MENU] -> “6. SETUP” -> F1[TYPE] -> Sys Dsn Tool. Following screen, top screen, is displayed.

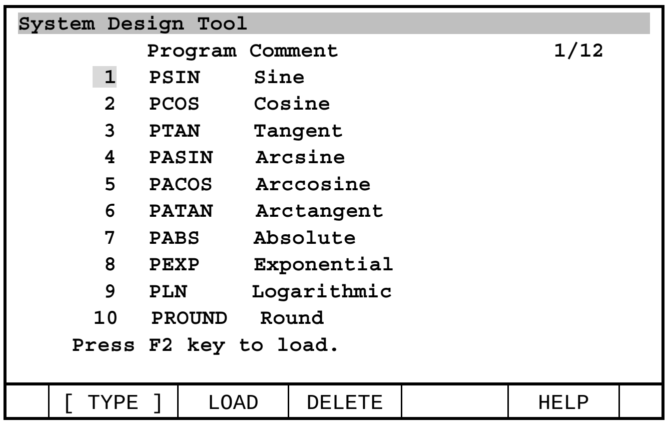


```
System Design Tool
                                  1/6
                                  Status
      1 Trigonometric Function      None
      2 Robot Setup Assistance      None
      3 TPP Memory Assist           None
      4 Sim Check/Unsim I/O         None
      5 Status Check/Set Function   None
      6 Others                      None

  [ TYPE ]   LOAD   DELETE          HELP
```

Each category and the load status of KAREL program of it are displayed.

Status	Description
Complete	All KAREL programs are loaded.
Partial	A part of KAREL programs are loaded.
None	None of KAREL programs is loaded.

To show KAREL programs of each category, set the cursor onto it and press the ENTER key. Detail screen is shown. Following is the sample of detail screen in the case of Trigonometric Function.



```
System Design Tool
         Program Comment               1/12
      1   PSIN    Sine
      2   PCOS    Cosine
      3   PTAN    Tangent
      4   PASIN   Arcsine
      5   PACOS   Arccosine
      6   PATAN   Arctangent
      7   PABS    Absolute
      8   PEXP    Exponential
      9   PLN     Logarithmic
     10   PROUND  Round
    Press F2 key to load.

  [ TYPE ]   LOAD   DELETE          HELP
```

The "Comment" of above screen shows the function of each KAREL program.

2.2 LOAD KAREL PROGRAMS OF SYSTEM DESIGN TOOL

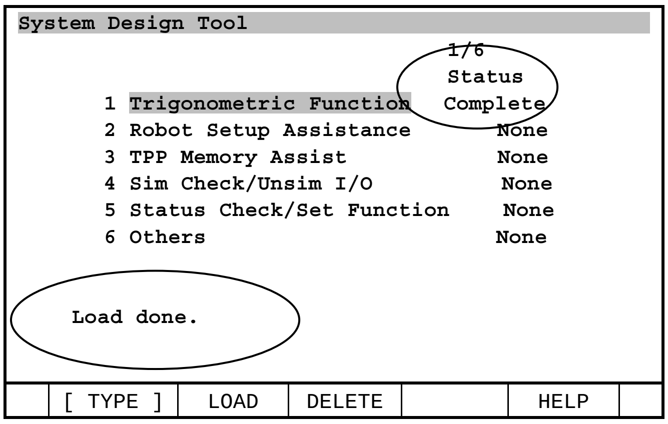
KAREL programs of System Design Tool are not loaded as of shipment time for preventing to waste TP program memory by unnecessary KAREL program. Then they should be loaded as necessary. This section describes how to load them. KAREL programs to be loaded are saved in FROM.

NOTE

KAREL program loading fails by the following cases. Please check the following before loading.
・Same TP program name exists before loading KAREL program.
・Status of KAREL program trying to load is selected, paused or running.
In the case some KAREL programs are not loaded, "SDTL-022 Failed to load %s." is displayed (%s is KAREL program name). Refer to subsection 2.2.3 to know the remedy.

2.2.1 Loading All

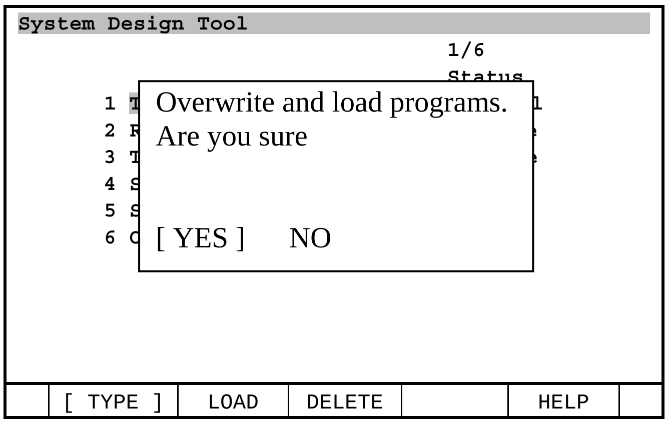
Set the cursor to each category. Press F2[LOAD], all KAREL programs in the category are loaded. The status is turned to "Complete" as follows after loading is completely done. "Load done." is displayed in prompt line.



```
System Design Tool
                                    1/6
                                   Status
      1 Trigonometric Function   Complete
      2 Robot Setup Assistance     None
      3 TPP Memory Assist          None
      4 Sim Check/Unsim I/O        None
      5 Status Check/Set Function  None
      6 Others                     None


  Load done.

 [ TYPE ]  LOAD  DELETE        HELP
```

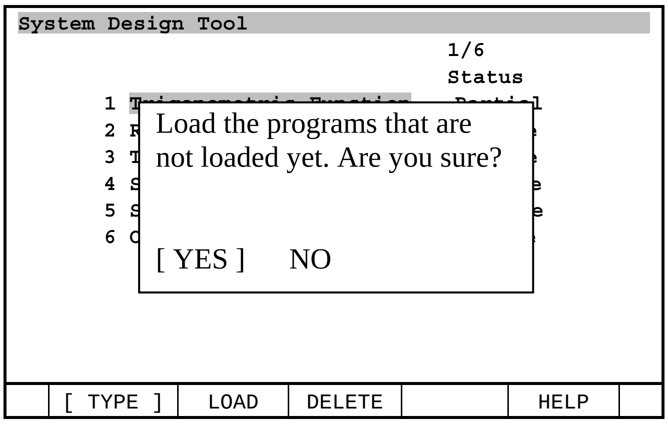
If a KAREL programs is not loaded, "SDTL-022 Failed to load %s." is displayed (%s is KAREL program name).

Confirm message is displayed when F2[LOAD] is pressed in the "Partial" status.



```
System Design Tool
                                    1/6
                                   Status
      1 Overwrite and load programs.
      2 Are you sure
      3
      4
      5
      6 [ YES ]   NO

 [ TYPE ]  LOAD  DELETE        HELP
```

Set the cursor onto YES and press the ENTER key to overwrite and load the KAREL programs. If "NO" is selected following is display ed.



```
System Design Tool
                                   1/6
                                   Status
      1 T                           l
      2 R                           e
      3 T                           e
      4 S                           e
      5 S                           e
      6 C
```

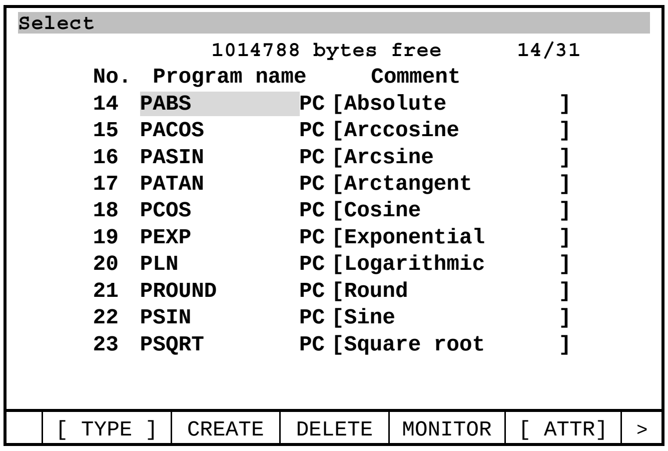
Load the programs that are not loaded yet. Are you sure?

[YES] NO

```
 [ TYPE ] | LOAD | DELETE |      | HELP
```

To load programs that are not loaded, set the cursor onto YES and press the ENTER key.
None of programs are loaded if NO is selected.

Loaded KAREL programs, extension .PC, are displayed in SELECT screen. Press F1[TYPE] -> "All" or "KAREL progs".



```
Select
               1014788 bytes free      14/31
      No.  Program name     Comment
      14  PABS         PC [Absolute         ]
      15  PACOS        PC [Arccosine        ]
      16  PASIN        PC [Arcsine          ]
      17  PATAN        PC [Arctangent       ]
      18  PCOS         PC [Cosine           ]
      19  PEXP         PC [Exponential      ]
      20  PLN          PC [Logarithmic      ]
      21  PROUND       PC [Round            ]
      22  PSIN         PC [Sine             ]
      23  PSQRT        PC [Square root      ]

 [ TYPE ] | CREATE | DELETE | MONITOR | [ ATTR] | >
```

If these are not displayed, set system variable $KAREL_ENB = 1.
Refer to the Section "Overview of KAREL" of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation) (B-83284EN).

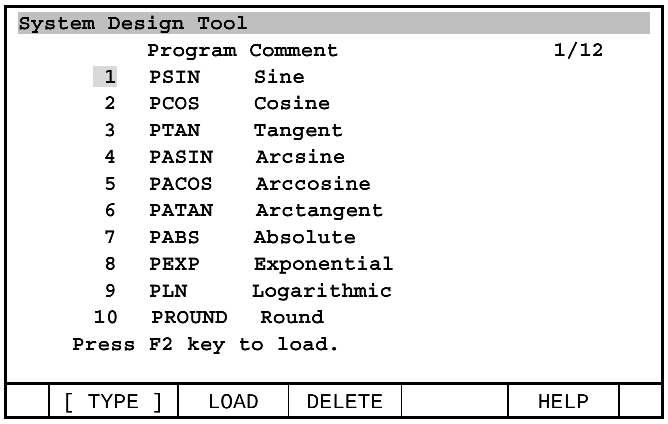
2.2.2 Loading Individually

This subsection explains to load KAREL programs individually used trigonometric function as a sample.

NOTE

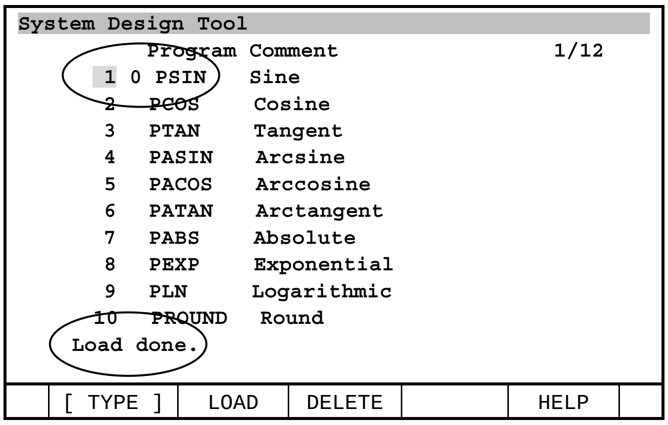
Don't load a program of Robot Setup Assistance individually. Load programs all at once. If you accidentally load a program of Robot Setup Assistance individually, delete all programs first. Then load them all at once.

Detail screen is displayed after setting the cursor onto trigonometric function in the above screen and pressing the ENTER key.



```
System Design Tool
         Program Comment                1/12
      1   PSIN    Sine
      2   PCOS    Cosine
      3   PTAN    Tangent
      4   PASIN   Arcsine
      5   PACOS   Arccosine
      6   PATAN   Arctangent
      7   PABS    Absolute
      8   PEXP    Exponential
      9   PLN     Logarithmic
     10   PROUND  Round
    Press F2 key to load.

   [ TYPE ]   LOAD   DELETE         HELP
```

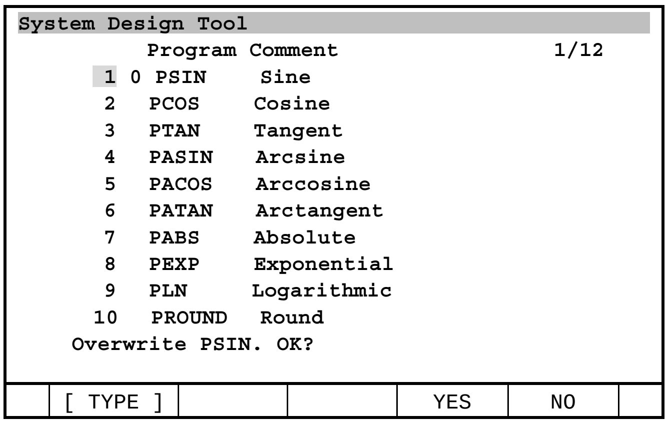
Set the cursor onto the index number of KAREL program and press F2[LOAD].



```
System Design Tool
         Program Comment                1/12
      1 0 PSIN   Sine
      2   PCOS    Cosine
      3   PTAN    Tangent
      4   PASIN   Arcsine
      5   PACOS   Arccosine
      6   PATAN   Arctangent
      7   PABS    Absolute
      8   PEXP    Exponential
      9   PLN     Logarithmic
     10   PROUND  Round
    Load done.

   [ TYPE ]   LOAD   DELETE         HELP
```

"Load done." is displayed on the prompt line and "0" is displayed next to loaded KAREL program.

Confirm message to overwrite is displayed if being loaded KAREL program is already loaded. Select F4[YES] to overwrite, or F5[NO] to not overwrite.



```
System Design Tool
         Program Comment                1/12
      1 0 PSIN    Sine
      2   PCOS    Cosine
      3   PTAN    Tangent
      4   PASIN   Arcsine
      5   PACOS   Arccosine
      6   PATAN   Arctangent
      7   PABS    Absolute
      8   PEXP    Exponential
      9   PLN     Logarithmic
     10   PROUND  Round
    Overwrite PSIN. OK?

   [ TYPE ]                  YES     NO
```

2.2.3 Remedy of Failed to Load

Loading KAREL Program is Running, Paused or Selected

Confirm whether it is OK to abort TP program or not. If it is OK, abort the KAREL program. Press FCTN key and select “ABORT (ALL)”.
Select other TP or KAREL program in SELECT screen.

Same Name TP Program Existed

Display SELECT screen. (Press SELECT key or MENU -> 0-- NEXT -- -> SELECT.)
In the case the same name TP program is already existed, KAREL program is never loaded.
Change the TP program name or delete it after backup to MC:.

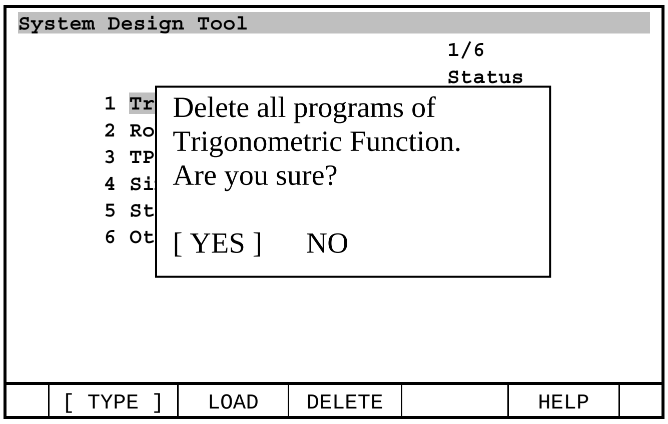
2.3 DELETE KAREL PROGRAMS OF SYSTEM DESIGN TOOL

This section explains how to delete KAREL programs of System Design Tool. It is allowed to delete the similar way of "LOAD KAREL PROGRAMS OF SYSTEM DESIGN TOOL" as described previously. Trigonometric function is used as a sample to delete them.

NOTE
If KAREL program intended to be deleted is selected, running or paused, delete operations fails. In the case some KAREL programs are not deleted, "SDTL-026 Failed to delete %s" is displayed (%s is KAREL program name). Preliminarily verify the program running status.
Select other TP or KAREL program in case to delete the now selected KAREL program.

2.3.1 Delete All

Set the cursor onto "Trigonometric Function". Press F3[DELETE], then following message is displayed.

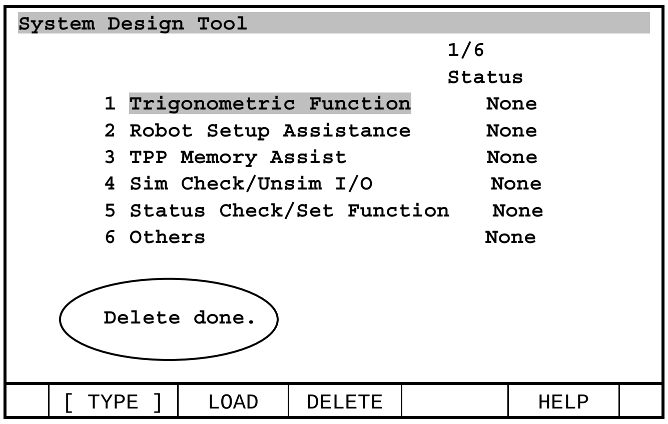


```
System Design Tool
                                  1/6
                                  Status
       1 Tr
       2 Ro
       3 TP
       4 Si
       5 St
       6 Ot
```

Delete all programs of
Trigonometric Function.
Are you sure?

[YES]   NO

Set the cursor to [YES] and press the ENTER key.
After deleting from the robot controller completely "Delete done." is displayed.



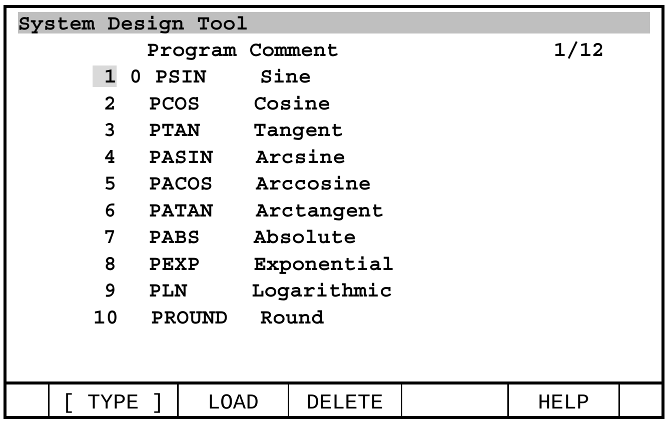
```
System Design Tool
                                  1/6
                                  Status
       1 Trigonometric Function      None
       2 Robot Setup Assistance      None
       3 TPP Memory Assist           None
       4 Sim Check/Unsim I/O         None
       5 Status Check/Set Function   None
       6 Others                      None

    Delete done.
```

2.3.2 Delete Individually

This subsection explains to delete KAREL programs individually.

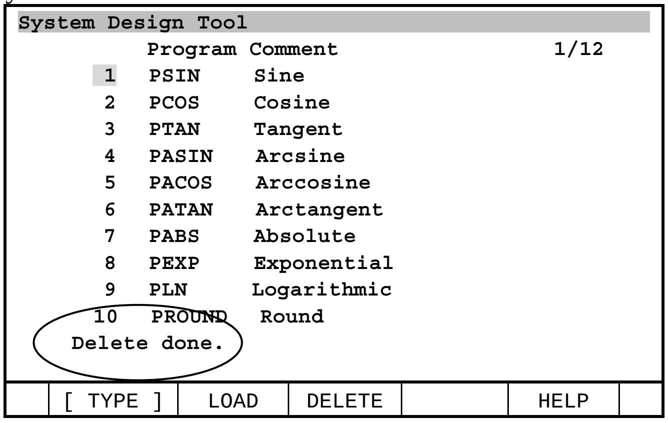
NOTE
Don’t delete a program of Robot Setup Assistance individually. Delete programs all at once. If you accidentally delete a program of Robot Setup Assistance individually, delete all programs first. Then load them all at once.

Set the cursor onto the program number and press F3[DELETE].



```
System Design Tool
          Program Comment                 1/12
      1 0 PSIN     Sine
      2   PCOS     Cosine
      3   PTAN     Tangent
      4   PASIN    Arcsine
      5   PACOS    Arccosine
      6   PATAN    Arctangent
      7   PABS     Absolute
      8   PEXP     Exponential
      9   PLN      Logarithmic
     10   PROUND   Round

   [ TYPE ]   LOAD   DELETE            HELP
```

"Delete done." is displayed.



```
System Design Tool
          Program Comment                 1/12
      1   PSIN     Sine
      2   PCOS     Cosine
      3   PTAN     Tangent
      4   PASIN    Arcsine
      5   PACOS    Arccosine
      6   PATAN    Arctangent
      7   PABS     Absolute
      8   PEXP     Exponential
      9   PLN      Logarithmic
     10   PROUND   Round
   Delete done.
   [ TYPE ]   LOAD   DELETE            HELP
```

2.3.3 Remedy of Failed to Delete

KAREL Program is Running, Paused or Selected

Confirm whether it is OK to abort TP program or not. If it is OK, abort the KAREL program. Press FCTN key and select “ABORT (ALL)”.
Select other TP or KAREL program in SELECT screen.

2.3.4 NOTE

Never to delete the KAREL programs of System Design Tool from SELECT screen. It must be deleted from System Design Tool screen described previously.

Once it is deleted from SELECT screen after loading from System Design Tool screen, it is loaded at the timing of cycle power again automatically. Delete it according to the procedure described above.

In the case that top screen of System Design Tool and its detail screen is displayed after once the KAREL programs are deleted from SELECT screen, the status is not correctly displayed. At top screen it is not allowed to indicate the correctly status "Complete", "Partial" and "None". At detail screen "0" is displayed at the left of deleted program from SELECT screen.

2.4 USAGE

Basically KAREL programs of System Design Tool are taught with CALL instruction. Needed value to calculate and register number to output results are specified by the parameters. Some KAREL programs do not need parameters.
There are some ways to specify parameters. Specifying constant, string, register or string register.

NOTE
Below is the case error is displayed. Specify correct parameters.
- Wrong type parameters are specified
- Lack of parameters.
- Out of range

After modifying the program, please set the cursor onto a line before CALL instruction and execute again.

NOTE
CALL instruction must be used to use KAREL programs of System Design Tool. RUN instruction does not work correctly.

NOTE
Set system variable $KAREL_ENB = 1 when KAREL program is taught. It is set 1 on System Design Tool option (J738).

2.4.1 Sample of Using Constants

As an example this subsection use COS function of trigonometric function.
60, degree to calculate, is input as 1st parameter. 1, register number to output the result, is input as the 2nd parameter. PCOS is described later.
The value of COS 60, that is 0.5, is output to R[1] in following sample.

Teach Sample

```
RSR0001
                                                  1/2
     1: CALL PCOS(60, 1)
    [END]
```

2.4.2 Sample of Using Register

This subsection explains how to use register.
Following is the teach sample. Suppose R[2] = 3, R[3] = 60, R[4] = 1.

Teach Sample

```
RSR0001
                                                  1/2
     1: CALL PCOS(R[R[2]], R[4])
    [END]
```

The same result is output to R[1] as above of specified constants as parameters.

Because R[2] is 3 in above sample, the 1st parameter (angle in degrees) is evaluated as 60 (R[R[2]] = R[3] = 60).

The 2nd parameter, register number to output , is R[4] = 1 in the same way.

> **NOTE**
> The value of register is used as parameter when register is specified by the parameters. Result is not output R[4] in above sample, except R[4] is 4.

2.4.3 Sample of Using String

> **NOTE**
> Set system variable $STRING_PRM to TRUE when string is specified by the parameter. It is set to TRUE on System Design Tool option (J738).

EXT_LOAD of TP Program Memory Assist described below is used as a sample, this subsection explains using string as a parameter.
Following is the sample. It loads the TP program according to the file list, MC:¥SUB¥LIST.DT.

Teach Sample

```
RSR0001
                                              1/2
      1: CALL EXT_LOAD('MC:¥SUB¥LIST')
     [END]
```

2.4.4 Sample of Using String Register

STRBERGL is used as a sample, this subsection explains using string register as a parameter.
Following is the sample to use it.

Teach Sample

```
RSR0001
                                              1/2
      1: CALL STRBERGL(11, SR[1], 2)
     [END]
```

Suppose SR[1] = “Body Error”. “KALM - 011 Body Error” is displayed after executing this sample.

2.4.5 Sample of Using Multitasking Function

KAREL programs of System Design Tool are allowed to execute by multitasking function. Please refer to the section “MULTITASKING FUNCTION” in the chapter “UTILITY” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know multitasking function.

> **NOTE**
> If the same KAREL program is called from various programs simultaneously, “INTP-326 File var is already used” is displayed. Note not to call the KAREL program simultaneously.

Create MAIN.TP and SUB1.TP as follows. Set motion group of SUB1.TP [*,*,*,*,*,*,*,*]. EXT_LOAD is described below.

<MAIN.TP>

```
1:J P[1] 100% FINE      ;
2:   RUN SUB1 ;
3:J P[2] 100% FINE      ;
4:   WAIT R[1]=1      ;
5:   CALL PNS0001      ;
```

<SUB1.TP>

```
1:   R[1]=0      ;
2:   CALL EXT_LOAD('MC:¥WORK1') ;
3:   R[1]=1      ;
```

Executing MAIN.TP and SUB1.TP are executed simultaneously.
This sample is efficient in the case that TP programs for work should be loaded preliminarily.

3 TRIGONOMETRIC FUNCTION

Trigonometric function enables various calculation described in the following sections. Each one has parameters of integer or real number to calculate or register number to output calculated result.

3.1 SIN FUNCTION

This function calculates SIN (1st parameter degree) and output it to the register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PSIN

SYNTAX

PSIN(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: An angle in degrees to calculate SIN.
register_idx: Register number to output result.

SAMPLE

CALL PSIN(30, 1) -> R[1] = 0.5

3.2 COS FUNCTION

This function calculates COS (1st parameter degree) and output it to the register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PCOS

SYNTAX

PCOS(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: An angle in degrees to calculate COS.
register_idx: Register number to output result.

SAMPLE

CALL PCOS(60, 1) -> R[1] = 0.5

3.3 TAN FUNCTION

This function calculates TAN (1st parameter degree) and output it to the register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PTAN

SYNTAX

PTAN(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: An angle in degrees to calculate TAN.
register_idx: Register number to output result.

SAMPLE

CALL PTAN(45, 1) -> R[1] = 1.0

3.4 SIN^{-1} (ARC SINE) FUNCTION

This function calculates SIN^{-1} (1st parameter value) and output it to the register specified by the 2nd parameter. The result is real number. Its unit is degree.

KAREL PROGRAM NAME: PASIN

SYNTAX

PASIN(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate SIN^{-1}.
register_idx: Register number to output result.

SAMPLE

CALL PASIN(0.5, 1) -> R[1] = 30.0

3.5 COS^{-1} (ARC COSINE) FUNCTION

This function calculates COS^{-1} (1st parameter value) and output it to the register specified by the 2nd parameter. The result is real number. Its unit is degree.

KAREL PROGRAM NAME: PACOS

SYNTAX

PACOS(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate COS^{-1}.
register_idx: Register number to output result.

SAMPLE

CALL PACOS(0.5, 1) -> R[1] = 60.0

3.6 TAN^{-1} (ARC TANGENT) FUNCTION

This function calculates TAN^{-1} (1st parameter value and 2nd parameter value) and output it to the register specified by the 3rd parameter. The result is real number. Its unit is degree.

KAREL PROGRAM NAME: PATAN

SYNTAX

PATAN(x, y, register_idx)
[in] x : REAL
[in] y : REAL
[out] register_idx : INTEGER

DETAIL

x: The value of X coordinate to calculate TAN^{-1}.
y: The value of Y coordinate to calculate TAN^{-1}.
register_idx: Register number to output result.

SAMPLE

CALL PATAN(1, 1, 1) -> R[1] = 45.0

3.7 ABSOLUTE VALUE

This function calculates absolute value of the 1st parameter. The result is output to the register specified by the 2nd parameter. The result is the same of 1st parameter.

KAREL PROGRAM NAME: PABS

SYNTAX

PABS(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate absolute value.
register_idx: Register number to output result.

SAMPLE

CALL PABS((-5), 1) -> R[1] = 5

3.8 EXPONENTIAL FUNCTION

This function calculates exponent (1st parameter value) and output it to the register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PEXP

SYNTAX

PEXP(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate exponent.
register_idx: Register number to output result.

SAMPLE

CALL PEXP(1, 1) -> R[1] = 2.71828 (= e^1)

3.9 LOGARITHM

This function calculates logarithm (1st parameter value) and output it to the register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PLN

SYNTAX

PLN(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate logarithm.
register_idx: Register number to output result.

SAMPLE

CALL PLN(2.781828, 1) -> R[1] = 0.99999

3.10 TRUNCATION

The result of calculation is the value of the 1st parameter after any fractional portion has been removed. The result is output to the register specified by the 2nd parameter. The result is integer.

KAREL PROGRAM NAME: PTRUNC

SYNTAX

PTRUNC(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate truncation.
register_idx: Register number to output result.

SAMPLE

CALL PTRUNC(10.5, 1) -> R[1] = 10

3.11 ROUND OFF

The result of calculation is INTEGER value closest to the 1st parameter. The result is output to the register specified by the 2nd parameter. The result is integer.

KAREL PROGRAM NAME: PROUND

SYNTAX

PROUND(x, register_idx)
[in] x : REAL
[out] register_idx : INTEGER

DETAIL

x: The value to calculate round off.
register_idx: Register number to output result.

SAMPLE

CALL PROUND(10.5, 1) -> R[1] = 11

3.12 SQUARE ROOT

This function calculates the square root of the 1st parameter. The result is output to register specified by the 2nd parameter. The result is real number.

KAREL PROGRAM NAME: PSQRT

SYNTAX

PSQRT(x, register_idx)
[in] x : INTEGER
[out] register_idx : INTEGER

DETAIL

x: The value to calculate square root. Positive integer or positive real number.
register_idx: Register number to output result.

SAMPLE

CALL PSQRT(2, 1) -> R[1] = 1.41421

4 ROBOT SETUP ASSISTANCE TOOL

This chapter describes Robot Setup Assistance Tool.

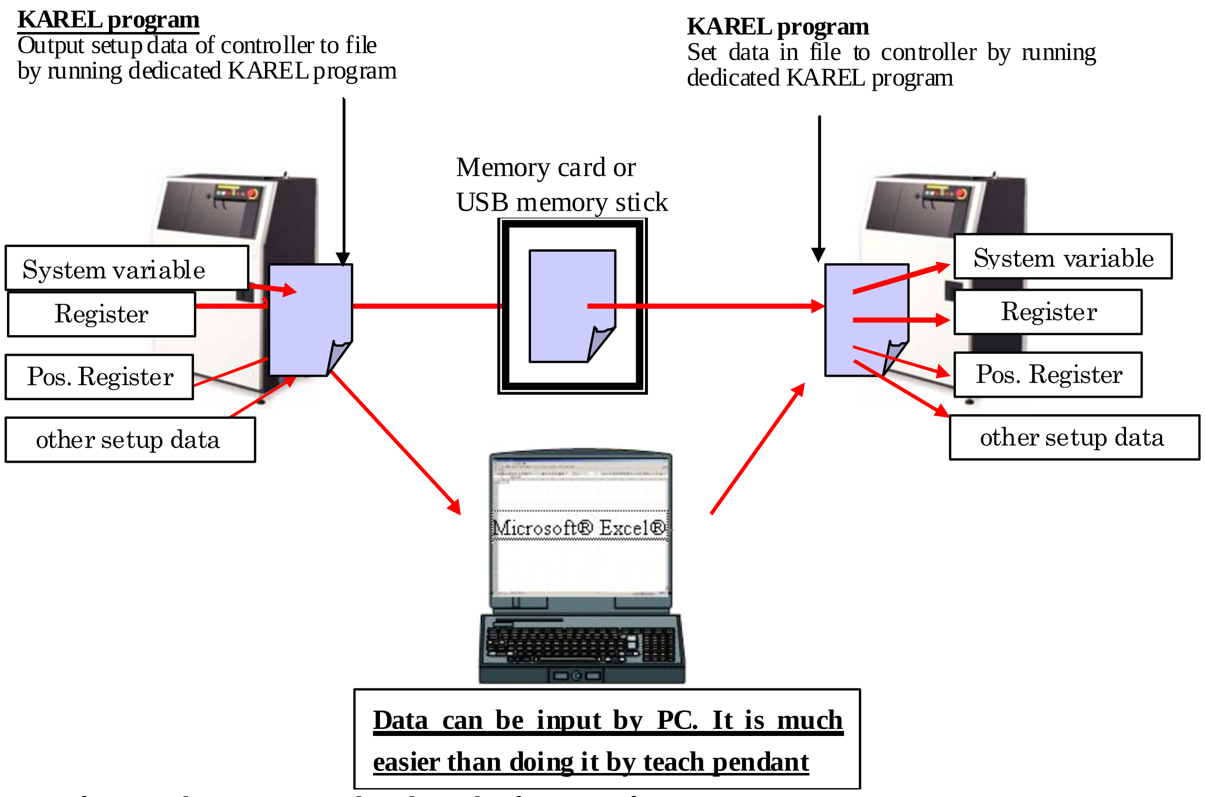
NOTE
Robot Setup Assistance tool option (J737) is included in System Design Tool option (J738). If you need this function only, please order option J737.

4.1 OVERVIEW

Robot Setup Assistance tool is function to setup Robot efficiently. With this tool, you can do following work efficiently.

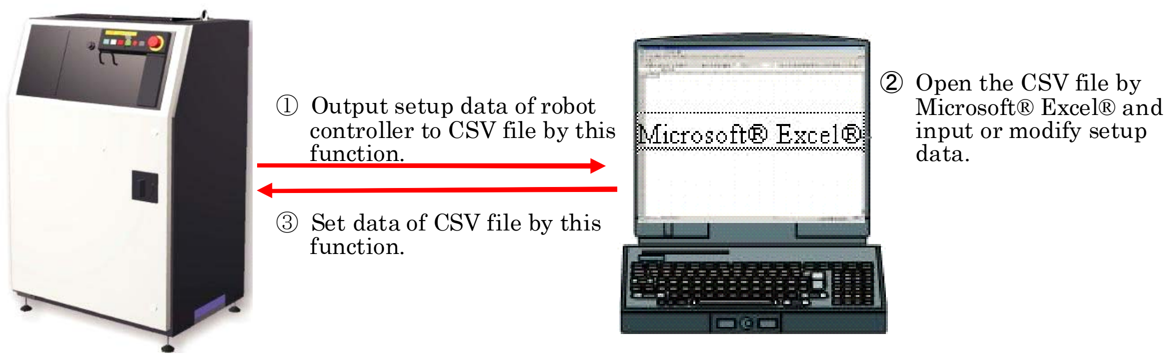
- **Setup various data of robot controllers**
- **Output data of various setup items to file**
- **Copy of setup data to another robot controller**

This function can output various setup data of R-30*i*B/R-30*i*B Mate (robot controller or controller hereafter) to CSV file. This function can also setup controller by reading CSV file.
You can display menu for this tool by loading KAREL programs by System Design Tool and run main program. Using the menu, you can output data to CSV file and setup data of CSV file.


KAREL program
Output setup data of controller to file by running dedicated KAREL program
KAREL program
Set data in file to controller by running dedicated KAREL program
Memory card or USB memory stick
System variable
Register
Pos. Register
other setup data
System variable
Register
Pos. Register
other setup data
Microsoft® Excel®
Data can be input by PC. It is much easier than doing it by teach pendant

Microsoft® Excel® is registered trademark of Microsoft Corporation.

Example 1

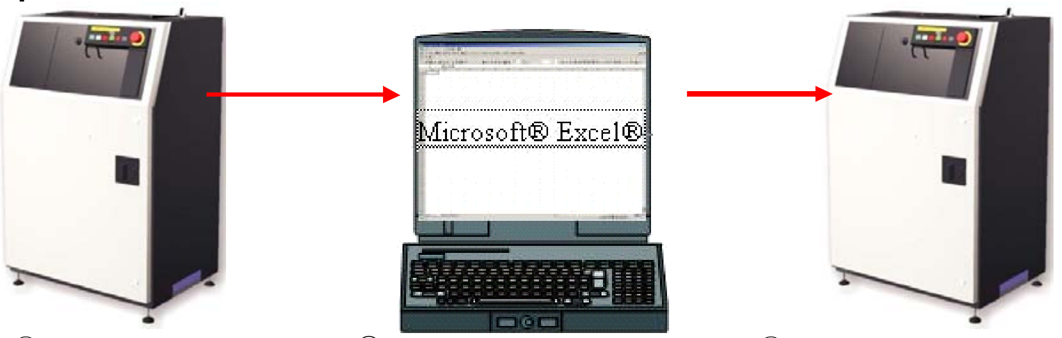


① Output setup data of robot controller to CSV file by this function.

② Open the CSV file by Microsoft® Excel® and input or modify setup data.

③ Set data of CSV file by this function.

Example 2

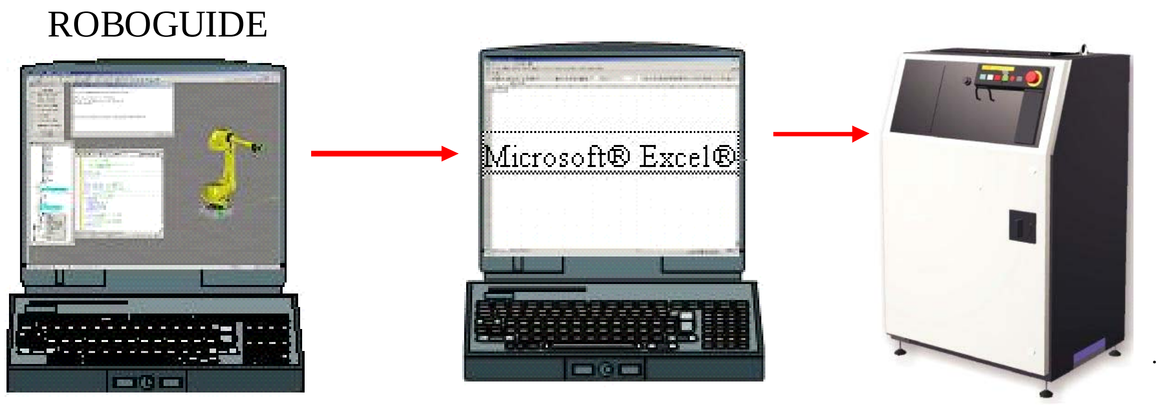


① Output setup data of robot controller to CSV file by this function.

② Open the CSV file by Microsoft® Excel® and input or modify setup data.

③ Set data of CSV file to another controller by this function.

Example 3



① Output setup data to CSV file by this function on ROBOGUIDE.

② Open the CSV file by Microsoft® Excel® and input or modify setup data.

③ Set data of CSV file to real robot controller by this

CSV (Comma Separated Value) format file is text file in which data is separated by comma. Even though you can create CSV file directly by text editor, Microsoft® Excel® can be used to create CSV file by inputting data and saving file in CSV format.
CSV file can be read and edited by Microsoft® Excel®.
Following table is an example of display of CSV file by Microsoft® Excel®.

Microsoft® Excel® is registered trademark of Microsoft Corporation.

A, B, C… of first line is column number of Microsoft® Excel®.

A	B	C	D	E	F
REM	Register				
REM	Index	Comment	INT/REAL	value	!!!!(Terminator)
REG	1		INT	0	!!!!
REM	Position register				
REM	Index	Comment	!!!!(Terminator)		
REM	Group(GPn)	Type	The number of extended axis/total axis	!!!!(Terminator)	
REM	X/J1	Y/J2	Z/J3	!!!!(Terminator)	
REM	W/J4	P/J5	R/J6	!!!!(Terminator)	
REM	E1	E2	E3	!!!!(Terminator)	
REM	Config	!!!!(Terminator)			
PREG	1	Test	!!!!		
	GP1	XYZ	6	!!!!	
	1806.997	3.33	1300	!!!!	
	-179.894	-90	0	!!!!	
	N U T, 0, 0, 0	!!!!			
REM	I/O Comment (DI)				
REM	Index	Comment	!!!!(Terminator)		
DO	1	Door 1 open	!!!!		
REM	User alarm				
REM	Index	Alarm message	Alarm severity	!!!!(Terminator)	
UALM	1	Door 1 open	WARN	!!!!	
END					

You can create new CSV file using format that is described in later section. However, it is easier to output CSV file as template and to modify it. We recommend using CSV file that is already output as template file.

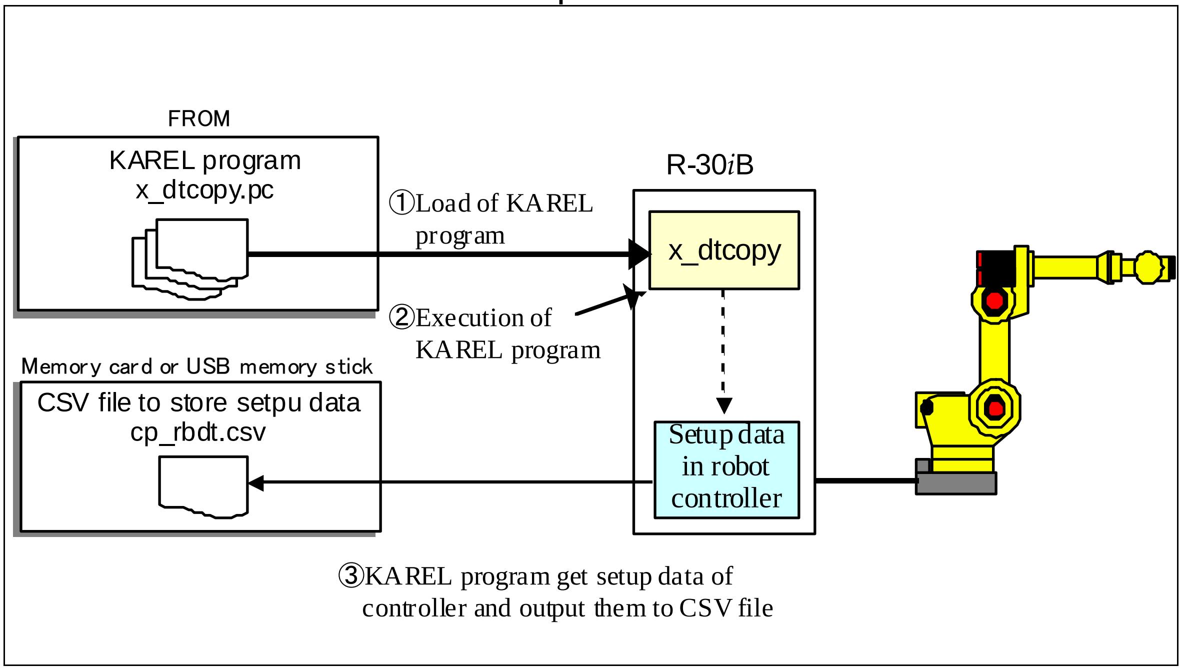
File name that this function uses is cp_rbdt.csv by default. You can change this filename when you use this function. You can use cp_xxxx.csv (x is numeric character from 0 to 9).

NOTE
Only limited setup data is supported by this function. For example, this function does not support setup data of spot application.

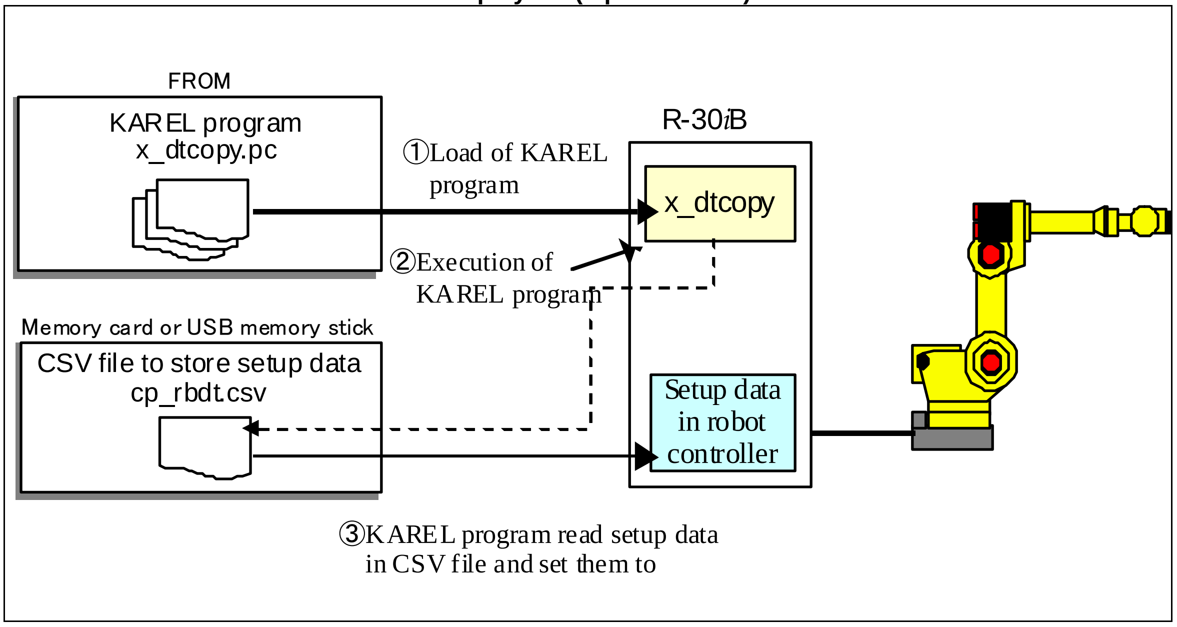
4.2 STRUCTURE OF FUNCTION

KAREL programs compose this function. You can use this function by loading KAREL programs in System Design Tool screen and by running x_dtcopy.pc.

Output to file


FROM
KAREL program
x_dtcopy.pc
Memory card or USB memory stick
CSV file to store setpu data
cp_rbdt.csv
①Load of KAREL program
②Execution of KAREL program
R-30*i*B
x_dtcopy
Setup data in robot controller
③KAREL program get setup data of controller and output them to CSV file

Setup by file (Input from file)


FROM
KAREL program
x_dtcopy.pc
Memory card or USB memory stick
CSV file to store setup data
cp_rbdt.csv
①Load of KAREL program
②Execution of KAREL program
R-30*i*B
x_dtcopy
Setup data in robot controller
③KAREL program read setup data in CSV file and set them to

NOTE
It is enough to load only one time in the sequence ①. It is not necessary to load in each time.

4.3 SUPPORTED DATA

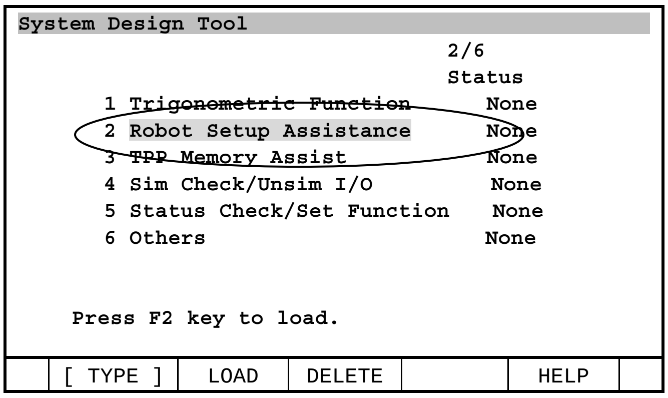
- Register
- Position Register
- Comment of I/O (DIO, RIO, GIO, AIO, SI, SO, UIO)
- Assignment of I/O (DIO, GIO, AIO, UIO)
- Comment of program
- Comment of position (P[ ]) of program
- User alarm
- Macro
- Frames (tool, user and jog)
- Reference position
- Payload
- Armload
- Space check function
- Automatic backup
- Items of system configuration screen
- Program timer
- String register
- Battery alarm
- Override select
- Warning/Reset in alarm history
- TCP/IP
- Error table
- Alarms that do not turn on FAULT signal
- RSR, PNS
- *i*Pendant backlight automatic blanking

NOTE
Some items don’t support all items in setup screens. Please refer to the section of each item.

4.4 HOW TO LOAD KAREL PROGRAMS

To use this function, you have to load KAREL programs. You can load KAREL programs in System Design Tool screen.

1 Press [MENU]
2 Press SETUP
3 Press F1, [TYPE]
4 Select “Sys Dsn Tool”. This leads you to screen like following figure.


```
System Design Tool
                                   2/6
                                   Status
      1 ~~Trigonometric Function~~      None
      2 Robot Setup Assistance      None
      3 ~~TPP Memory Assist~~           None
      4 Sim Check/Unsim I/O         None
      5 Status Check/Set Function   None
      6 Others                      None


   Press F2 key to load.
```

5 Move cursor to Robot Setup Assistance.
6 Press F3, Load.

It can also delete all related KAREL programs all at once. Please refer to the section 2.3 "DELETE KAREL PROGRAMS OF SYSTEM DESIGN TOOL".

NOTE
Don’t delete or load a KAREL program of Robot Setup Assistance individually. Delete or load programs all at once. If you accidentally delete or load a KAREL program of Robot Setup Assistance individually, delete all KAREL programs first. Then load them all at once.

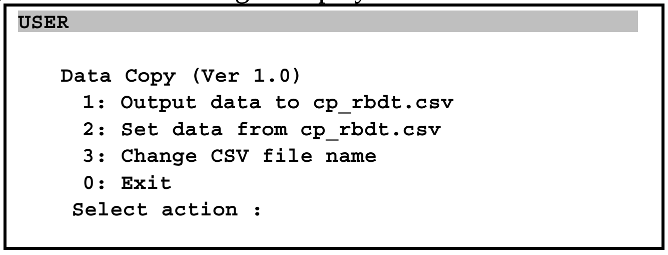
4.5 USAGE

4.5.1 How to Display Menu

By running KAREL program, this function can display menu. You can select following operation in the menu.

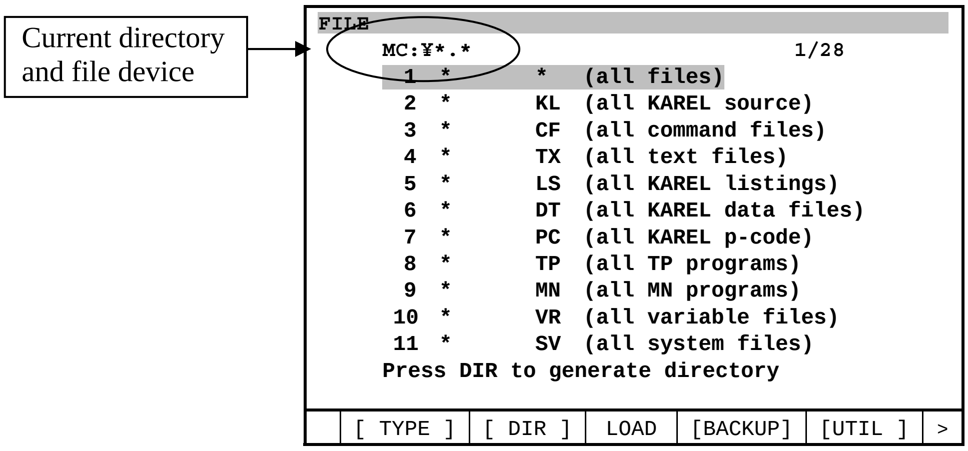
- Data output
- Setup of data from CSV file
- Change of file name
- Exit

1 Load KAREL programs by procedure in previous section.
2 Display SELECT screen
3 Select All from F1, [TYPE]
4 Select KAREL program x_dtcopy
5 Run x_dtcopy by Shift+FWD.
KAREL program execution continues after release of Shift key.
6 By execution of x_dtcopy, USER menu is displayed
After a short time, menu like following is displayed.


```
USER

   Data Copy (Ver 1.0)
     1: Output data to cp_rbdt.csv
     2: Set data from cp_rbdt.csv
     3: Change CSV file name
     0: Exit
    Select action :
```

4.5.2 Output Data to CSV File

This function can output various setting to CSV file. File is output in current directory of current device selected by file screen, MC: for example.



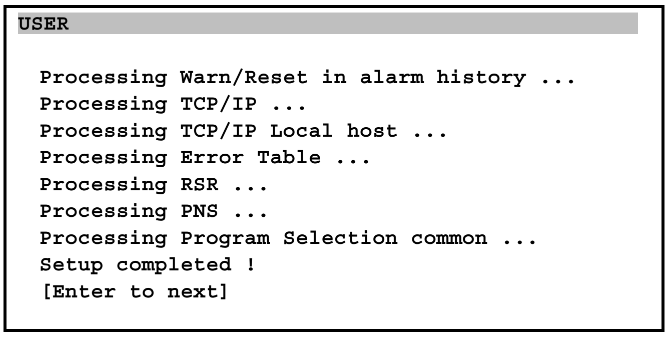
```
FILE
     MC:¥*.*                              1/28
       1  *       *   (all files)
       2  *       KL  (all KAREL source)
       3  *       CF  (all command files)
       4  *       TX  (all text files)
       5  *       LS  (all KAREL listings)
       6  *       DT  (all KAREL data files)
       7  *       PC  (all KAREL p-code)
       8  *       TP  (all TP programs)
       9  *       MN  (all MN programs)
      10  *       VR  (all variable files)
      11  *       SV  (all system files)
     Press DIR to generate directory

   [ TYPE ] [ DIR ]  LOAD  [BACKUP] [UTIL ]  >
```

NOTE

Output data when file device currently selected by file screen is ready to be written. Otherwise, alarms happen after file output even though message “Data output completed!” is displayed. For example:

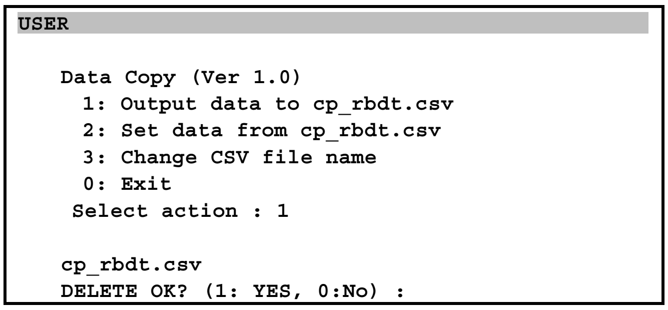
1 Don’t output file to MC: without inserting memory card to the slot.
2 Don’t output file by ROBOGUIDE with opening file by Microsoft® Excel®.

1 Select file device.
2 Select “1 Output data to cp_rbdt.csv” from menu of this function.
File output starts. After output completes, message “Data output completed!” is displayed.



```
USER

  Processing Warn/Reset in alarm history ...
  Processing TCP/IP ...
  Processing TCP/IP Local host ...
  Processing Error Table ...
  Processing RSR ...
  Processing PNS ...
  Processing Program Selection common ...
  Setup completed !
  [Enter to next]
```

If output file already exists, warning message “cp_rbdt.csv already exist. Delete OK? (1:Yes, 0:No)” is displayed. If you want to overwrite the file, input 1. Otherwise, input 0.



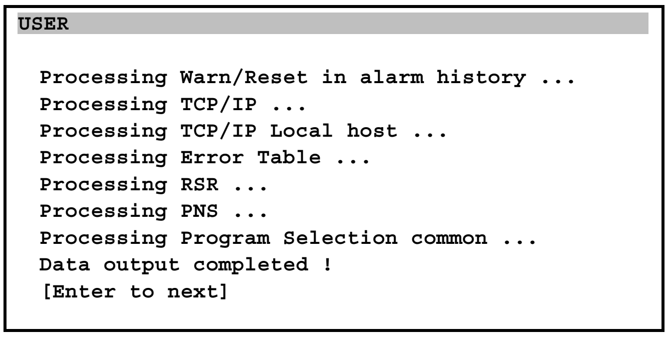
```
USER

   Data Copy (Ver 1.0)
     1: Output data to cp_rbdt.csv
     2: Set data from cp_rbdt.csv
     3: Change CSV file name
     0: Exit
    Select action : 1

   cp_rbdt.csv
   DELETE OK? (1: YES, 0:No) :
```

3 Prompt message “[Enter to next]” is displayed. Press Enter key to return to menu.

4.5.3 Setup Data From CSV File

This function can set data written in file to robot controller. Data file to read is CSV file. Its format is described in later sections. File in current directory of currently selected file device is read.

1 Place CSV file in current directory of currently selected file device.
2 Select “Set data from cp_rbdt.csv” from menu of this function.
Data read and setup begin. After data setup complete, “Setup Completed!” is displayed.



```
USER

  Processing Warn/Reset in alarm history ...
  Processing TCP/IP ...
  Processing TCP/IP Local host ...
  Processing Error Table ...
  Processing RSR ...
  Processing PNS ...
  Processing Program Selection common ...
  Data output completed !
  [Enter to next]
```

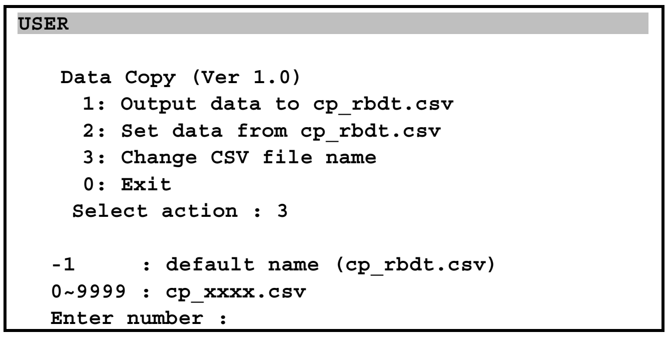
3 Prompt message “[Enter to next]” is displayed. Press Enter key to return to menu.

NOTE
After data set, cycle power is needed.

4.5.4 Change CSV File Name

Default file name is cp_rbdt.csv. You can change the file name to cp_xxxx.csv (xxxx is numeric characters). You can set the name back to default name, cp_rbdt.csv.

1 Select “Change CSV file name” from menu of this function.
2 Input number to decide file name.



```
USER

   Data Copy (Ver 1.0)
     1: Output data to cp_rbdt.csv
     2: Set data from cp_rbdt.csv
     3: Change CSV file name
     0: Exit
    Select action : 3

  -1     : default name (cp_rbdt.csv)
  0~9999 : cp_xxxx.csv
  Enter number :
```

File name is decided by entering number from –1 to 9999. If you input –1, file name is set back to default name, cp_rbdt.csv. If you input value between 0 and 9999, the number is used as the last 4 character of file name. For example, if you input 109, the new file name is cp_0109.csv.

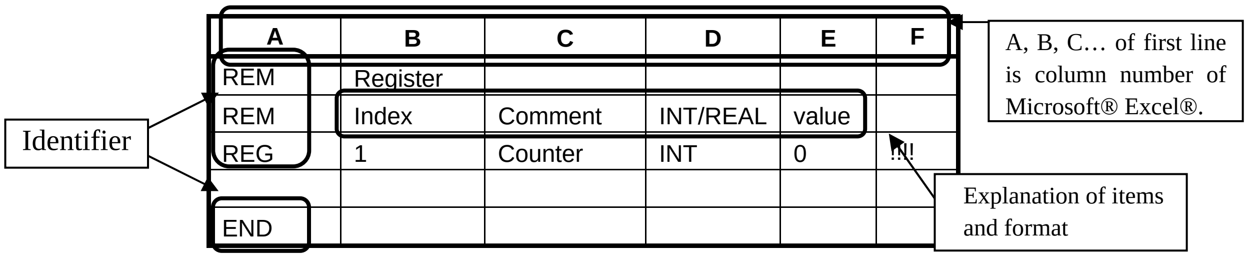
```
USER
     2: Set data from cp_rbdt.csv
     3: Change CSV file name
     0: Exit
    Select action : 3

  -1     : default name (cp_rbdt.csv)
  0~9999 : cp_xxxx.csv
  Enter number :  109
  File name was changed to cp_0109.csv
  [Enter to next]
```

3 Prompt message "[Enter to next]" is displayed. Press Enter key to return to menu.

4.6 OVERVIEW OF CSV FILE FORMAT

Data identifier

The first filed of data line is identifier to show kind of data of the line. Column A of following table has "REM", "REG" and "END". These are identifier.


Identifier
A, B, C… of first line is column number of Microsoft® Excel®.
Explanation of items and format
A
B
C
D
E
F
REM
Register
REM
Index
Comment
INT/REAL
value
REG
1
Counter
INT
0
!!N!
END

Meaning of identifiers is as follows. Input them in uppercase.

Identifier	Description
REM	Comment line
END	End of file
REG	Numeric register
PREG	Position register
DI	Comment of digital input signal
DIA	Assignment of digital input signal
DO	Comment of digital output signal
DOA	Assignment of digital output signal
RI	Comment of robot input signal
RO	Comment of robot output signal
GI	Comment of group input signal
GIA	Assignment of group input signal
GO	Comment of group output signal
GOA	Assignment of group output signal
AI	Comment of analog input signal
AIA	Assignment of analog input signal
AO	Comment of analog output signal
AOA	Assignment of analog output signal
SI	Comment of operator panel input signal
SO	Comment of operator panel output signal
UI	Comment of peripheral input signal
UIA	Assignment of peripheral input signal
UO	Comment of peripheral output signal
UOA	Assignment of peripheral output signal
PRG	Comment of program

Identifier	Description
POS	Comment of position of program
UALM	User alarm
MCR	Macro
TFRM	Tool frame
UFRM	User frame
JFRM	Jog frame
REF	Reference position
PLD	Payload setting
ALD	Armload setting
RSP	Space check function
ATBK	Automatic backup
SYSC	System configuration screen
PTIM	Program timer
SREG	String register
BLAL	Battery alarm
OVSL	Override select
NOHIS	Warning and reset in alarm history
TCPIP	TCP/IP
TCPIP_HL	TCP/IP (local host)
ER_TBL	Error severity
ER_NOALM	Alarms that do not turn on FAULT signal
RSR	RSR
PNS	PNS
PRG_SEL	Common item of RSR and PNS. Program selection
BLNK	*i*Pendant backlight automatic blanking

Terminator (!!!!)

Basically, each line has to complete with “!!!!”, which is terminator of line. The exception is comment line (REM) and end line (END).

Identifiers that have 2 or more lines

Some kinds of identifier have 2 or more data lines. The 2nd or later lines need empty field at the head of lines. Empty field is field that does not have character.

A	B	C	D	E
REM	Position register			
REM	Index	Comment	!!!!(Terminator)	
REM	Group(GPn)	Type	The number of extended axis/total axis	!!!!(Terminator)
REM	X/J1	Y/J2	Z/J3	!!!!(Terminator)
REM	W/J4	P/J5	R/J6	!!!!(Terminator)
REM	E1	E2	E3	!!!!(Terminator)
R	[illegible]	inator)		
PREG	1		!!!!	
	GP1	XYZ	0	!!!!
	1592	0	1085	!!!!
	-180	0	0	!!!!
	N U T, 0, 0, 0	!!!!		

The head of 2nd and later lines are empty filed

This PREG has 5 lines

4.7 REMOVAL OF UNNECESSARY DATA

When you create data file using file output by this function, it is better to remove unnecessary lines. It lowers possibility of unexpected setup of data. It reduces time to process.
You have to remove lines not to disturb format of data. This means you have to consider format of each identifier. In addition, it is helpful to understand how this function output file.

Making use of delimiter between data sets

This function output data of an identifier all at once. Then the function output data of another identifier. Following table shows example.

A	B	C	D	E	F
REM	Register				
REM	Index	Comment	INT/REAL	value	!!!!(Terminator)
REG	1	Comment of R[1]	INT	0	
REG	2		INT	0	
REG	3		INT	0	!!!!
REG	4		INT	0	!!!!
Snip					
REG	198		INT	0	!!!!
REG	199		INT	0	!!!!
REG	200	Comment of R[200]	INT	0	!!!!
REM	Position register				
REM	Index	Comment	!!!!(Terminator)		
REM	Group(GPn)	Type	The number of extended axis/total axis	!!!!(Terminato	
REM	X/J1	Y/J2	Z/J3	!!!!(Terminato	
REM	W/J4	P/J5	R/J6	!!!!(Terminator)	
REM	E1	E2	E3	!!!!(Terminator)	
REM	Config	!!!!(Terminator)			
PREG	1		!!!!		
	GP1	XYZ	U	!!!!	
PREG	2		!!!!		
	GP1	XYZ	U	!!!!	
Snip					
PREG	99		!!!!		
	GP1	XYZ	U	!!!!	
PREG	100		!!!!		
	GP1	XYZ	U	!!!!	
REM	I/O Comment (DI)				
REM	Index	Comment	!!!!(Terminator)		
DI	1		!!!!		
DI	2		!!!!		

Register("REG") and explanation comment for it ("REM")

Empty line as delimiter

Position register ("PREG") and explanation comment for it ("REM")

Empty line as delimiter

This example is output by controller with 200 registers and 100 position registers. This function output register 1 through 200 first. Then it output blank line and position register 1 through 100. The rest of data are output in the same way. If you remove all lines between blank line before and after position register section, you can remove all description for position registers.

4.8 FORMAT OF DATA

When this function outputs data, comment (REM) lines are written before data set of each identifier. The comment lines express data format of identifier. You can refer to the comment in addition to this manual. This section shows the comment in addition to data lines.

4.8.1 Comment Lines in Data File

Identifier

REM

Syntax

Identifier + comment string

Example

A	B
REM	This is an example of comment
REM	When you output file, comment that you can
REM	Refer to know data format is output

Detail

This identifier is used to write comment in CSV file. When data is set from file, lines that start with this identifier are ignored.

4.8.2 END of CSV File

Identifier

END

Syntax

Identifier

Example

A	B	C	D
END			

Detail

This identifier specifies the end of file. You must write this identifier with line feed at the end of file.

4.8.3 Register

Identifier

REG

Syntax

Identifier + index + comment + data type (INT/REAL) + value + terminator

Example

A	B	C	D	E	F
REM	Register				
REM	Index	Comment	INT/REAL	value	

A	B	C	D	E	F
REG	1	This is integer	INT	0	!!!!(Terminator)
REG	2	This is real	REAL	1.23	!!!!

Detail

Item	Description
Index	Index of register. If index is out of range when data is set, message “Illegal index” is displayed.
Comment	Maximum 16 characters.
INT/REAL	If register has integer value, this filed should be INT. Otherwise (real number), REAL.
Value	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.

Note

When file is output, all registers are output.

4.8.4 Position Register

Identifier

PREG

Syntax

Format of the 3rd and later lines depend on data type and the number of axes.

The 1st line
  Identifier + Index + Comment + terminator
The 2nd line
  Empty field + group number + data type + the number of extended axes/ total axes + terminator
The 3rd line
  Empty field + X (J1) + Y (J2) + Z (Z3) + terminator
The 4th line
  Empty field + W (J4) + P (J5) + R (Z6) + terminator
The 5th line
  Empty field + Extended axis 1 + Extended axis 2 + Extended axis 3 + terminator
The 6th line (Cartesian type only)
  Empty field + configuration + terminator

If there are 2 or more groups, the 2nd – 6th lines are repeated as required.
When data is set from file, you have to write data for all groups.
You cannot change the order of lines. You cannot omit lines.
To avoid format error, it is recommended to output CSV file before setting data from file and to use output file as template file.

Example

A	B	C	D	E
REM	Position register			
REM	Index	Comment	!!!!(Terminator)	
REM	Group(GPn)	Type	The number of extended axis/total axis	!!!!(Terminator)
REM	X/J1	Y/J2	Z/J3	!!!!(Terminator)
REM	W/J4	P/J5	R/J6	!!!!(Terminator)
REM	E1	E2	E3	!!!!(Terminator)
REM	Config	!!!!(Terminator)		
PREG	1	Ex. cartesian	!!!!	
	GP1	XYZ	0	!!!!

A	B	C	D	E
	1807	0	1300	!!!!
	180	-90	0	!!!!
	N U T, 0, 0, 0	!!!!		
	GP2	XYZ	0	!!!!
	1807	0	1300	!!!!
	180	-90	0	!!!!
	N U T, 0, 0, 0	!!!!		
PREG	2	Ex. joint pos	!!!!	
	GP1	JPOS	6	!!!!
	1	2	3	!!!!
	4	5	6	!!!!
	GP2	JPOS	6	!!!!
	10	11	12	!!!!
	13	14	15	!!!!
PREG	3		!!!!	
	GP1	XYZ	U	!!!!
	GP2	XYZ	U	!!!!

Detail

Item	Description
Index	Index of position register. If index is out of range when data is set, message “Illegal index” is displayed.
Comment	Maximum 16 characters.
Group (GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
Type	Data type of position register. One of following is specified. JPOS Joint position XYZ XYZWPR RXYZWPR XYZWPREXT (Cartesian type with extended axis)
The number of extended axes/total axes	If “Type” field is RXYZ, this field is the number of extended axes. If XYZ, this field should be 0. If “Type” field is JPOS, this field is the number of total axes of the group. When position register is uninitialized, this field is “U”. The 3rd, 4th and the 5th line are not output. When data is set from file, data of “U” is ignored. Setting data from file does not make position registers uninitialized.
X/J1, Y/J2, Z/J3, W/J4, P/J5, R/J6	These fields specify value of each data.
E1, E2, E3	These fields are output only when data type is RXYZ. For setting data, these values should be specified only when data type is RXYZ. Value of extended axes are specified
Config	This field specifies string that represents configuration data. Line with this field is needed only for Cartesian data type (XYZ and RXYZ). String for configuration data depends on robot. Make use of output file to decide string for configuration.

Note

Matrix type is not supported.

4.8.5 Comment of I/O

Identifier

Identifier depends on type of I/O. Following identifiers are supported.

Identifier	Description
DI	Comment of digital input
DO	Comment of digital output
RI	Comment of robot input
RO	Comment of robot output
GI	Comment of group input
GO	Comment of group output
AI	Comment of analog input
AO	Comment of analog output
SI	Comment of operator panel input
SO	Comment of operator panel output
UI	Comment of peripheral input
UO	Comment of peripheral output

Syntax

Identifier + index + comment + terminator

Example

Following table is example of DI. The other types has similar format.

A	B	C	D
REM	I/O Comment (DI)		
REM	Index	Comment	!!!!(Terminator)
DI	1	Door 1 open	!!!!

Detail

Item	Description
Index	Index of signal. This function supports limited range of index. The maximum index is as follows. DI, DO: 2048 RI, RO: 16 GI, GO: 64 AI, AO: 64 SI, SO: 64 UI: 36 UO: 40
Comment	Maximum 24 characters.

Note

This function output comment of signals that satisfy following condition (A) and (B) at the same time.

(A) Index is equal to or less than maximum value (Please refer to table in “Detail” above).

(B) One of following is satisfied.

- Comment is set
- Comment is not set but value of I/O can be read.

4.8.6 Assignment of I/O

Identifier

Identifier depends on type of I/O. Following identifiers are supported.

Identifier	Description
DIA	Assignment of digital input.
DOA	Assignment of digital output.
GIA	Assignment of group input.
GOA	Assignment of group output.
AIA	Assignment of analog input.
AOA	Assignment of analog output.
UIA	Assignment of peripheral input.
UOA	Assignment of peripheral output.

Syntax

DI, DO, GI, GO, UI and UO

Identifier + Index + Rack number + Slot number + Start point + the number of points + terminator

AI and AO

Identifier + Index + Rack number + Slot number + Channel number + the number of points (always 1) + terminator

Example

Following table is an example of DI. DO, UI and UO have similar format.

A	B	C	D	E	F	G
REM	I/O Assign (DI)					
REM	Index	Rack number	Slot number	Start point	Number of points	!!!!(Terminator)
DIA	1	0	1	19	8	!!!!

Following table is an example of GI. GO has similar format.

A	B	C	D	E	F	G
REM	I/O Assign (GI)					
REM	Index	Rack number	Slot number	Start point	Number of points	!!!!(Terminator)
GIA	5	0	2	25	10	!!!!

Following table is an example of AI. AO has similar format.

A	B	C	D	E	F	G
REM	I/O Assign (AI)					
REM	Index	Rack number	Slot number	Channel	Number of points	!!!!(Terminator)
AIA	1	0	1	1	1	!!!!

Detail

Item	Description
Index	Logical index to start assignment.
Rack number Slot number	Rack number and slot number.
Start point	This is for DI, DO, GI, GO, UI and UO. Physical point to start assignment.
Channel	This is for AI and AO. This field specifies channel number.
Number of points	For DI, DO, UI and UO, the maximum number is 8. Assignment that has more than 9 successive points is divided into assignment of 8 points or less. This is true to both data output and data set. For GI and GO, this field specifies the number of points for the signal. For AI and AO, this field should be 1.

NOTE

If assignment read from CSV file is wrong assignment for the controller, it is not actually assigned. However, error message is not displayed. Set valid assignment. Be careful that assignment in CSV file is valid on controller that read the CSV file. Be careful especially when you use CSV file made by ROBOGUIDE on real controller.

Note

When data is set, this function clears assignment of predefined range of index first. Following table shows the range.

I/O	Range
DO, DI	1-2048 (Usually, only 1 through 512 are displayed)
GI, GO	1-64
AI, AO	1-64
UI	1-36
UO	1-40

4.8.7 Comment of Program

Identifier

PRG

Syntax

Identifier + Program name + Program type + Comment + terminator

Example

A	B	C	D	E
REM	Program			
REM	Program name	Type	Comment	!!!!(Terminator)
PRG	PNS0001	TP	Body 1	!!!!

Detail

Item	Description
Program name	Program name. Maximum 36 characters. When data is set from file and when specified program does not exist, the line is ignored.
Type	When data is set from file, this field is ignored. TP: TP program MR: Macro Program PC: KAREL program VR: KAREL variable Comment of TP and MR can be set by this function. If sub type is CH, it is treated as TP.
Comment	Maximum 16 characters.

4.8.8 Comment of Position of Program

Identifier

POS

Syntax

Identifier + Program name + Position number + Comment + terminator

Example

A	B	C	D	E
REM	Comment of Position data			
REM	Program name	Position number	Comment	!!!!(Terminator)
POS	PNS0001	1	Home position	!!!!
POS	PNS0001	2		!!!!

Detail

Item	Description
Program name	Program name. Maximum 36 characters if setting is default. When data is set from file and when specified program does not exist, the line is ignored.
Position number	When data is set from file and when specified position does not exist, the line is ignored.
Comment	Maximum 16 characters.

Note

When file is output, positions are output from P[1] to P[999] if exist.

4.8.9 User Alarm

Identifier

UALM

Syntax

Identifier + Index + Alarm message + Severity + terminator

Example

A	B	C	D	E
REM	User alarm			
REM	Index	Alarm message	Alarm severity	!!!!(Terminator)
UALM	1	This is WARN	WARN	!!!!
UALM	2	This is STOP.L	STOP.L	!!!!
UALM	3	This is STOP.G	STOP.G	!!!!
UALM	4	This is ABORT.L	ABORT.L	!!!!
UALM	5	This is ABORT.G	ABORT.G	!!!!

Detail

Item	Description
Index	Index of user alarm. If this is out of range when data is set from file, message “Illegal index” is displayed.
Alarm message	Message of user alarm. Maximum 28 characters.
Alarm severity	This field is specified by one of following values. WARN STOP.L STOP.G ABORT.L ABORT.G

Note

If alarm severity does not correspond to one of severity in above table, the value is output as numerical value when file is output. However, you should not set such value.

4.8.10 Macro

Identifier

MCR

Syntax

Identifier + Index + Macro name + Program name + Assign type + Index of assign + terminator

Example

A	B	C	D	E	F	G
REM	Macro					
REM	Index	Macro name	Program name	Assign type	Index	!!!!(Terminator)
MCR	1	Sample macro1	AA_01	NO	0	!!!!
MCR	2	Sample macro2	AA_02	UK	1	!!!!
MCR	3	Sample macro3	AA_03	SU	1	!!!!
MCR	4	Sample macro4	AA_04	MF	1	!!!!
MCR	5	Sample macro5	AA_05	DI	1	!!!!
MCR	6	Sample macro6	AA_06	RI	1	!!!!
MCR	7	Sample macro7	AA_07	UI	1	!!!!
MCR	8	Sample macro8	AA_08	F	1	!!!!
MCR	9	Sample macro9	AA_09	M	1	!!!!
MCR	10	Sample macro10	AA_10	UK	2	!!!!

Detail

Item	Description
Index	Index of Macro. If this is out of range when data is set from file, message “Illegal index” is displayed.
Macro name	Name of macro. Maximum 36 characters. When data is set from file, preceding spaces are not deleted. It is because such macro name is valid.
Program name	Maximum 36 characters. When data is set from file, it is set even if the program does not exist.
Assign type	Type of assignment of macro. NO No assignment. This corresponds to “-“ of macro screen. UK User key SU Shift key + User key MF Manual function SP Operator panel DI Digital input RI Robot input UI Peripheral input F Flag M Marker
Index (The 2nd one)	Index of assignment. When data is set from file, this function does not check if index is in valid range. Please set valid value.

4.8.11 Frames

Identifier

Identifier depends on type of frame. Following identifiers are supported.

Identifier	Description
TFRM	Tool frame
UFRM	User frame
JFRM	Jog frame

Syntax

The 1st line

Identifier + Group number + index + comment + terminator

The 2nd line

Empty field + X + Y + Z + terminator

The 3rd line

Empty field + W + P + R + terminator

The data set is always composed of 3 lines. You have to set above lines in this order and no lines can be omitted.

Example

A	B	C	D	E
REM	Tool Frame			
REM	Group(GPn)	Index	Comment	!!!!(Terminator)
REM	X	Y	Z	!!!!(Terminator)
REM	W	P	R	!!!!(Terminator)
TFRM	GP1	1	Eoat1	!!!!
	0	0	0	!!!!
	0	0	0	!!!!

Detail

Item	Description
Group(GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
Index	Index of frame.
Comment	Maximum 20 characters.
X,Y,Z,W,P,R	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.

Note

When file is output, all frames are output.

4.8.12 Reference Position

Identifier

REFP

Syntax

The 1st line

Identifier + Group number + index + comment + enable/disable of reference position + enable/disable of home position + signal type + signal index + terminator

The 2nd line – the 10th line

Empty field + position of axis + tolerance + terminator

The data set is always composed of 10 lines. You have to set above lines in this order and no lines can be omitted.

Example

A	B	C	D	E	F	G	H	I
REM	Reference position							

A	B	C	D	E	F	G	H	I
REM	Group(GPn)	Index	Comment	ENABLE/ DISABLE	Is a valid Home	Signal type	Signal Index	!!!!
REM	J1-J9 position	Tolerance	!!!!					
REFP	GP1	1		DISABLE	DISABLE	DO	0	!!!!
	0	2	!!!!					
	-40	2	!!!!					
	0	2	!!!!					
	0	2	!!!!					
	-90	2	!!!!					
	0	2	!!!!					
	0	0	!!!!					
	0	0	!!!!					
	0	0	!!!!					

Detail

Item	Description
Group(GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
Index	Index of reference position. If this is out of range when data is set from file, message “Illegal index” is displayed.
Comment	Maximum 16 characters.
ENABLE/DISABLE	Enable/Disable of the reference position.
Is a valid Home	If this is valid home, this field is ENABLE. Otherwise, this field is DISABLE.
Signal type	Type of output signal. This is specified by DO or RO.
Signal index	Index of output signal.
J1 - J9 position Tolerance	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.

Note

When file is output, all reference positions are output.

4.8.13 Payload Setting

Identifier

PLD

Syntax

The 1st line

Identifier + Group number + index + comment + payload + terminator

The 2nd line

Empty field + Center of gravity X + Center of gravity Y + Center of gravity Z + terminator

The 3rd line

Empty field + Inertia X + Inertia Y + Inertia Z + terminator

The data set is always composed of 3 lines. You have to set above lines in this order and no lines can be omitted.

Example

A	B	C	D	E	F
REM	Payload				
REM	Group(GPn)	Index	Comment	Payload[kg]	!!!!(Terminator)
REM	Center X	Center Y	Center Z	!!!!(Terminator)	
REM	Inertia X	Inertia Y	Inertia Z	!!!!(Terminator)	
PLD	GP1	1		165	!!!!
	10	20	5	!!!!	
	0	0	0	!!!!	

Detail

Item	Description
Group(GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
Index	Index of payload setting If this is out of range when data is set from file, message “Illegal index” is displayed.
Comment	Maximum 16 characters.
Center X, Y, Z , Inertia X, Y, Z	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.

Note

When file is output, all payload settings of all groups are output.
This function does not support current payload number.

4.8.14 Armload Setting

Identifier

ALD

Syntax

Identifier + Group number + Load on J1 + Load on J3 + terminator

Example

A	B	C	D	E
REM	Arm load			
REM	Group(GPn)	ARM LOAD AXIS#1	ARM LOAD AXIS#3	!!!!(Terminator)
ALD	GP1	10	10.23	!!!!

Detail

Item	Description
Group(GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
ARM LOAD AXIS#1, #3	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.

Note

When file is output, armloads of all groups are output.

4.8.15 Space Check Function

Identifier

RSP

Syntax

The 1st line
Identifier + Group number + index + Comment + terminator
The 2nd line
Empty field + Enable/Disable + output signal type + output signal index + input signal type + input signal index + terminator
The 3rd line
Empty field + priority high/low + Inside/Outside + length/vertex + User frame number + Tool frame number + terminator
The 4th line
Empty field + Base vertex X + Base vertex Y + Base vertex Z + terminator
The 5th line
Empty field + The 2nd vertex X/Side length X + The 2nd vertex Y/Side length Y + The 2nd vertex Z/Side length Z + terminator

Example

A	B	C	D	E	F	G
REM	Space check function					
REM	Group(GPn)	Index	Comment	!!!!(Terminator)		
REM	ENABLE/DISABLE	Output Type	Output Index	Input Type	Input Index	!!!!(Terminator)
REM	Priority(HIGH/LOW)	INSIDE/OUTSIDE	length/vertex	UF n	UT n	!!!!(Terminator)
REM	Base X	Base Y	Base Z	!!!!(Terminator)		
REM	2nd/Length X	2nd/Length Y	2nd/Length Z	!!!!(Terminator)		
RSP	GP1	1		!!!!		
	DISABLE	DO	0	DI	0	!!!!
	HIGH	INSIDE	LENGTH	0	1	!!!!
	1600	500	1200	!!!!		
	150	150	300	!!!!		

Detail

Item	Description
Group(GPn)	Group number. Only 3 characters are recognized. The first 2 characters should be “GP”. The last character is group number. If group number is out of range when data is set, message “Illegal group” is displayed.
Index	Index of space check function setup data. If this is out of range when data is set from file, message “Illegal index” is displayed.
Comment	Maximum 10 characters.
ENABLE/DISABLE	Enable/Disable of the space.
Output Type	Type of Output signal. DO or RO.
Output Index	Index of Output signal.
Input Type	Type of Output signal. DI or RI.
Input Signal	Index of Input signal.
Priority (HIGH/LOW)	Specify HIGH or LOW.
INSIDE/OUTSIDE	Specify INSIDE or OUTSIDE.
length/vertex	Specify LENGTH or VERTEX.
UF, UT	The number of user frame and tool frame.

Item	Description
Base X, Y, Z	When data is output, the number of decimal places is 3. When you create CSV file to set data, do likewise.
2nd/Length X, Y, Z	“length/vertex” field in the 3rd line decides the meaning of these fields. If LENGTH, these fields are length of each side. If VERTEX, these fields shows position of the 2nd vertex.

Note

When file is output, all space settings are output.
If value of UF or UT is negative, error happens. But there is no more value check for them.

4.8.16 Automatic Backup

Identifier

ATBK

Syntax

The 2nd field is item number. It identifies what the item is. Format depends on “item number” field.

Item number 10
Identifier + Item number + Item name + Value of interval + Unit of interval + terminator
The other item number
Identifier + Item number + Item name + Value of item + terminator

Example

A	B	C	D	E
REM	Automatic Backup			
REM	Item number	Item name	value	!!!!(Terminator)
ATBK	1	Automatic Backup	ENABLE	!!!!
ATBK	2	Device	FRA:¥	!!!!
ATBK	3	Backup Time1	**:**	!!!!
ATBK	4	Backup Time2	**:**	!!!!
ATBK	5	Backup Time3	**:**	!!!!
ATBK	6	Backup Time4	**:**	!!!!
ATBK	7	Backup Time5	**:**	!!!!
ATBK	8	Backup at DI rising	0	!!!!
ATBK	9	Backup at Power up	ENABLE	!!!!
ATBK	10	Interval	7	DAY
ATBK	11	Backup in progress	0	!!!!
ATBK	12	Error occurs at backup	0	!!!!

Detail

The 2nd field, item number identifies the meaning of line data. The 3rd item, item name is just for explanation of the item. It is ignored when data is set from file.

Item number	Item name	Description
1	Automatic Backup	Enable or Disable of automatic backup. This is specified by ENABLE or DISABLE.
2	Device	This specifies file device and directory. File device is specified by “MC:¥” or “FRA:¥”. If subdirectory is used, Write device name and subdirectory name together. “MC:¥AGM¥” for example.
3-7	Backup Time 1-5	These fields are specified in 5 or 4 characters. Hours (1 or 2 characters) : minutes (2 characters) If minute is less than 10, add leading 0. 12:05 for example.

Item number	Item name	Description
8	Backup at DI rising	Index of signal is specified.
9	Backup at Power up	ENABLE or DISABLE
10	Interval	The line of item number 10 has 2 value fields. The 1st one is positive integer number that shows interval value. The 2nd value field decides unit of the value. The 2nd one is specified by MIN, HOUR or DAY.
11	Backup in progress	Index of output signal is specified.
12	Error occurs at backup	Index of output signal is specified.

Note

When backup time is uninitialized, it is output as “**:**”. If corresponding field is “**:**”, backup time is set to “**:**” as specified.

4.8.17 System Config

Identifier

SYSC

Syntax

The 2nd field is item number. It identifies what the item is. Format depends on “item number” field.
In general, format is as follows. However, there may be 2 or more value fields.

Identifier + item number + item name + value + terminator

Each item is described later

Example

A	B	C	D	E	F	G
REM	System config					
REM	Item number	Item name	value	!!!!(Terminator)		
SYSC	1	Use HOT START	ENABLE	!!!!		
SYSC	2	I/O power fail recovery	RECOVER ALL	!!!!		
SYSC	3	COLD START Autoexec program		!!!!		
SYSC	4	HOT START Autoexec program		!!!!		
SYSC	5	HOT START done signal	0	!!!!		
SYSC	6	Restore selected program	ENABLE	!!!!		
SYSC	7	Enable UI signals	ENABLE	!!!!		
SYSC	8	START for CONTINUE only	DISABLE	!!!!		
SYSC	9	CSTOPI for ABORT	DISABLE	!!!!		
SYSC	10	Abort all programs by CSTOPI	DISABLE	!!!!		
SYSC	11	PROD_START depend on PNSTROBE	DISABLE	!!!!		
SYSC	12	Detect FAULT_RESET signal	FALL	!!!!		
SYSC	13	Use PPABN signal	GP1	DISABLE	!!!!	
SYSC	14	WAIT timeout	30	!!!!		
SYSC	15	RECEIVE timeout	30	!!!!		
SYSC	16	Return to top of program	ENABLE	!!!!		
SYSC	17	Original program name	F1	RSR	!!!!	
SYSC	17	Original program name	F2	PNS	!!!!	
SYSC	17	Original program name	F3	STYLE	!!!!	
SYSC	17	Original program name	F4	JOB	!!!!	
SYSC	17	Original program name	F5	TEST	!!!!	
SYSC	18	Default logical command	F2		4	!!!!
SYSC	18	Default logical command	F3		*	!!!!

A	B	C	D	E	F	G
SYSC	18	Default logical command	F4		*	!!!!
SYSC	19	Maximum of ACC instruction	150	!!!!		
SYSC	20	Minimum of ACC instruction	0	!!!!		
SYSC	21	Auto display of alarm menu	DISABLE	!!!!		
SYSC	22	Force Message	ENABLE	!!!!		
SYSC	23	Allow Force I/O in AUTO mode	ENABLE	!!!!		
SYSC	24	Allow chg. ovrd. in AUTO mode	ENABLE	!!!!		
SYSC	25	Signal to set in AUTO mode	0	!!!!		
SYSC	26	Signal to set in T1 mode	0	!!!!		
SYSC	27	Signal to set in T2 mode	0	!!!!		
SYSC	28	Signal to set if E-STOP	0	!!!!		
SYSC	29	Set if INPUT SIMULATED	0	!!!!		
SYSC	30	Set if OUTPUT SIMULATED	0	!!!!		
SYSC	31	Sim. Input Wait Delay	0	!!!!		
SYSC	32	Set if SIm. Skip Enabled	0	!!!!		
SYSC	33	Set when prompt displayed	0	!!!!		
SYSC	34	WAIT for DI range	0	0	!!!!	
SYSC	35	WAIT for DI time	0	!!!!		
SYSC	36	WAIT for DI output	0	!!!!		
SYSC	37	Signal if OVERRIDE = 100	0	!!!!		
SYSC	38	Hand broken	GP1	ENABLE	!!!!	
SYSC	39	Remote/Local setup	OP PANEL KEY	!!!!		
SYSC	40	External I/O(ON:Remote)	DI	0	!!!!	
SYSC	41	Multi Program　Selection	DISABLE	!!!!		

Detail

- The 2nd field, item number identifies the meaning of line data.
- The 3rd field, item name is just for explanation of the item. It is ignored when data is set from file.
- Enable/disable is specified by ENABLE or DISABLE.
- Input in uppercase as specifying character string.

Item number	Item name	Description
1	Use HOT START	ENABLE or DISABLE.
2	I/O power fail recovery	This is specified by following character string. NOT RECOVER RECOVER SIM UNSIMULATE RECOVER ALL
3	COLD START Autoexec program	By default setting, maximum 36 characters.
4	HOT START Autoexec program	By default setting, maximum 36 characters.
5	HOT START done signal	Index of signal specified by 0 or more integer.
6	Restore selected program	ENABLE or DISABLE.
7	Enable UI signals	ENABLE or DISABLE.
8	START for CONTINUE only	ENABLE or DISABLE.
9	CSTOPI for ABORT	ENABLE or DISABLE.
10	Abort all programs by CSTOPI	ENABLE or DISABLE.
11	PROD_START depend on PNSTROBE	ENABLE or DISABLE.
12	Detect FAULT_RESET signal	RISE or FALL.

Item number	Item name	Description
13	USE PPABN signal	This line has 2 value fields. The 1st one is group number. - Only 3 characters are recognized. - The first 2 characters should be “GP”. - The last character is group number. - If group number is out of range when data is set, message “Illegal group” is displayed. The 2nd value is ENABLE or DISABLE
14	WAIT timeout	When data is output, the number of decimal places is 2. When you create CSV file to set data, do likewise.
15	RECEIVE timeout	When data is output, the number of decimal places is 2. When you create CSV file to set data, do likewise.
16	Return to top of program	ENABLE or DISABLE
17	Original program name	This line has 2 value fields. The 1st one is function key. - Only 2 characters are recognized. - The 1st character must be F. - The 2nd character specifies function key by its number. - One line of item number 17 defines program name for one function key only. - 5 line of item number 17 are needed to define program name from F1 to F5. The 2nd value field is “Original program name”. - Maximum 7 characters.
18	Default logical command	This line has 3 value fields. The 1st value field is function key. - Please refer to explanation of item 17. The 2nd value field is Name. - Maximum 8 characters. - Leading spaces are not removed. The 3rd value field is the number of lines. - This range is from 0 to 4. - If lines are not set yet, this value is output as “ *” (Space and asterisk). The character “ *” can be used to set the value when data is set from file.
19	Maximum of ACC instruction	This is specified by integer value of 0 or more. NOTE: If 501 or more is set, message “Illegal data” is displayed.
20	Minimum of ACC instruction	Please refer to item number 19.
21	Auto display of alarm menu	ENABLE or DISABLE.
22	Force Message	This is specified by following character string. DISABLE ENABLE ENBL(TP OFF)
23	Allow Force I/O in AUTO mode	ENABLE or DISABLE.
24	Allow chg ovrd in AUTO mode	ENABLE or DISABLE.
25	Signal to set in AUTO mode	This is specified by integer value of 0 or more.
26	Signal to set in T1 mode	This is specified by integer value of 0 or more.
27	Signal to set in T2 mode	This is specified by integer value of 0 or more.
28	Signal to set if E-STOP	This is specified by integer value of 0 or more.
29	Set if INPUT SIMULATED	This is specified by integer value of 0 or more.
30	Set if OUTPUT IMULATED	This is specified by integer value of 0 or more.
31	Sim Input Wait Delay	This is specified by real value. When data is output, the number of decimal places is 2. When you create CSV file to set data, do likewise.

Item number	Item name	Description
32	Set If Sim Skip Enabled	This is specified by integer value of 0 or more.
33	Set when prompt displayed	This is specified by integer value of 0 or more.
34	WAIT for DI range	This line has 2 value fields. The 1st one is start number. The 2nd one is end number
35	WAIT for DI time	When data is output, the number of decimal places is 2. When you create CSV file to set data, do likewise.
36	WAIT for DI output	This is specified by integer value of 0 or more.
37	Signal if OVERRIDE = 100	This is specified by integer value of 0 or more.
38	Hand broken	This line has 2 value fields. The 1st value field is group number. - Only 3 characters are recognized. - The first 2 characters should be “GP”. - The last character is group number. - If group number is out of range when data is set, message “Illegal group” is displayed. The 2nd value field is ENABLE/DISABLE.
39	Remote/Local setup	This is specified by following character string. REMOTE LOCAL EXTERNAL I/O OP PANEL KEY Please note that you cannot omit space in above string. All letters are capital.
40	External I/O(ON:Remote)	This line has 2 value fields. The 1st one is type of signal. Following character string can be used. DI DO RI RO UI UO The 2nd value field is index of signal. It is specified by integer value of 0 ore more.
41	Multi program Selection	ENABLE or DISABLE.

Following items of system config screen are not supported.
- WJNT for default motion
- Reset CHAIN FAILURE detection
- UOP auto assignment

4.8.18 Program Timer

Identifier
PTIM

Syntax
Identifier + Index + timer count value + comment + terminator

Example

A	B	C	D	E
REM	Program timer			
REM	Index	Timer count	Comment	!!!!(Terminator)
PTIM	1	25.9	Cycle time	!!!!

Detail

Item	Description
Index	Index of program timer. If this is out of range when data is set from file, message “Illegal index” is displayed.
Timer Count	When data is output, the number of decimal places is 2. When you create CSV file to set data, do likewise.
Comment	Maximum 13 characters.

Note

When file is output, all program timers are output.

4.8.19 String Register

Identifier

SREG

Syntax

Identifier + Index + value string + comment + terminator

Example

A	B	C	D	E
REM	String register			
REM	Index	String	Comment	!!!!(Terminator)
SREG	1	String value of SREG[1]	SREG[1] comment	!!!!

Detail

Item	Description
Index	Index of string register. If this is out of range when data is set from file, message “Illegal index” is displayed.
String	Value of string register. This function supports maximum 120 characters.
Comment	Maximum 16 characters.

Note

When file is output, all string registers are output.

4.8.20 Battery Alarm

Identifier

BLAL

Syntax

Identifier + ENABLE/DISABLE + terminator

Example

A	B	C
REM	Battery alarm	
REM	Consider battery of mechanical unit	!!!!(Terminator)
BLAL	ENABLE	!!!!

Detail

Item	Description
Consider battery of mechanical unit	ENABLE or DISABLE.

4.8.21 Override Select

Identifier

OVSL

Syntax

The 1st line

Identifier + ENABLE/DISABLE + Signal 1 + Signal 2 + terminator

The 2nd line

Empty field + override of OFF_OFF + override of OFF_ON + override of ON_OFF + override of ON_ON + terminator

The data set is always composed of 2 lines. You have to set above lines in this order and no lines can be omitted.

Example

A	B	C	D	E	F
REM	Override select				
REM	ENABLE/DISABLE	Signal 1	Signal 2	!!!!(Terminator)	
REM	OFF_OFF	OFF_ON	ON_OFF	ON_ON	!!!!(Terminator)
OVSL	DISABLE	****	****	!!!!	
	10	20	50	100	!!!!

Detail

Item	Description
ENABLE/DISABLE	Enable/disable of override select. This is specified by ENABLE or DISABLE.
Signal 1	Index of signal 1.
Signal 2	Index of signal 2.
OFF_OFF	Override is specified by integer. Range is from 1 to 100.
OFF_ON	
ON_OFF	
ON_ON	

Note

When index of signal 1 or signal 2 is uninitialized, it is output as “****”. When data is set from file and the field is “****”, corresponding setting is not change. Index is not set to “****”.

4.8.22 Warning/Reset in Alarm History

Identifier

NOHIS

Syntax

Identifier + WARN not recorded + RESET not recorded + terminator

Example

A	B	C	D
REM	Warn/Reset in alarm history		
REM	WARN not recorded	RESET not recorded	!!!!(Terminator)
NOHIS	0	0	!!!!

Detail

Item	Description
WARN not recorded	If alarm of WARN is not recorded, 1. Otherwise, 0

Item	Description
RESET not recorded	If reset is not recorded, 1. Otherwise, 0

4.8.23 TCP/IP

Identifier
TCPIP

Syntax
Identifier + item number + item name + value + terminator

Example

A	B	C	D	E
REM	TCP/IP			
REM	Item number	Item name	value	!!!!(Terminator)
TCPIP	1	Host name	ROBOT	!!!!
TCPIP	2	IP address(port1)	****	!!!!
TCPIP	3	IP address(port2)	****	!!!!
TCPIP	4	Subnet Mask(port1)	255.255.255.0	!!!!
TCPIP	5	Subnet Mask(port2)	255.255.255.0	!!!!
TCPIP	6	Router IP Address	****	!!!!

Detail
- The 2nd field, item number identifies the meaning of line data.
- The 3rd field, item name is just for explanation of the item. It is ignored when data is set from file.

Item number	Item name	Description
1	Host name	Host name or robot controller. Maximum 18 characters.
2	IP address(port1)	IP address of port1. This function doesn’t check if it is valid address or not.
3	IP address(port2)	IP address of port2. This function doesn’t check if it is valid address or not.
4	Subnet Mask (port1)	Subnet mask of port1. This function doesn’t check if it is valid mask or not.
5	Subnet Mask (port2)	Subnet mask of port2. This function doesn’t check if it is valid mask or not.
6	Router IP Address	IP address of router. This function doesn’t check if it is valid address or not.

Note
- If robot controllers of same address exist, it can cause network trouble. Please pay attention to it when you set data from file.
- Uninitialized IP address of controller and Router is output as “****”. When data is set from file, “****” is not set.
- This function assumes host name is composed of alphabet and numeric characters.

4.8.24 TCP/IP Host (Local)

Identifier
TCPIP_HL

Syntax
Identifier + index + host name + IP address + terminator

Example

A	B	C	D	E
REM	TCP/IP Local host			

A	B	C	D	E
REM	Index	Host name	IP address	!!!!(Terminator)
TCPIP_HL	1	TestHost	192.168.0.1	!!!!
TCPIP_HL	2	****	****	!!!!

Detail

Item	Description
Index	Index of setting.
Host name	Maximum 10 characters
IP address	IP address of host. This function does not check if it is valid address or not.

Note

- Uninitialized IP address is output as “****”. When data is set from file, “****” is not set.
- This function assumes host name is composed of alphabet and numeric characters.

4.8.25 Error Table

Identifier

ER_TBL

Syntax

Identifier + index + ID + error code + Severity + Display + terminator

Example

File output adds comment lines that show relationship between ID (Facility code) and kinds of alarm (TPIF, SYST for example). The number of these lines depend on the version of system software of robot controller.

A	B	C	D	E	F	G	I	J	K	L
REM	Error Table									
REM	Facility Code list									
REM	0	OS	1	SRIO	2	FILE	3	PROG	4	COND
REM	5	ELOG	6	MCTL	7	MEMO	8	GUID	9	TPIF
REM	10	FLPY	11	SRVO	12	INTP	13	PRIO	14	TPAX
REM	15	MOTN	16	VARS	17	ROUT	18	WNDW	19	JOG
REM	20	APPL	21	LANG	22	ASBN	23	SPOT	24	SYST
REM	25	SCIO	26	PALT	27	UAPL	28	PMON	29	TOOL
REM	30	SVGN	31	PWD	32	VISN	33	DICT	34	KCLI
REM	35	TRAN	36	TKSP	37	COPT	38	APSH	39	ISD
REM	40	DMER	41	MHND	42	CMND	43	RPM	44	LNTK
REM	45	WEAV	46	TCPP	47	TAST	48	MUPS	49	MIGE
REM	50	LSR	51	SEAL	52	PAIN	53	ARC	54	TRAK
REM	55	CALB	56	SP	57	MACR	58	SENS	59	COMP
REM	60	THSR	61	QMGR	62	ELSE	63	ELSE	64	DJOG
REM	65	OPTN	66	HRTL	67	HOST	68	MEnet	69	SSPC
REM	70	FIG	71	LDG	72	DX	73	CNTR	74	LODC
REM	75	PFMS	76	DNET	77	GBOX	78	PAL2	79	SHAP
REM	80	ATCP	81	CART	82	CD	83	MARL	84	DMDR
REM	85	FRSY	86	PNT1	87	FLEX	88	IB-S	89	RTCP
REM	90	TG	91	FORC	92	PROF	93	RPC	94	VC
REM	95	ISDT	96	GUI	97	CUST	98	POST	99	ELSE
REM	100	NTOS	101		102	ATGP	103	WMAP	104	ASST

A	B	C	D	E	F	G	I	J	K	L
REM	105	FIND	106	CALM	107		108	LSTP	109	LECO
REM	110	MLOG	111	APC	112	ACAL	113	TPCL	114	CPMO
REM	115	PALL	116	TJOG	117	CVIS	118	BBOX	119	TMAT
REM	120	TANG	121	DPMO	122	KALM	123	SDTL	124	ICRZ
REM	125		126		127		128	ELSE	129	XMLF
REM	130	RIPE	131	SPRM	132		133		134	
REM	135	SVTL	136	FXTL	137	PNT2	138	ATZN	139	VDIG
REM	140	PTPG	141	SYMR	142	TEST	143	SENC	144	BRCH
REM	145	FPLN	146	NCBN	147		148		149	DTBR
REM	150	*****								
REM	Index	Facility Code	Error Code	Severity	Display	!!!!				
ER_TBL	1	11	1	DEFAULT	DEFAULT	!!!!				
ER_TBL	2	50	1	STOP	DEFAULT	!!!!				
ER_TBL	3	50	2	DEFAULT	DEFAULT	!!!!				
ER_TBL	4	50	3	ABORT	DEFAULT	!!!!				
ER_TBL	5	50	4	DEFAULT	DEFAULT	!!!!				
ER_TBL	6	50	5	DEFAULT	DEFAULT	!!!!				
ER_TBL	7	50	6	DEFAULT	DEFAULT	!!!!				
ER_TBL	8	50	7	DEFAULT	DEFAULT	!!!!				
ER_TBL	9	50	8	DEFAULT	DEFAULT	!!!!				
ER_TBL	10	****	0	DEFAULT	DEFAULT	!!!!				

Detail

Item	Description
Index	Index of setting. If index is out of range when data is set, message “Illegal index” is displayed
ID	This is value that shows the kind of alarm. For example, 11 means SRVO alarm. Please refer to error table screen of controller and comment of output CSV file.
Error code	Number of error. 14 of TPIF-14 for example.
Severity	This is specified by following character string. DEFAULT STOP STOPALL ABORT ABORTALL
Display	This is specified by following character string. DEFAULT ACTIVE NODISP NOERLOG NOERLIN

Note

Uninitialized ID is output as “****”. When data is set from file, lines whose ID is “****” are ignored.

4.8.26 Alarms That Do Not Turn on FAULT Signal

Identifier

ER_NOALM

Syntax

Identifier + item number + item name + value + terminator

Example

A	B	C	D	E
REM	ER_NO_ALM			
REM	Item number	Item name	value	!!!!(Terminator)
ER_NOALM	1	No alarm enable(1/0)	0	!!!!
ER_NOALM	2	Number of error code	5	!!!!
ER_NOALM	3	Error code 1	11001	!!!!
ER_NOALM	4	Error code 2	11002	!!!!
ER_NOALM	5	Error code 3	11003	!!!!
ER_NOALM	6	Error code 4	11007	!!!!
ER_NOALM	7	Error code 5	11037	!!!!
ER_NOALM	8	Error code 6	0	!!!!
ER_NOALM	9	Error code 7	0	!!!!
ER_NOALM	10	Error code 8	0	!!!!
ER_NOALM	11	Error code 9	0	!!!!
ER_NOALM	12	Error code 10	0	!!!!

Detail

- The 2nd field, item number identifies the meaning of line data.
- The 3rd field, item name is just for explanation of the item. It is ignored when data is set from file.

Item number	Item name	Description
1	No alarm enable(1/0)	Enable/Disable of the function. 0 means that the function is disabled. 1 means enabled.
2	Number of error code	The number of error codes that do not turn on FAULT signal.
3-42	Error code 1-40	Error code that does not turn on FAULT signal. ID (facility code) that means kind of error *1000 + error number. For example, 11007 corresponds to SRVO-007. Please refer to error table screen of controller or comment just before ER_TBL of output CSV file to know relationship between ID and kind of alarm.

4.8.27 RSR

Identifier

RSR

Syntax

The 2nd field is item number. It identifies what the item is. Format depends on “item number” field.
In general, format is as follows. However, there may be 2 or more value fields.

Identifier + item number + item name + value + terminator

Example

A	B	C	D	E	F
REM	RSR				
REM	Item number	Item name	value	!!!!(Terminator)	
RSR	1	RSR1 program number	ENABLE	0	!!!!

A	B	C	D	E	F
RSR	2	RSR2 program number	ENABLE	0	!!!!
RSR	3	RSR3 program number	ENABLE	0	!!!!
RSR	4	RSR4 program number	ENABLE	0	!!!!
RSR	5	RSR5 program number	DISABLE	0	!!!!
RSR	6	RSR6 program number	DISABLE	0	!!!!
RSR	7	RSR7 program number	DISABLE	0	!!!!
RSR	8	RSR8 program number	DISABLE	0	!!!!
RSR	21	Job prefix	RSR	!!!!	

Detail

- Among setting of RSR, settings that are related to RSR only are included in identifier “RSR”. Acknowledge function, Acknowledge pulse width and Base number are included in identifier “PRG_SEL”
- The 2nd field, item number identifies the meaning of line data.
- The 3rd field, item name is just for explanation of the item. It is ignored when data is set from file.

Item number	Item name	Description
1	RSR1 program number	This line has 2 value fields. The 1st one is specified by ENABLE or DISABLE. The 2nd one is program number. - This is specified by integer value of 0 or more. - There is no maximum number check. But specify 9999 or less.
2	RSR2 program number	Please refer to item number 1, RSR1 program number.
3	RSR3 program number	
4	RSR4 program number	
5	RSR5 program number	
6	RSR6 program number	
7	RSR7 program number	
8	RSR8 program number	
21	Job prefix	Only 3 characters are recognized.

4.8.28 PNS

Identifier

PNS

Syntax

Identifier + Job prefix + terminator

Example

A	B	C
REM	PNS	
REM	Job prefix	!!!!(Terminator)
PNS	PNS	!!!!

Detail

- Among setting of PNS, settings that are related to PNS only are included in identifier “PNS”. Base number and Acknowledge pulse width are included in identifier “PRG_SEL”

Item	Description
Job prefix	Only 3 characters are recognized.

4.8.29 Program Selection Common

Identifier

PRG_SEL

Syntax

The 1st line

Identifier + Program select mode + terminator

The 2nd line

Empty field + Base number + Acknowledge function + Acknowledge pulse + terminator

The data set is always composed of 2 lines. You have to set above lines in this order and no lines can be omitted.

Example

A	B	C	D	E
REM	Program Selection common			
REM	Program select mode	!!!!(Terminator)		
REM	Base number	Acknowledge function	Acknowledge pulse	!!!!(Terminator)
PRG_SEL	PNS	!!!!		
	0	DISABLE	400	!!!!

Detail

Item	Description
Program Selection mode	This is specified by following character string. RSR PNS STYLE If the other value is used when data is set from file, message "Illegal data " is displayed. "OTHER" is not supported.
Base number	Base number of RSR and PNS. This is specified by integer value of 0 or more.
Acknowledge function	This field corresponds to "Acknowledge function" of RSR and Style. This is specified by ENABLE or DISABLE.
Acknowledge pulse	This field corresponds to Acknowledge pulse width of RSR, PNS and Style.

4.8.30 *i*Pendant Automatic Blanking of Backlight

Identifier

BLNK

Syntax

Identifier + ENABLE/DISABLE + Timer of blanking + terminator

Example

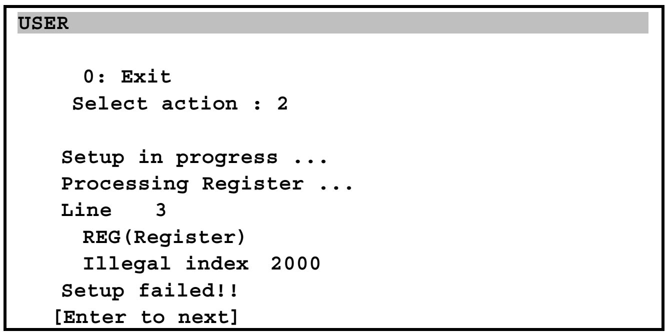
A	B	C	D
REM	iPendant automatic blanking		
REM	ENABLE/DISABLE	Timer of blanking(minute)	!!!!(Terminator)
BLNK	DISABLE	5	!!!!

Detail

Item	Description
ENABLE/DISABLE	Enable/Disable of automatic blanking of *i*Pendant backlight. This is specified by ENABLE or DISABLE.
Timer of blanking (minute)	This is specified by integer value. Unit is minute. Minimum value is 5. Maximum value is 1000.

4.9 TIPS FOR TROUBLE SHOOTING

When you use this function, the main cause of errors is data set from wrong formatted CSV. When you encounter errors at data setting, please confirm contents of CSV file.
If format error is found, line number and error information is displayed.

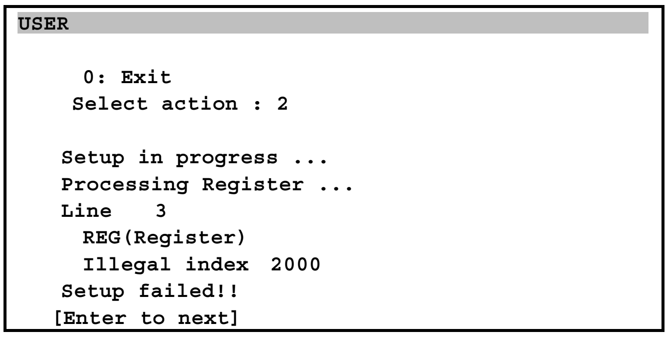


```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing Register ...
   Line    3
     REG(Register)
     Illegal index  2000
   Setup failed!!
  [Enter to next]
```

(A) "Illegal index"

This means wrong index is specified. For example, if you specify register index that does not exist, it can causes this error.
In following example, register 2000 is specified. If it does not exist, error message is displayed when data is set from file.

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,2000,This is integer,INT,0,!!!!
```



```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing Register ...
   Line    3
     REG(Register)
     Illegal index  2000
   Setup failed!!
  [Enter to next]
```

(B) Illegal terminator

When terminator (!!!!) is expected at the end of line and it is not found, it causes error. In following example, "!!!!" is replaced by "!!!".

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,1, !!! instead !!!!,INT,0,!!!
```

```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing Register ...
   Line    3
     REG(Register)
     Illegal terminator!!!
   Setup failed!!
  [Enter to next]
```

In following example, no string exists where terminator, “!!!!” is expected.

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,     4,No !!!! is found,INT,             0
```

```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing Register ...
   Line    3
     REG(Register)
     Illegal terminator    (No character)
   Setup failed!!
  [Enter to next]
```

(C) Illegal data

This is the most common error message that is displayed when unexpected data is found at setting data from file. Following cases are examples.

- String is found where numeric value is expected.

```
REG, ABC, illegal index,INT,             0, !!!!
```

- Negative value is found where integer value that is 0 or more is expected.

```
REG, -1, illegal index,INT,             0, !!!!
```

- Unexpected value is found where keyword, ENABLE for example, is expected

```
BLNK,Neither DISABLE nor ENABLE,5,!!!!
```

```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing iPendant automatic blanking .
   Line    3
     BLNK(iPendant automatic blanking)
     Illegal data Neither ENABLE nor DISABLE
   Setup failed!!
  [Enter to next]
```

(D) Illegal group

Wrong formatted data is found where group number should be specified by “GPn” (n is group number). Following example causes error if it is used for 1 group system.

```
SYSC,    38,Hand broken,GP2,ENABLE,!!!!
```

```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing System config...
   Line    1
     SYSC(System config)
     Illegal group 2
   Setup failed!!
  [Enter to next]
```

(E) Unknown data type

Proper identifier is not found where it is expected. In following example, INVALID is illegal identifier.

```
INVALID,    38,Hand broken,GP1,ENABLE,!!!!
```

```
USER
     2: Set data from cp_0011.csv
     3: Change CSV file name
     0: Exit
    Select action : 2

   Setup in progress ...
   Line    1
    Unknown data type INVALID
   Setup failed!!
  [Enter to next]
```

(F) Illegal config

Position register of Cartesian data type specifies configuration data. If string of configuration is wrong, this message is displayed. In following example, string of configuration of position register 2 is wrong.

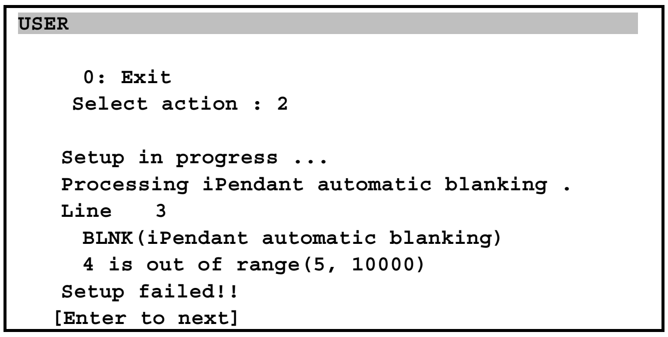
```
PREG,     1,This is correct,!!!!
,GP1,XYZ,0,!!!!
, 1592.000,       -.000, 1085.000,!!!!
,    180.000,       -.000,         .000,!!!!
,"N U T, 0, 0, 0",!!!!
PREG,     2,This is wrong,!!!!
,GP1,XYZ,0,!!!!
, 1592.000,       -.000, 1085.000,!!!!
,    180.000,       -.000,         .000,!!!!
,"M U T, 0, 0, 0",!!!!
PREG,     3,,!!!!
,GP1,XYZ,U,!!!!
```

```
USER

   Setup in progress ...
   Processing Register ...
   CNV?STR_CONF failed: 15121
   Line    18
     PREG(Position register)
     Illegal config.
   N U T, 0, 0, 0
   Setup failed!!
  [Enter to next]
```

(G) %d is out of range(min, max)

This message is displayed when integer value displayed in place of %d is out of range. You may see the other data, for example “illegal data” instead. In following example, 4 minute is specified as timer count of *i*Pendant automatic blanking. However, minimum value is 5.

```
REM,iPendant automatic blanking
REM,ENABLE/DISABLE,Timer of blanking(minute),!!!!(Terminator)
BLNK,DISABLE,4,!!!!
```

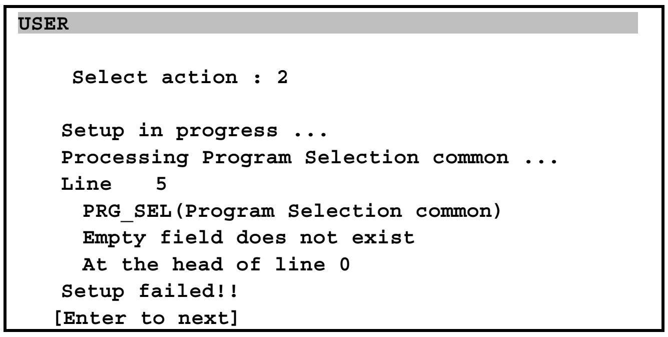


```
USER

     0: Exit
    Select action : 2

   Setup in progress ...
   Processing iPendant automatic blanking .
   Line   3
     BLNK(iPendant automatic blanking)
     4 is out of range(5, 10000)
   Setup failed!!
  [Enter to next]
```

(H) Empty field does not exist at the head of line

If data for an identifier is composed of 2 or more lines, the 2nd or later lines must start with empty field (field that does not have character). This message is displayed the empty field is not found. In following example, the 2nd line starts with “0” because empty field is lost. It should be empty field.

```
REM,Program Selection common
REM,Program select mode,!!!!(Terminator)
REM,Base number,Acknowledge function,Acknowledge pulse width,!!!!(Terminator)
PRG_SEL,PNS,!!!!
0,DISABLE, 400,!!!!
```



```
USER

    Select action : 2

   Setup in progress ...
   Processing Program Selection common ...
   Line   5
     PRG_SEL(Program Selection common)
     Empty field does not exist
     At the head of line 0
   Setup failed!!
  [Enter to next]
```

4.10 CAUTIONS FOR SETTING DATA FROM FILE

Removal of space

When data is set from file, leading and trailing spaces are removed in most cases. Following cases are examples.

(A) Fields that expect numeric value
(B) Fields that expect keywords (ENABLE for example)
(C) Fields that expect program name
(D) Empty field at the head of line

Following cases are example of field whose leading and trailing spaces are not removed.

(A) Comment

(B) Macro name
(C) Host name
(D) Message of user alarm

Read of keyword

Many fields expect keywords like ENABLE and DISABLE. Because keywords are defined in capital letters, please write keywords in capital letters. When ENABLE is expected, enable is not allowed.

The number of settings

Robot controller can change the number of some kinds of setup data. For example, you can change the number of registers. This function does not change the number of such data. If you copy setting of a robot controller to another one by this function, please note that make the number of settings match by yourself.

Use of 2byte character at the end of comment

Comment has limited length. If the last byte is the 1st byte of Chinese character, it is replaced to space. For example, maximum length of register comment is 16. If 16 and 17 bytes of comment fields is Chinese character, space is set as the 16th byte.

Set of real number

When real number is set from CSV file, there is precision error.

Scientific notation

Scientific notation, 9.999E+5 for example, is not supported.

4.11 EDIT BY Microsoft® Excel®

CSV file can be edited by Microsoft® Excel®. However, data that you do not edit can be changed automatically. Please confirm that data is correct.

4.11.1 String That is Recognized as Long Numeric Value

Suppose that position 3 (P[3]) of program PNS0001 has comment "1234567890123456".
This function output the data as follows.

```
REM,Comment of Position data
REM,Program name,Position number,Comment,!!!!(Terminator)
POS,PNS0001                                    ,     3,1234567890123456,!!!!
```

However, if it is opened by Microsoft® Excel®, you may see data like following table.

A	B	C	D	E
REM	Comment of Position data			
REM	Program name	Position number	Comment	!!!!(Terminator)
POS	PNS0001	3	1.23457E+15	!!!!

A, B, C… of first line is column number of Microsoft® Excel®.

If file is saved in this state, comment is changed even though user does not change it.

```
REM,Comment of Position data,,,
REM,Program name,Position number,Comment,!!!!(Terminator)
POS,PNS0001                                    ,3,1.23457E+15,!!!!
```

Scientific notation is not supported. Please confirm that scientific notation is not used if you have this kind of data.

4.12 MORE INFORMATION ABOUT CSV FILE FORMAT

Line of CSV file

ASCII data written in CSV file is separated by "," (comma). At the end of line, "," does not exist. "CR, LF" is recorded instead. Both CR and LF are control code. CR is carriage return. LF is line feed. If you use "," in data itself, data should be sandwiched by " (double quotation). If " (double quotation) is used in data, data should be sandwiched by " and you should add double quotation after double quotation as data.

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,     1,work count,INT,              0,!!!!
REG,     2,"work1, work2",REAL,       1.230,!!!!
REG,     3,"Work ""A"" done",INT,              0,!!!!
```

Display by Microsoft® Excel®

A	B	C	D	E	F
REM	Register				
REM	Index	Comment	INT/REAL	value	!!!!(Terminator)
REG	1	work count	INT	0	!!!!
REG	2	wrok1, work2	REAL	1.23	
REG	3	Work "A" done	INT	0	

End of CSV file

CSV file needs END "CR, LF" line. Following is sample of text file data.

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,     1,work count,INT,              0,!!!!
REG,     2,"work1, work2",REAL,       1.230,!!!!
REG,     3,"Work ""A"" done",INT,              0,!!!!
REG,     4,,INT,              0,!!!!

REM,PNS
REM,Job prefix,!!!!(Terminator)
PNS,PNS,!!!!

REM,Program Selection common
REM,Program select mode,!!!!(Terminator)
REM,Base number,Acknowledge function,Acknowledge pulse width,!!!!(Terminator)
PRG_SEL,PNS,!!!!
, 0,DISABLE, 400,!!!!

REM,iPendant automatic blanking
REM,ENABLE/DISABLE,Timer of blanking(minute),!!!!(Terminator)
BLNK,DISABLE,5,!!!!

END
```

4.12.1 Example of Whole File

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,     1,work counter,INT,              0,!!!!

REM,Position register
REM,Index,Comment,!!!!(Terminator)
REM,Group(GPn),Type,The number of extended axis/total axis,!!!!(Terminator)
REM,X/J1,Y/J2,Z/J3,!!!!(Terminator)
REM,W/J4,P/J5,R/J6,!!!!(Terminator)
REM,E1,E2,E3,!!!!(Terminator)
REM,Config,!!!!(Terminator)
PREG,     1,The 1st home pos,!!!!
,GP1,JPOS,6,!!!!
,         .000,    -51.477,    -22.417,!!!!
,         .000,    -67.583,       -.000,!!!!

REM,I/O Comment (DO)
REM,Index,Comment,!!!!(Terminator)
DO,     1,Door 1 open,!!!!

REM,User alarm
REM,Index,Alarm message,Alarm severity,!!!!(Terminator)
UALM,1,Door 1 is open,WARN,!!!!
UALM,2,This is STOP.L,STOP.L,!!!!
UALM,3,This is STOP.G,STOP.G,!!!!
UALM,4,This is ABORT.L,ABORT.L,!!!!
UALM,5,This is ABORT.G,ABORT.G,!!!!
UALM,6,,STOP.L,!!!!
UALM,7,,STOP.L,!!!!
UALM,8,,STOP.L,!!!!
UALM,9,,STOP.L,!!!!
UALM,10,,STOP.L,!!!!

END
```

4.12.2 Comment (REM) in CSV File

```
REM, This is example of comment
REM, When this function output CSV file
REM, Comment to show format is added
```

4.12.3 END of CSV File

```
END
```

4.12.4 Register

```
REM,Register
REM,Index,Comment,INT/REAL,value,!!!!(Terminator)
REG,     1,This is integer,INT,              0,!!!!
REG,     2,This is real,REAL,       1.230,!!!!
```

4.12.5 Position Register

```
REM,Position register
REM,Index,Comment,!!!!(Terminator)
REM,Group(GPn),Type,The number of extended axis/total axis,!!!!(Terminator)
REM,X/J1,Y/J2,Z/J3,!!!!(Terminator)
REM,W/J4,P/J5,R/J6,!!!!(Terminator)
REM,E1,E2,E3,!!!!(Terminator)
REM,Config,!!!!(Terminator)
PREG,    1,Ex. cartesian,!!!!
,GP1,RXYZ,3,!!!!
, 1807.000,      0.000, 1300.000,!!!!
,   180.000,   -90.000,      0.000,!!!!
,      0.000,      0.000,      0.000,!!!!
,"N U T, 0, 0, 0",!!!!
,GP2,XYZ,0,!!!!
, 1807.000,      0.000, 1300.000,!!!!
,   180.000,   -90.000,      0.000,!!!!
,"N U T, 0, 0, 0",!!!!
PREG,    2,Ex. joint pos,!!!!
,GP1,JPOS,9,!!!!
,      1.000,      2.000,      3.000,!!!!
,      4.000,      5.000,      6.000,!!!!
,      7.000,      8.000,      9.000,!!!!
,GP2,JPOS,6,!!!!
,    10.000,    20.000,    30.000,!!!!
,    40.000,    50.000,    60.000,!!!!
PREG,    3,,!!!!
,GP1,RXYZ,U,!!!!
,GP2,XYZ,U,!!!!
```

4.12.6 Comment of I/O

```
REM,I/O Comment (DI)
REM,Index,Comment,!!!!(Terminator)
DI,     1,Door 1 open,!!!!
```

4.12.7 Assignment of I/O

Following text is an example of DI. DO, UI and UO have similar format.

```
REM,I/O Assign   (DI)
REM,Index,Rack number,Slot number,Start point,Number of points,!!!!(Terminator)
DIA,1,0,1,19,8,!!!!
```

Following text is an example of GI. GO has similar format.

```
REM,I/O Assign   (GI)
REM,Index,Rack number,Slot number,Start point,Number of points,!!!!(Terminator)
GIA,5,0,2,25,10,!!!!
```

Following text is an example AI. AO has similar format.

```
REM,I/O Assign   (AI)
REM,Index,Rack number,Slot number,Channel,Number of points,!!!!(Terminator)
AIA,1,0,1,1,1,!!!!
```

4.12.8 Comment of Program

```
REM,Program
REM,Program name,Type,Comment,!!!!(Terminator)
PRG,PNS0001                                    ,TP,Body 1,!!!!
```

4.12.9 Comment of Position (P[ ]) of Program

```
REM,Comment of Position data
REM,Program name,Position number,Comment,!!!!(Terminator)
POS,PNS0001                                    ,     1,Home position,!!!!
POS,PNS0001                                    ,     2,,!!!!
```

4.12.10 User Alarm

```
REM,User alarm
REM,Index,Alarm message,Alarm severity,!!!!(Terminator)
UALM,1,This is WARN,WARN,!!!!
UALM,2,This is STOP.L,STOP.L,!!!!
UALM,3,This is STOP.G,STOP.G,!!!!
UALM,4,This is ABORT.L,ABORT.L,!!!!
UALM,5,This is ABORT.G,ABORT.G,!!!!
```

4.12.11 MACRO

```
REM,Macro
REM,Index,Macro name,Program name,Assign type,Index,!!!!(Terminator)
MCR,     1,Sample macro1,AA_01,NO, 0,!!!!
MCR,     2,Sample macro2,AA_02,UK, 1,!!!!
MCR,     3,Sample macro3,AA_03,SU, 2,!!!!
MCR,     4,Sample macro4,AA_04,MF, 1,!!!!
MCR,     5,Sample macro5,AA_05,SP, 4,!!!!
MCR,     6,Sample macro6,AA_06,DI, 1,!!!!
MCR,     7,Sample macro7,AA_07,RI, 1,!!!!
MCR,     8,Sample macro8,AA_08,UI, 7,!!!!
MCR,     9,Sample macro9,AA_09,F, 1,!!!!
MCR,    10,Sample macro10,AA_10,M, 1,!!!!
```

4.12.12 Frame

```
REM,Tool Frame
REM,Group(GPn),Index,Comment,!!!!(Terminator)
REM,X,Y,Z,!!!!(Terminator)
REM,W,P,R,!!!!(Terminator)
TFRM,GP1, 1,Eoat1,!!!!
,       0.000,       0.000,       0.000,!!!!
,       0.000,       0.000,       0.000,!!!!
```

4.12.13 Reference Position

```
REM,Reference position
REM,Group(GPn),Index,Comment,ENABLE/DISABLE,Is a valid Home,Signal type,Signal Index,!!!!(Terminator)
REM,J1-J9 position,Tolerance,!!!!(Terminator)
REFP,GP1,1,Base 1 start,DISABLE,DISABLE,DO,     2,!!!!
,      0.000,      2.000,!!!!
,    -40.000,      2.000,!!!!
,      0.000,      2.000,!!!!
,      0.000,      2.000,!!!!
,    -90.000,      2.000,!!!!
,      0.000,      2.000,!!!!
,      0.000,      0.000,!!!!
,      0.000,      0.000,!!!!
,      0.000,      0.000,!!!!
```

4.12.14 Payload

```
REM,Payload
REM,Group(GPn),Index,Comment,Payload[kg],!!!!(Terminator)
REM,Center X,Cneter Y,Center Z,!!!!(Terminator)
REM,Inertia X,Inertia Y,Inertia Z,!!!!(Terminator)
PLD,GP1, 1,Payload Example,   100.00,!!!!
,     10.00,     20.00,     30.00,!!!!
,      4.00,      5.00,      6.00,!!!!
```

4.12.15 Armload

```
REM,Arm load
REM,Group(GPn),ARM LOAD AXIS#1,ARM LOAD AXIS#3,!!!!(Terminator)
ALD,GP1,     10.00,     10.23,!!!!
```

4.12.16 Space Check Function

```
REM,Space check function
REM,Group(GPn),Index,Comment,!!!!(Terminator)
REM,ENABLE/DISABLE,Output Type,Output Index,Input Type,Input Index,!!!!(Terminator)
REM,Priority(HIGH/LOW),INSIDE/OUTSIDE,length/vertex,UF n,UT n,!!!!(Terminator)
REM,Base X,Base Y,Base Z,!!!!(Terminator)
REM,2nd/Length X,2nd/Length Y,2nd/Length Z,!!!!(Terminator)
RSP,GP1, 1,Space1,!!!!
,DISABLE,DO,   10,DI,   12,!!!!
,HIGH,INSIDE,LENGTH,   0,   1,!!!!
,   1600.0,     500.0,   1200.0,!!!!
,    150.0,     200.0,    300.0,!!!!
```

4.12.17 Automatic Backup

```
REM,Automatic Backup
REM,Item number,Item name,value,!!!!(Terminator)
ATBK,     1,Automatic Backup,ENABLE,!!!!
ATBK,     2,Device,MC:¥SUB2¥,!!!!
ATBK,     3,Backup Time 1,**:**,!!!!
ATBK,     4,Backup Time 2,**:**,!!!!
ATBK,     5,Backup Time 3,**:**,!!!!
ATBK,     6,Backup Time 4,**:**,!!!!
ATBK,     7,Backup Time 5,**:**,!!!!
ATBK,     8,Backup at DI rising,       0,!!!!
ATBK,     9,Backup at Power up,ENABLE,!!!!
ATBK,    10,Interval,       7,DAY,!!!!
ATBK,    11,Backup in progress,       0,!!!!
ATBK,    12,Error occurs at backup,       0,!!!!
```

4.12.18 System Config

```
REM,System config
REM,Item number,Item name,value,!!!!(Terminator)
SYSC,     1,Use HOT START,ENABLE,!!!!
SYSC,     2,I/O power fail recovery,RECOVER ALL,!!!!
SYSC,     3,COLD START Autoexec program,,!!!!
SYSC,     4,HOT START Autoexec program,,!!!!
SYSC,     5,HOT START done signal,       0,!!!!
SYSC,     6,Restore selected program,ENABLE,!!!!
SYSC,     7,Enable UI signals,ENABLE,!!!!
SYSC,     8,START for CONTINUE only,DISABLE,!!!!
SYSC,     9,CSTOPI for ABORT,DISABLE,!!!!
SYSC,    10,Abort all programs by CSTOPI,DISABLE,!!!!
SYSC,    11,PROD_START depend on PNSTROBE,DISABLE,!!!!
SYSC,    12,Detect FAULT_RESET signal,FALL,!!!!
SYSC,    13,Use PPABN signal,GP1,DISABLE,!!!!
SYSC,    14,WAIT timeout,    30.00,!!!!
SYSC,    15,RECEIVE timeout,    30.00,!!!!
SYSC,    16,Return to top of program,ENABLE,!!!!
SYSC,    17,Original program name,F1,RSR,!!!!
SYSC,    17,Original program name,F2,PNS,!!!!
SYSC,    17,Original program name,F3,STYLE,!!!!
SYSC,    17,Original program name,F4,JOB,!!!!
SYSC,    17,Original program name,F5,TEST1234,!!!!
SYSC,    18,Default logical command,F2,, 4,!!!!
SYSC,    18,Default logical command,F3,, *,!!!!
SYSC,    18,Default logical command,F4,, *,!!!!
SYSC,    19,Maximum of ACC instruction,    150,!!!!
SYSC,    20,Minimum of ACC instruction,       0,!!!!
SYSC,    21,Auto display of alarm menu,DISABLE,!!!!
SYSC,    22,Force Message,ENABLE,!!!!
SYSC,    23,Allow Force I/O in AUTO mode,ENABLE,!!!!
SYSC,    24,Allow chg. ovrd. in AUTO mode,ENABLE,!!!!
SYSC,    25,Signal to set in AUTO mode,       0,!!!!
SYSC,    26,Signal to set in T1 mode,       0,!!!!
SYSC,    27,Signal to set in T2 mode,       0,!!!!
SYSC,    28,Signal to set if E-STOP,       0,!!!!
SYSC,    29,Set if INPUT SIMULATED,       0,!!!!
SYSC,    30,Set if OUTPUT SIMULATED,       0,!!!!
SYSC,    31,Sim. Input Wait Delay,     0.00,!!!!
SYSC,    32,Set if SIm. Skip Enabled,       0,!!!!
SYSC,    33,Set when prompt displayed,       0,!!!!
SYSC,    34,WAIT for DI range,       0,       0,!!!!
SYSC,    35,WAIT for DI time,     0.00,!!!!
SYSC,    36,WAIT for DI output,       0,!!!!
SYSC,    37,Signal if OVERRIDE = 100,       0,!!!!
SYSC,    38,Hand broken,GP1,ENABLE,!!!!
SYSC,    39,Remote/Local setup,OP PANEL KEY,!!!!
SYSC,    40,External I/O(ON:Remote),DI,       0,!!!!
SYSC,    41,Multi Program   Selection,DISABLE,!!!!
```

4.12.19 Program Timer

```
REM,Program timer
REM,Index,Timer count,Comment,!!!!(Terminator)
PTIM,     1,         12.50,Cycle time,!!!!
```

4.12.20 String Register

```
REM,String register
REM,Index,String,Comment,!!!!(Terminator)
SREG,     1,String value of SREG[1],SREG[1] comment,!!!!
```

4.12.21 Battery Alarm

```
REM,Battery alarm
REM,Consider battery of mechanical unit,!!!!(Terminator)
BLAL,ENABLE,!!!!
```

4.12.22 Override Select

```
REM,Override select
REM,ENABLE/DISABLE,Signal 1,Signal 2,!!!!(Terminator)
REM,OFF_OFF,OFF_ON,ON_OFF,ON_ON,!!!!(Terminator)
OVSL,DISABLE, 1, 2,!!!!
, 10, 20, 50, 100,!!!!
```

4.12.23 Warn/Reset in Alarm History

```
REM,Warn/Reset in alarm history
REM,WARN not recorded,RESET not recorded,!!!!(Terminator)
NOHIS,0,0,!!!!
```

4.12.24 TCP/IP

```
REM,TCP/IP
REM,Item number,Item name,value,!!!!(Terminator)
TCPIP,     1,Host name,ROBOT,!!!!
TCPIP,     2,IP address(port1),192.168.0.1,!!!!
TCPIP,     3,IP address(port2),****,!!!!
TCPIP,     4,Subnet Mask (port1),255.255.255.0,!!!!
TCPIP,     5,Subnet Mask (port2),255.255.0.0,!!!!
TCPIP,     6,Router IP Address,192.168.0.2,!!!!
```

4.12.25 TCP/IP Host (Local)

```
REM,TCP/IP Local host
REM,Index,Host name,IP Address,!!!!(Terminator)
TCPIP_HL,1,TestHost,192.168.0.3,!!!!
TCPIP_HL,2,****,****,!!!!
```

4.12.26 Error Table

```
REM,Error Table
REM,Facility Code list
REM,0,OS    ,1,SRIO,2,FILE,3,PROG,4,COND
REM,5,ELOG,6,MCTL,7,MEMO,8,GUID,9,TPIF
REM,10,FLPY,11,SRVO,12,INTP,13,PRIO,14,TPAX
REM,15,MOTN,16,VARS,17,ROUT,18,WNDW,19,JOG
REM,20,APPL,21,LANG,22,ASBN,23,SPOT,24,SYST
REM,25,SCIO,26,PALT,27,UAPL,28,PMON,29,TOOL
REM,30,SVGN,31,PWD ,32,VISN,33,DICT,34,KCLI
REM,35,TRAN,36,TKSP,37,COPT,38,APSH,39,ISD
REM,40,DMER,41,MHND,42,CMND,43,RPM ,44,LNTK
REM,45,WEAV,46,TCPP,47,TAST,48,MUPS,49,MIGE
REM,50,LSR ,51,SEAL,52,PAIN,53,ARC ,54,TRAK
REM,55,CALB,56,SP    ,57,MACR,58,SENS,59,COMP
REM,60,THSR,61,QMGR,62,ELSE,63,ELSE,64,DJOG
REM,65,OPTN,66,HRTL,67,HOST,68,MENT,69,SSPC
REM,70,FIG ,71,LDG ,72,DX    ,73,CNTR,74,LODC
REM,75,PFMS,76,DNET,77,GBOX,78,PAL2,79,SHAP
REM,80,ATCP,81,CART,82,CD    ,83,MARL,84,DMDR
REM,85,FRSY,86,PNT1,87,FLEX,88,IB-S,89,RTCP
REM,90,TG    ,91,FORC,92,PROF,93,RPC ,94,VC
REM,95,ISDT,96,GUI ,97,CUST,98,POST,99,ELSE
REM,100,NTOS,101,,102,ATGP,103,WMAP,104,ASST
REM,105,FIND,106,CALM,107,,108,LSTP,109,LECO
REM,110,MLOG,111,APC ,112,ACAL,113,TPCL,114,CPMO
REM,115,PALL,116,TJOG,117,CVIS,118,BBOX,119,TMAT
REM,120,TANG,121,DPMO,122,KALM,123,,124,
REM,125,,126,,127,,128,ELSE,129,XMLF
REM,130,RIPE,131,SPRM,132,PICK,133,,134,
REM,135,SVTL,136,FXTL,137,PNT2,138,ATZN,139,VDIG
REM,140,PTPG,141,SYMR,142,TEST,143,SENC,144,BRCH
REM,145,FPLN,146,****
REM,Index,Facility Code,Error code,Severity,Display,!!!!(Terminator)
ER_TBL,1,11,1,DEFAULT,DEFAULT,!!!!
ER_TBL,2,50,1,STOP,DEFAULT,!!!!
ER_TBL,3,50,2,STOPALL,DEFAULT,!!!!
ER_TBL,4,50,3,ABORT,DEFAULT,!!!!
ER_TBL,5,50,4,ABORTALL,DEFAULT,!!!!
ER_TBL,6,50,5,DEFAULT,ACTIVE,!!!!
ER_TBL,7,50,6,DEFAULT,NODISP,!!!!
ER_TBL,8,50,7,DEFAULT,NOERLOG,!!!!
ER_TBL,9,50,8,DEFAULT,NOERLIN,!!!!
```

4.12.27 Alarms That Do Not Turn on FAULT Signal

```
REM,ER_NO_ALM
REM,Item number,Item name,value,!!!!(Terminator)
ER_NOALM,     1,No alarm enable(1/0), 0,!!!!
ER_NOALM,     2,Number of error code, 5,!!!!
ER_NOALM,     3,Error code 1, 11001,!!!!
ER_NOALM,     4,Error code 2, 11002,!!!!
ER_NOALM,     5,Error code 3, 11003,!!!!
ER_NOALM,     6,Error code 4, 11007,!!!!
ER_NOALM,     7,Error code 5, 11037,!!!!
ER_NOALM,     8,Error code 6, 0,!!!!
ER_NOALM,     9,Error code 7, 0,!!!!
(snip)
ER_NOALM,    40,Error code 38, 0,!!!!
ER_NOALM,    41,Error code 39, 0,!!!!
ER_NOALM,    42,Error code 40, 0,!!!!
```

4.12.28 RSR

```
REM,RSR
REM,Item number,Item name,value,!!!!(Terminator)
RSR,     1,RSR1 program number,ENABLE,0,!!!!
RSR,     2,RSR2 program number,ENABLE,0,!!!!
RSR,     3,RSR3 program number,ENABLE,0,!!!!
RSR,     4,RSR4 program number,ENABLE,0,!!!!
RSR,     5,RSR5 program number,DISABLE,0,!!!!
RSR,     6,RSR6 program number,DISABLE,0,!!!!
RSR,     7,RSR7 program number,DISABLE,0,!!!!
RSR,     8,RSR8 program number,DISABLE,0,!!!!
RSR,    21,Job prefix,RSR,!!!!
```

4.12.29 PNS

```
REM,PNS
REM,Job prefix,!!!!(Terminator)
PNS,PNS,!!!!
```

4.12.30 Program Selection Common

```
REM,Program Selection common
REM,Program select mode,!!!!(Terminator)
REM,Base number,Acknowledge function,Acknowledge pulse width,!!!!(Terminator)
PRG_SEL,PNS,!!!!
, 0,DISABLE, 400,!!!!
```

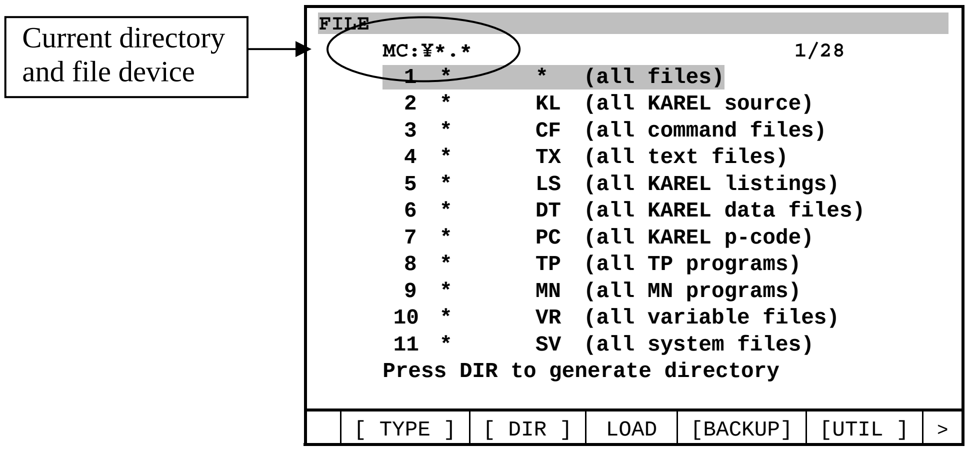
4.12.31 *i*Pendant Backlight Automatic Blanking

```
REM,iPendant automatic blanking
REM,ENABLE/DISABLE,Timer of blanking(minute),!!!!(Terminator)
BLNK,DISABLE,5,!!!!
```

4.13 CAUTION

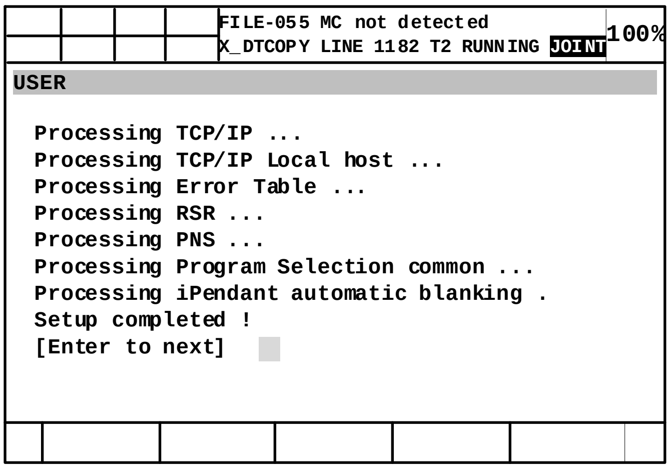
Insert Memory card (and the like) before use

This function uses current directory of current file device. Please use this function after files in current directory of current device are ready to read and write.



```
FILE
   MC:¥*.*                                1/28
    1  *       *   (all files)
    2  *       KL  (all KAREL source)
    3  *       CF  (all command files)
    4  *       TX  (all text files)
    5  *       LS  (all KAREL listings)
    6  *       DT  (all KAREL data files)
    7  *       PC  (all KAREL p-code)
    8  *       TP  (all TP programs)
    9  *       MN  (all MN programs)
   10  *       VR  (all variable files)
   11  *       SV  (all system files)
   Press DIR to generate directory

 [ TYPE ] [ DIR ]  LOAD  [BACKUP] [UTIL ]  >
```

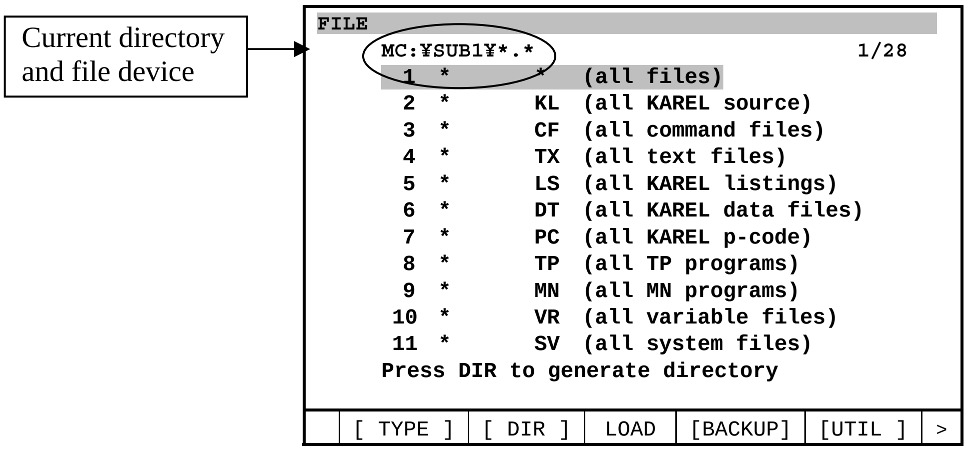
For example, do not output file to MC: without “Flash ATA card” inserted. Alarms, “File-055 MC not detected” for example, are displayed. On USER alarm screen, file output looks complete but actually not. Alarm is displayed.



```
FILE-055 MC not detected
X_DTCOPY LINE 1182 T2 RUNNING JOINT 100%
USER

 Processing TCP/IP ...
 Processing TCP/IP Local host ...
 Processing Error Table ...
 Processing RSR ...
 Processing PNS ...
 Processing Program Selection common ...
 Processing iPendant automatic blanking .
 Setup completed !
 [Enter to next]
```

Use of current directory

This function uses current directory of current file device. Please confirm where current directory is. In addition, do not change current directory while using this function.



```
FILE
    MC:¥SUB1¥*.*                                  1/28
      1  *       *   (all files)
      2  *       KL  (all KAREL source)
      3  *       CF  (all command files)
      4  *       TX  (all text files)
      5  *       LS  (all KAREL listings)
      6  *       DT  (all KAREL data files)
      7  *       PC  (all KAREL p-code)
      8  *       TP  (all TP programs)
      9  *       MN  (all MN programs)
     10  *       VR  (all variable files)
     11  *       SV  (all system files)
    Press DIR to generate directory

   [ TYPE ] [ DIR ]  LOAD  [BACKUP] [UTIL ]  >
```

For example, suppose you remove Flash ATA card when subdirectory of MC: is selected. This causes change of current directory to root of MC:. If you would like to use subdirectory after you insert Flash ATA card again, you have to select subdirectory again in file screen.

Cycle power after set of data

You have to cycle power after setting data from CSV file.

Abort ROBOT SETUP ASSISTANCE TOOL (X_DTCOPY) before switching the screen

It is allowed to switch to other screen from the top screen of System Design Tool unless aborting robot setup assistance tool after outputting the file or complete to set some configuration, however, main program, X_DTCOPY, keeps running. Abort it from robot setup assistance tool screen to input 0 in the menu before switching the screen.

In case of fail to delete the KAREL program

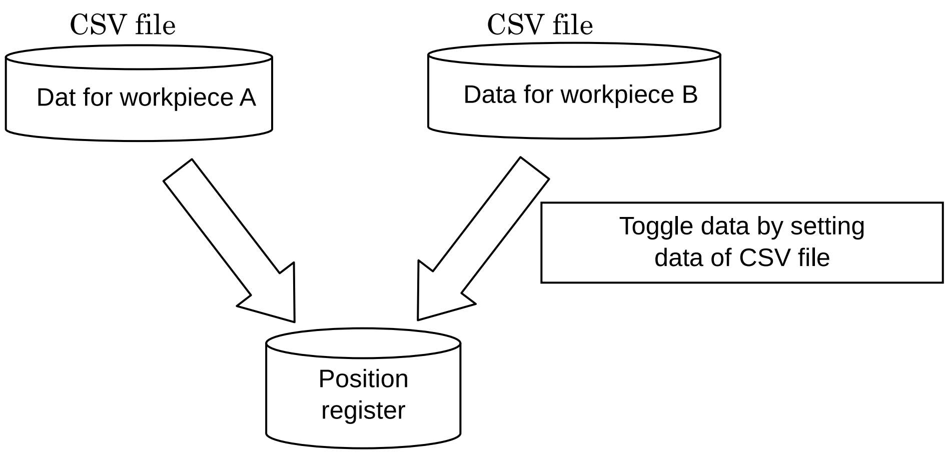
If "SDTL-026 Failed to delete %s" (%s is the KAREL program name) is displayed while deleting the KAREL programs of robot setup assistance tool, some of KAREL program may be still executing. Abort it from robot setup assistance tool screen to input 0 in the menu before switching the screen.

4.14 Data set from CSV file to position register

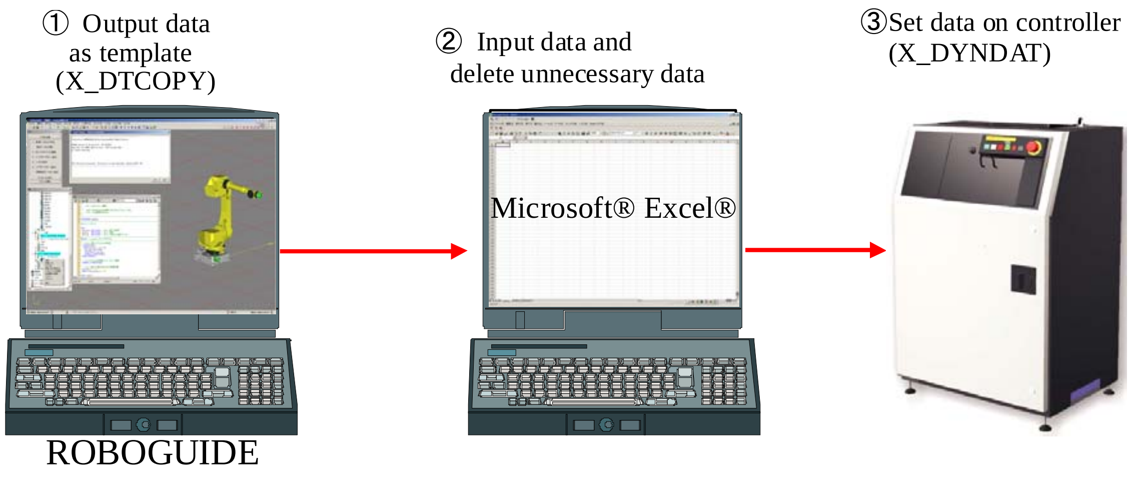
This section describes KAREL program X_DYNDAT, which set data from CSV file to robot controller. In this section, data set from CSV file by X_DYNDAT is called “this function”. This function is available on 7DC1/06 or later. This function is included in Robot setup assistance tool (J737). If you would like to clear position register, please order System design tool (J738) instead of J737.

4.14.1 Overview

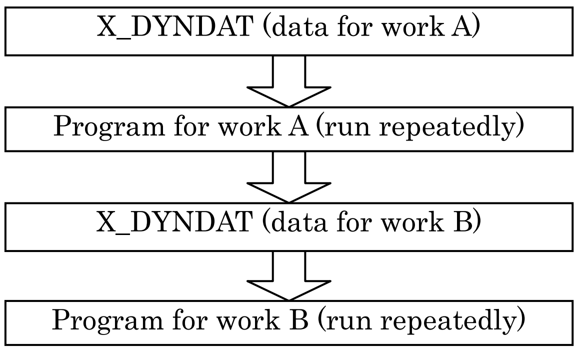
X_DYNDAT is a program to set data in CSV file to robot controller.
Register, position register, string register, tool frame and user frame can be set.
This is function to change data written above, especially position registers, according to program.



Format of CSV file is same as main part of robot setup assistance tool, X_DTCOPY. Please refer to section 4.6 “OVERVIEW OF CSV FILE FORMAT” and section 4.8 “FORMAT OF DATA” for more detail about format. Please refer to subsection 4.14.4 “supported data” for available identifier and how to create CSV file.
Because of common data format, you can output CSV file by X_DTCOPY and use it for X_DYNDAT. Note that you need modification.


① Output data as template (X_DTCOPY)
② Input data and delete unnecessary data
③Set data on controller (X_DYNDAT)
Microsoft® Excel®
ROBOGUIDE

NOTE
It takes time for X_DYNDAT to set data. It is not for frequent data set. Data set takes long time especially when you set many position registers. For example, set of 100 position registers all at once may take 10 seconds or more.


X_DYNDAT (data for work A)
Program for work A (run repeatedly)
X_DYNDAT (data for work B)
Program for work B (run repeatedly)

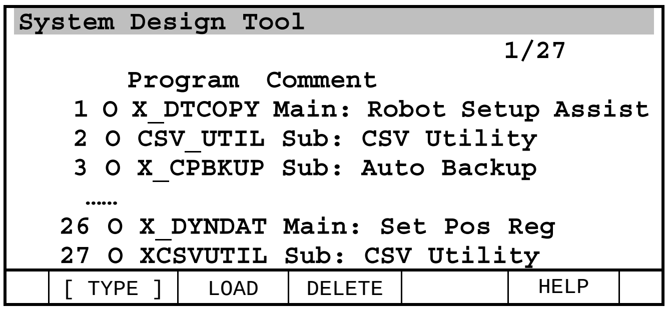
Both main program of Robot setup assistance tool (X_DTCOPY) and X_DYNDAT can set data from CSV file. However, purpose and functionality of them are different. Please refer to subsection 4.14.15 for more detail on difference.

Microsoft® Excel® is registered trademark of Microsoft Corporation.

4.14.2 Usage

1 Load all KAREL programs of Robot setup assistance tool.
Please refer to section 4.4 "HOW TO LOAD KAREL PROGRAMS"
2 Place CSV file in file device like MC:.
3 Run X_DYNDAT directly or call the KAREL program from TP program.
File path is decided by argument or STRING register. For more detail, please refer to next section.

This function is available on 7DC1/06 or later. If your controller supports the function, robot setup assistance tool screen of system design tool displays X_DYNDAT.



```
System Design Tool
                                 1/27
      Program  Comment
   1 O X_DTCOPY Main: Robot Setup Assist
   2 O CSV_UTIL Sub: CSV Utility
   3 O X_CPBKUP Sub: Auto Backup
    ……
  26 O X_DYNDAT Main: Set Pos Reg
  27 O XCSVUTIL Sub: CSV Utility
```

4.14.3 Program for data set from CSV:X_DYNDAT

Program name:X_DYNDAT

Syntax1:No parameter

X_DYNDAT

Syntax2:with 1 parameter

X_DYNDAT (file_name)
[IN] file_name : STRING

Detail

- file_name specifies path of CSV file. X_DYNDAT sets data from specified CSV file to robot controller.
- If there is no argument, X_DYNDAT uses file specified by string register.
  In both case, file name should be specified by full path.
- X_DYNDAT of SYNTAX2 uses SR[1] by default.
- Index of string register for SYNTAX2 is decided by KAREL variable, [X_DYNDAT]IDEFSRIDX.
  It is variable IDEFSRIDX of KAREL program X_DYNDAT.

Time for data set and how to invoke X_DYNDAT

You can run X_DYNDAT directly, as a task. Or you can call X_DYNDAT from another TP program. Generally speaking, execution by the former way (direct RUN as a task) is faster than the latter (call from TP program).
When you run X_DYNDAT as a task, you cannot hand arguments to the program. File path must be specified by string register.

NOTE
This function can set string register. Be careful not to change file path data in string register by mistake.

NOTE

It takes time for X_DYNDAT to set data. It is not for frequent data set. Data set takes long time especially when you set many position registers. For example, set of 100 position registers all at once may take 10 seconds or more.

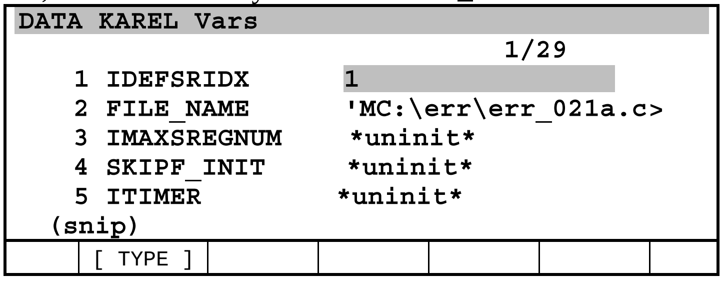
Initialization of position register

This function can set position register. No to use unexpected data, it would be good to initialize position registers before setting them. For initializing position register, KAREL program CRPOSREG, which is included in system design tool (J738), can be used.
X_DYNDAT and X_DTCOPY cannot make position register uninitialized.

Example1: X_DYNDAT with no parameter

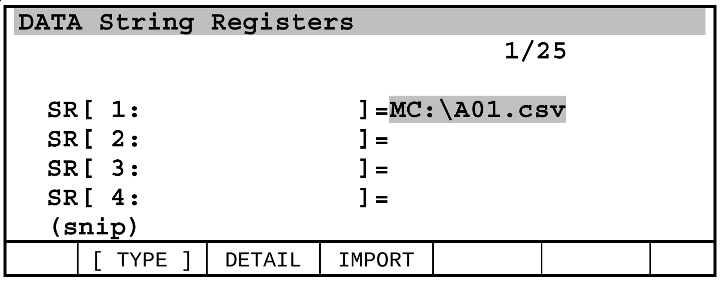
File path is specified by string register. In this example, SR[1] is used.

1 Set KAREL variable IDEFSRIDX of KAREL program X_DYNDAT to 1
If it is uninitialized, it is automatically set to 1 when X_DYNDAT is run or called with no argument.



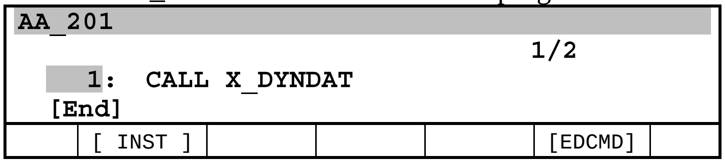
```
DATA KAREL Vars
                                  1/29
   1 IDEFSRIDX       1
   2 FILE_NAME       'MC:\err\err_021a.c>
   3 IMAXSREGNUM      *uninit*
   4 SKIPF_INIT       *uninit*
   5 ITIMER          *uninit*
  (snip)
```

2 Set file path to SR[1]
In this example, MC:¥A01.CSV is used.



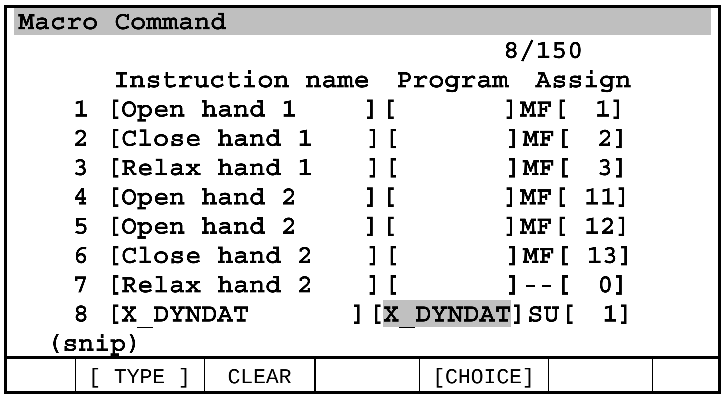
```
DATA String Registers
                                  1/25

  SR[ 1:                 ]=MC:\A01.csv
  SR[ 2:                 ]=
  SR[ 3:                 ]=
  SR[ 4:                 ]=
  (snip)
```

3 Use X_DYNDAT with no argument
For example, you can call X_DYNDAT from another TP program.



```
AA_201
                                   1/2
     1:  CALL X_DYNDAT
  [End]
```

You can run X_DYNDAT directly, as a task. In this case, no argument is handed to X_DYNDAT. If you run X_DYNDAT as MACRO after step 1 and 2 above, file specified by SR[1] is used. In this case, MC:¥A01.CSV is used.



```
Macro Command
                                   8/150
      Instruction name  Program  Assign
   1  [Open hand 1     ][        ]MF[  1]
   2  [Close hand 1    ][        ]MF[  2]
   3  [Relax hand 1    ][        ]MF[  3]
   4  [Open hand 2     ][        ]MF[ 11]
   5  [Open hand 2     ][        ]MF[ 12]
   6  [Close hand 2    ][        ]MF[ 13]
   7  [Relax hand 2    ][        ]--[  0]
   8  [X_DYNDAT       ][X_DYNDAT]SU[  1]
 (snip)
     [ TYPE ]  CLEAR        [CHOICE]
```

As you can see in this example, RUN of X_DYNDAT reads file specified by string register. Please note that this function can set string register. If you specify string register for file path in CSV file, path in the string register is changed.

NOTE
Please be careful not to change file path by setting string register value by mistake.

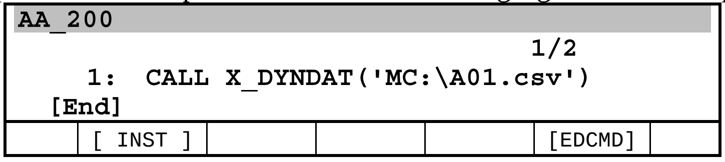
If there is no need to set string register, remove lines for string register from CSV file if exist. Lines for string register start with SREG. X_DTCOPY outputs lines for string registers with identifier SREG. Following is sample output of string register by X_DTCOPY. This is an example of display by Microsoft® Excel®

	A	B	C	D	E
1426	REM	String register			
1427	REM	Index	String	Comment	!!!!(Terminator)
1428	SREG	1	PNS0001	Main program	!!!!

Lines that start with REM is comment.

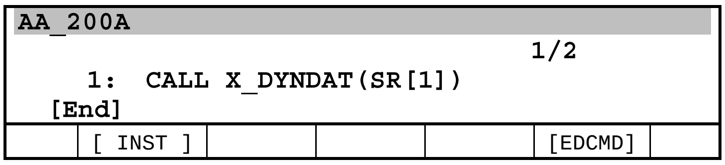
Example2: X_DYNDAT with parameter

The first argument specifies the file path of CSV file. Following figure is an example.



```
AA_200
                                    1/2
    1:  CALL X_DYNDAT('MC:\A01.csv')
 [End]
     [ INST ]                    [EDCMD]
```

You can use string register as the first argument. In this case, value of string register must be less than equal to 34.



```
AA_200A
                                    1/2
    1:  CALL X_DYNDAT(SR[1])
 [End]
     [ INST ]                    [EDCMD]
```

4.14.4 Supported data

This function can set following data from CSV file.
- Numeric register
- Position register
- String register
- Tool frame

- User frame

Tool frame and User frame don't include currently selected frame number.

Data format of CSV file is same as main part of robot setup assistance tool, X_DTCOPY.
Following table lists identifier available. In addition to data listed above, identifier of comment and file end are included.

Identifier	Description
REM	Comment line
END	End of file
REG	Numeric register
PREG	Position register
SREG	String register
TFRM	Tool frame
UFRM	User frame

When you create CSV file, please use CSV file output by X_DTCOPY as template.

NOTE
File output by X_DTCOPY includes data that is not supported by X_DYNDAT. Please make sure to remove lines for data that is not supported.

X_DTCOPY cannot output CSV file with identifier supported by this function.

4.14.5 Difference between X_DTCOPY and X_DYNDAT

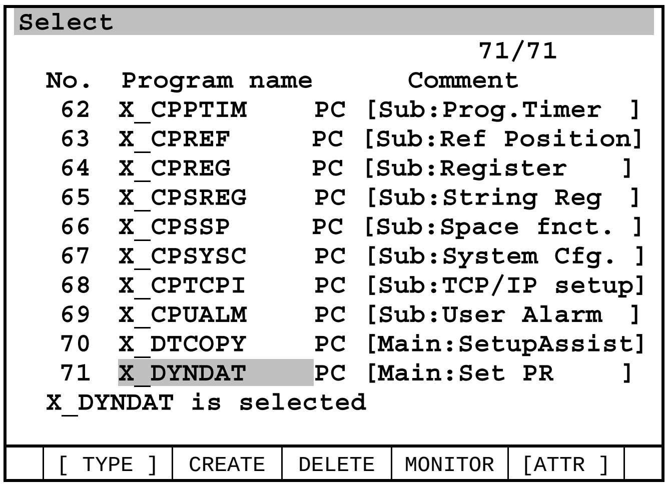
X_DTCOPY is a program to support initial setup of robot system. After system setup completes, the program is not used basically.
On the contrary, X_DYNDAT is used after initial setup. It is a program to change data like register and position register before production program actually uses the data.

X_DYNDAT is different from X_DTCOPY in following points.
- X_DYNDAT doesn't have menu.
  If program is executed without error, execution completes without manual input.
- When error is detected, error is displayed.
- Data output to CSV file is not supported.
- Only part of identifiers is supported.
  Identifier that is supported by X_DTCOPY but not supported by X_DYNDAT causes error.

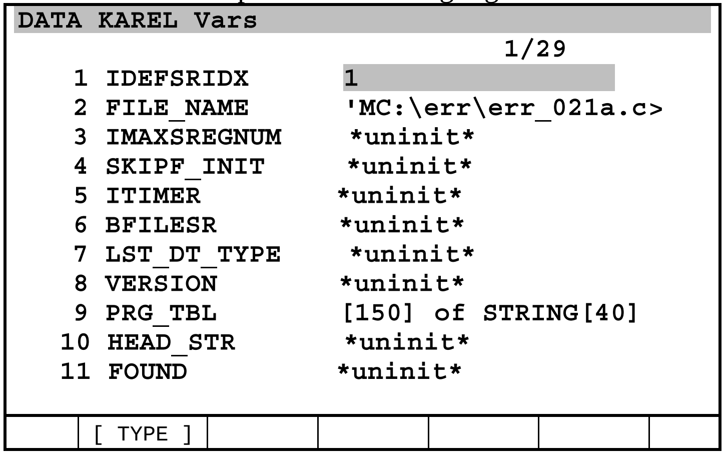
4.14.6 File path for X_DYNDAT without argument

As described in subsection 4.14.3, X_DYNDAT without argument uses value of string register as file path of CSV file. Index of the string register is specified by KAREL variable IDEFSRIDX of program X_DYNDAT.
The variable can be set by KAREL variable screen. Please use following procedure.

1 Select X_DYNDAT by SELECT screen.


```
Select
                                   71/71
  No.  Program name      Comment
   62  X_CPPTIM    PC  [Sub:Prog.Timer  ]
   63  X_CPREF     PC  [Sub:Ref Position]
   64  X_CPREG     PC  [Sub:Register    ]
   65  X_CPSREG    PC  [Sub:String Reg  ]
   66  X_CPSSP     PC  [Sub:Space fnct. ]
   67  X_CPSYSC    PC  [Sub:System Cfg. ]
   68  X_CPTCPI    PC  [Sub:TCP/IP setup]
   69  X_CPUALM    PC  [Sub:User Alarm  ]
   70  X_DTCOPY    PC  [Main:SetupAssist]
   71  X_DYNDAT    PC  [Main:Set PR     ]
  X_DYNDAT is selected

  [ TYPE ]  CREATE  DELETE  MONITOR  [ATTR ]
```

2 Press DATA key.
3 Press F1, [TYPE] and select “KAREL Vars”.
KAREL variable screen is displayed. If you selected X_DYNDAT on the same pane, the screen lists global variables of X_DYNDAT.
If screen is divided into 2 or 3 panes, please display KAREL variable screen by the same pane you selected X_DYNDAT. If you selected X_DYNDAT by the 2nd pane, display KAREL variable screen on the 2nd pane, too.
4 Move cursor to IDEFSRIDX and input index of string register.


```
DATA KAREL Vars
                                  1/29
    1 IDEFSRIDX      1
    2 FILE_NAME      'MC:\err\err_021a.c>
    3 IMAXSREGNUM     *uninit*
    4 SKIPF_INIT      *uninit*
    5 ITIMER         *uninit*
    6 BFILESR        *uninit*
    7 LST_DT_TYPE     *uninit*
    8 VERSION        *uninit*
    9 PRG_TBL        [150] of STRING[40]
   10 HEAD_STR        *uninit*
   11 FOUND          *uninit*

  [ TYPE ]
```

NOTE
Don’t change any other variable.

Just after load of X_DYNDAT, IDEFSRIDX is uninitialized. If IDEFSRIDX is uninitialized when X_DYNDAT is RUN or called without argument, the variable is initialized to 1. It means that default value is virtually 1.

4.14.7 Caution

Resume after error

If you call X_DYNDAT from TP program, move cursor onto a line before CALL instruction before resume. If X_DYNDAT was RUN as a task, abort it and RUN X_DYNDAT again.

These are true when power failure occurred during execution of X_DYNDAT.

Display of format error

If CSV file has format error, error of SDTL is displayed. For example, when X_DYNDAT detects identifier DI, “SDTL-048 Wrong data type (Line %s)” is displayed. Identifier DI is not supported by X_DYNDAT although it is supported by X_DTCOPY.
Contents of %s may not be displayed properly if it is not displayed by the language that was selected when the error occurred. Please display the error by the same language. This is true to the other format error, from SDTL-048 to SDTL-052.

Usage of position register

Position registers are interpreted based on currently selected tool and user frame.

⚠ **WARNING**

Before using position register set by X_DYNDAT, please confirm proper tool frame and user frame are selected and they are setup properly. If selection or data of frames are wrong, it may cause unexpected motion.

Timing of execution of X_DYNDAT

If you set position register by X_DYNDAT between motion statements, backward execution may not work as you expected. Suppose following program is halted at line and operator tries backward execution. If X_DYNDAT changes PR[1-3], backward path of execution of line 1, 2 and 3 is not same as previous FWD execution. Position register of destination points are already changed.

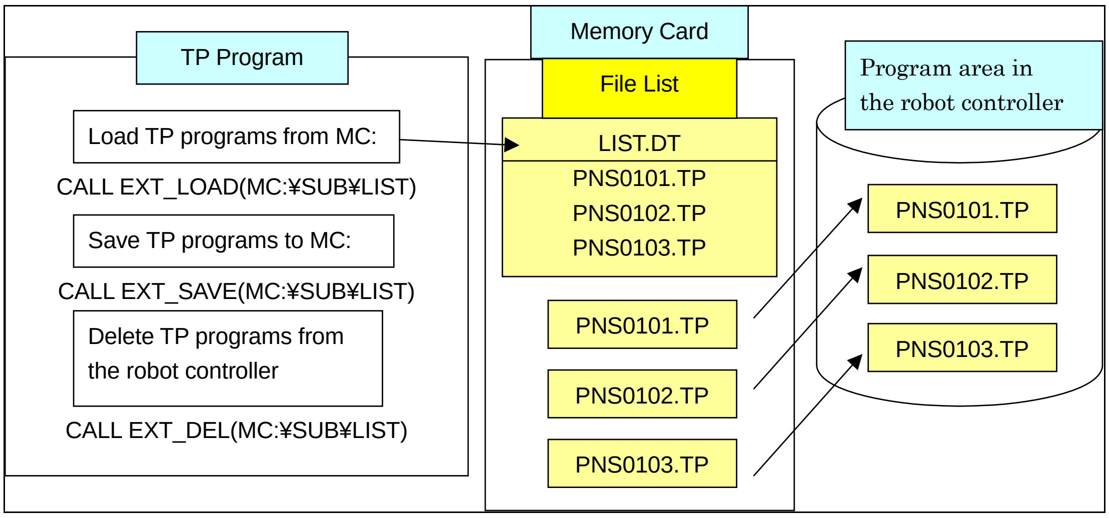
```
1:J   PR[1] 100% FINE
2:L   PR[2] 100mm/sec FINE
3:L   PR[3] 100mm/sec FINE
4:    CALL X_DYNDAT
5:J   PR[1] 100% FINE
6:L   PR[2] 100mm/sec FINE
7:L   PR[3] 100mm/sec FINE
```

Please run X_DYNDAT as a task before running program that have motion statements. Execution of X_DYNDAT before execution of production program is recommended. Please refer to figure in subsection 4.14.1, Overview.
Otherwise, Call X_DYNDAT before execution of a series of motion statements. Following example calls X_DYNDAT at line 1. However, run of X_DYNDAT as a task is faster than CALL. In addition, call in production program itself causes data set every time the program is run. It would cause longer cycle time.

```
1:    CALL X_DYNDAT
2:J   PR[1] 100% FINE
3:L   PR[2] 100mm/sec FINE
4:L   PR[3] 100mm/sec FINE
5:J   PR[4] 100% FINE
6:L   PR[5] 100mm/sec FINE
7:L   PR[6] 100mm/sec FINE
```

5 TP PROGRAM MEMORY ASSIST

Program memory gets low in some cases that there are a lot of TP programs or very big size of TP programs to be machined.
It runs out of program memory in some cases. TP Program Memory Assist is very useful under such circumstances. It is allowed to save TP programs to external device (memory card (MC:), USB memory (UD1:)) or FTP (C1:~C8:) etc, load them from it and delete them from the robot controller according to specified file list (text file).


TP Program
Load TP programs from MC:
CALL EXT_LOAD(MC:¥SUB¥LIST)
Save TP programs to MC:
CALL EXT_SAVE(MC:¥SUB¥LIST)
Delete TP programs from the robot controller
CALL EXT_DEL(MC:¥SUB¥LIST)
Memory Card
File List
LIST.DT
PNS0101.TP
PNS0102.TP
PNS0103.TP
PNS0101.TP
PNS0102.TP
PNS0103.TP
Program area in the robot controller
PNS0101.TP
PNS0102.TP
PNS0103.TP

5.1 FILE LIST OF TP PROGRAM MEMORY ASSIST

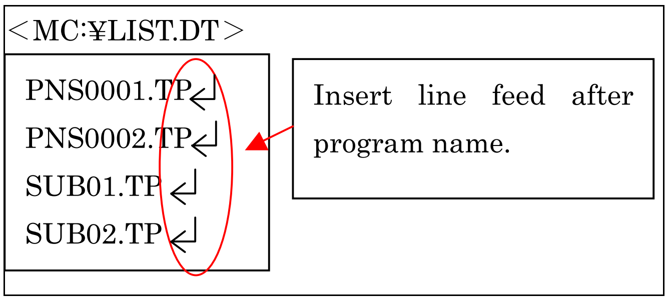
Describe TP program name into the file list. KAREL programs of TP Program Memory Assist save, delete or load according to its list. Use PC to create the file list.

5.1.1 Create the File List

File list name is LIST.DT at this explanation as an example. File list name is allowed to change. Extension must be DT. Describe TP program name that the target to save, delete or load into it.

Pay attention to following points and create it.

1. Only one program per one line must be written. Do not insert space before or after the program. Program name must be written at the head of line. Line feed must insert just after the program name.
2. Line feed must be inserted to the end of the file list.
3. Extension of file list must be DT.



NOTE

Maximum of program name is 36 characters. Please refer to the chapter "PROGRAM STRUCTURE" of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know detail of program name.

5.1.2 Usage of the File List

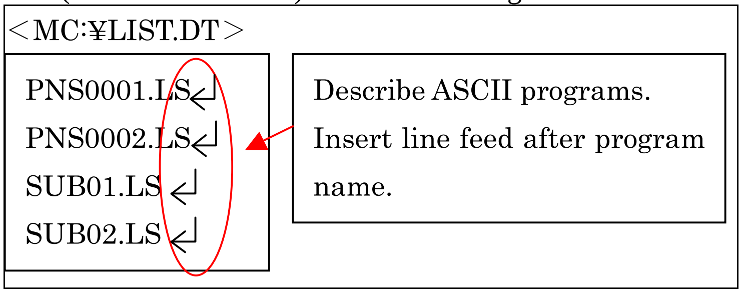
Prepare the created file list into root or sub directory of the external device. Prepare TP programs described in the list into the same directory.

LIST.DT, PNS0001.TP, PNS0002.TP, SUB01.TP and SUB02.TP are prepared to the same directory, MC:¥, on above sample.

5.1.3 Using ASCII PROGRAM LOADER FUNCTION (software option R507)

TP Program Memory Assist function allows you to load ASCII programs if ASCII PROGRAM LOADER FUNCTION software option is installed. Refer to the Section “ASCII PROGRAM LOADER FUNCTION” of the Chapter “FILE INPUT/OUTPUT” in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know ASCII PROGRAM LOADER FUNCTION.

Describe ASCII programs (the extension is LS) into file list. Usage is the same of above.



＜MC:¥LIST.DT＞

```
PNS0001.LS↲
PNS0002.LS↲
SUB01.LS↲
SUB02.LS↲
```

Describe ASCII programs.
Insert line feed after program name.

Refer to the subsection “ASCII save” in the chapter “FILE INPUT/OUTPUT” in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know how to output ASCII files.

NOTE

Pay attention to following points if you describe ASCII programs into file list.

- “Delete program” is not allowed to executed. “FILE-024 Illegal file type” is displayed.
  It is allowed to delete the programs to use the file list of TP program that has TP extension.
- Only ASCII programs (LS files) are saved if “Save program” is executed. TP programs are not saved.

5.2 LOAD TP PROGRAM

Load TP programs according to the file list of 1st parameter.

KAREL PROGRAM NAME: EXT_LOAD

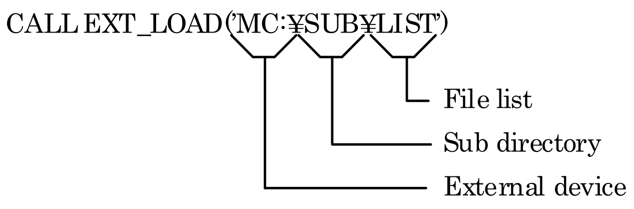
SYNTAX

EXT_LOAD(list_name)
[in] list_name : STRING

DETAIL

list_name: File path (without extension) to load programs from external device to the robot controller. String included device name, sub directory and file list name.

SAMPLE



> **NOTE**
> - If the same program name exists in the robot controller, it is overwritten.
> - "MEMO - 126 No more available TPP memory" is displayed if the robot controller does not have enough memory.
> - "SDTL-017 Failed to load %s" is displayed when you try to load the executing or pausing program.

5.3 SAVE TP PROGRAM

Save TP programs according to the file list of 1st parameter.

KAREL PROGRAM NAME: EXT_SAVE

SYNTAX

EXT_SAVE(list_name)
[in] list_name : STRING

DETAIL

list_name: File path (without extension) to save programs from the robot controller to external device. String included device name, sub directory and file list name.

SAMPLE

CALL EXT_SAVE('MC:¥SUB¥LIST')

> **NOTE**
> If the same program name exists in the external device, it is overwritten.

5.4 DELETE TP PROGRAM

Delete TP programs according to the file list of 1st parameter.

KAREL PROGRAM NAME: EXT_DEL

SYNTAX

EXT_DEL(list_name)
[in] list_name : STRING

DETAIL

list_name: File path (without extension) to delete programs the robot controller from to
external device. String included device name, sub directory and file list name.

SAMPLE

CALL EXT_DEL(‘MC:¥SUB¥LIST’)

Refer to section 9.4 "SELECT TP PROGRAMS BY WORKS" to see the application sample.

5.5 NOTE

- Alarm or warning is displayed in following cases.
    - No TP program described in the file list exists.
    - Running or pausing TP program is intended to delete or load.
    - Invalid parameter is taught.
- Before using this function files in external device must be saved for deleting by miss operation such as doing format from FILE screen.
- KAREL programs are not supported power failure recovery. If power is turned off when the KAREL program of this function is running, please set the cursor onto a line before CALL instruction and execute again.
- Please do not remove the external device while this function is running. Otherwise, TP programs are not correctly loaded, saved or deleted.
- The time of executing this function is different by the number of programs or external devices.
- In the case of a lot of programs are treated, you had better to use external device formatted by FAT32. In some cases you do not make effective use of this function because there is an upper limit (256) to save programs into root directory of external device formatted by FAT.
- String length as specified parameter that includes device, directory, and file list name must be lower 24. If length is out of range, “SDTL - 007 Invalid file list” is displayed.
- :¥ must be input in the parameter in the case EXT_LOAD, EXT_DEL or EXT_SAVE is called.
  Sample : EXT_LOAD(‘MC:¥SUB¥LIST’)
  If they are not detected, “SDTL – 007 Invalid file list” is displayed.

6 SIM CHECK/UNSIM I/O

It is allowed to check I/O simulated status. It is allowed to prevent forgetting to unsimulate after simulating I/O to modify TP program.

It is allowed to unsimulate I/O.

NOTE
This function is enabled in only the case that “Digital/Analog I/O:” in test cycle setup screen is ENABLE. If it is DISABLE it cannot unsimulate I/O by this function.

Selecting following show test cycle setup screen.
[MENU] -> 2 TEST CYCLE -> F1[TYPE] -> Setup.

⚠ **WARNING**
The controller uses signals to control the peripheral equipment. The simulated I/O may adversely affect the security of the system. Check the use of signals in the system before attempting simulated I/O. In some cases it is very dangerous that to unsimulate I/O causes running the robot or peripheral.

6.1 CHECK DI/O SIMULATED STATUS

This function checks DI/O simulated status. In case that integer 0 is specified by the 2nd parameter, it checks all of DI/O simulated status from 1 to the value set in “Max. number setting screen”, and if one or more I/O is simulated status 1 is output to register.
Its screen is selected in controlled start menu. Refer to the chapter “SPECIAL OPERATION” in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know the detail of it.

KAREL PROGRAM NAME: CKDIOSIM

SYNTAX

CKDIOSIM(in_or_out, io_idx, register_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is DI, 2 is DO.
io_idx: I/O number to check.
register_idx: Register number to output result. If unsimulated, 0 is output. If simulated, 1 is output.

SAMPLE

CALL CKDIOSIM(1, 1, 1) -> R[1] = 1 (If DI[1] is simulated.)

Refer to the section 9.2 "ALARM IS DISPLAYED IF I/O IS SIMULATED" about the application sample.

6.2 CHECK AI/O SIMULATED STATUS

This function checks AI/O simulated status. In case that integer 0 is specified by the 2nd parameter, it checks all of AI/O simulated status from 1 to 64, and if one or more I/O is simulated status 1 is output to register.

KAREL PROGRAM NAME: CKAIOSIM

SYNTAX

CKAIOSIM(in_or_out, io_idx, register_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is AI, 2 is AO.
io_idx: I/O number to check.
register_idx: Register number to output result. If unsimulated, 0 is output. If simulated, 1 is output.

SAMPLE

CALL CKAIOSIM(1, 1, 1) -> R[1] = 1 (If AI[1] is simulated.)

6.3 CHECK GI/O SIMULATED STATUS

This function checks GI/O simulated status. In case that integer 0 is specified by the 2nd parameter, it checks all of GI/O simulated status from 1 to 64, and if one or more I/O is simulated status 1 is output to register.

KAREL PROGRAM NAME: CKGIOSIM

SYNTAX

CKGIOSIM(in_or_out, io_idx, register_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is GI, 2 is GO.
io_idx: I/O number to check.
register_idx: Register number to output result. If unsimulated, 0 is output. If simulated, 1 is output.

SAMPLE

CALL CKGIOSIM(1, 1, 1) -> R[1] = 1 (If GI[1] is simulated.)

6.4 CHECK RI/O SIMULATED STATUS

This function checks RI/O simulated status. In case that integer 0 is specified by the 2nd parameter, it checks all of RI/O simulated status from 1 to the value of group number * 8, and if one or more I/O is simulated status 1 is output to register. For instance, if there are 3 groups a maximum of 24 points are checked.

KAREL PROGRAM NAME: CKRIOSIM

SYNTAX

CKRIOSIM(in_or_out, io_idx, register_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is RI, 2 is RO.
io_idx: I/O number to check.
register_idx: Register number to output result. If unsimulated, 0 is output. If simulated, 1 is output.

SAMPLE

CALL CKRIOSIM(1, 1, 1) -> R[1] = 1 (If RI[1] is simulated.)

6.5 CHECK WI/O SIMULATED STATUS

This function checks WI/O simulated status. In case that integer 0 is specified by the 2nd parameter, it checks all of WI/O simulated status from 1 to 32, and if one or more I/O is simulated status 1 is output to register.

KAREL PROGRAM NAME: CKWIOSIM

SYNTAX

CKWIOSIM(in_or_out, io_idx, register_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is WI, 2 is WO.
io_idx: I/O number to check.
register_idx: Register number to output result. If unsimulated, 0 is output. If simulated, 1 is output.

SAMPLE

CALL CKWIOSIM(1, 1, 1) -> R[1] = 1 (If WI[1] is simulated.)

6.6 UNSIMULATE DI/O

This function unsimulates DI/O. In case that integer 0 is specified by the 2nd parameter, it unsimulates all of DI/O from 1 to the value set in “Max. number setting screen”.
Its screen is selected in controlled start menu. Refer to the chapter “SPECIAL OPERATION” in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know the detail of it.

KAREL PROGRAM NAME: CRDIOSIM

SYNTAX

CRDIOSIM(in_or_out, io_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is DI, 2 is DO.
io_idx: I/O number to check.

SAMPLE

CALL CRDIOSIM(1, 1) -> Unsimulate DI[1]

6.7 UNSIMULATE AI/O

This function unsimulates AI/O. In case that integer 0 is specified by the 2nd parameter, it unsimulates all of AI/O from 1 to 64.

KAREL PROGRAM NAME: CRAIOSIM

SYNTAX

CRAIOSIM(in_or_out, io_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is AI, 2 is AO.
io_idx: I/O number to check.

SAMPLE

CALL CRAIOSIM(1, 1) -> Unsimulate AI[1]

6.8 UNSIMULATE GI/O

This function unsimulates GI/O. In case that integer 0 is specified by the 2nd parameter, it unsimulates all of GI/O from 1 to 64.

KAREL PROGRAM NAME: CRGIOSIM

SYNTAX

CRGIOSIM(in_or_out, io_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is GI, 2 is GO.
io_idx: I/O number to check.

SAMPLE

CALL CRGIOSIM(1, 1) -> Unsimulate GI[1]

6.9 UNSIMULATE RI/O

This function unsimulates RI/O. In case that integer 0 is specified by the 2nd parameter, it unsimulates all of RI/O from 1 to the value of group number * 8.

KAREL PROGRAM NAME: CRRIOSIM

SYNTAX

CRRIOSIM(in_or_out, io_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is RI, 2 is RO.
io_idx: I/O number to check.

SAMPLE

CALL CRRIOSIM(1, 1) -> Unsimulate RI[1]

6.10 UNSIMULATE WI/O

This function unsimulates WI/O. In case that integer 0 is specified by the 2nd parameter, it unsimulates all of WI/O from 1 to 32.

KAREL PROGRAM NAME: CRWIOSIM

SYNTAX

CRWIOSIM(in_or_out, io_idx)
[in] in_or_out : INTEGER
[in] io_idx : INTEGER

DETAIL

in_or_out: Specifying input or output. 1 is WI, 2 is WO.
io_idx: I/O number to check.

SAMPLE

CALL CRWIOSIM(1, 1) -> Unsimulate WI[1]

6.11 UNSIMULATE All I/O

This function unsimulate all I/O.

KAREL PROGRAM NAME: CRIOSIM

SYNTAX

CRIOSIM

DETAIL

There is no parameter.

SAMPLE

CALL CRIOSIM

7 STATUS CHECK/SET FUNCTION

This function allows you to check the robot status from TP programs. It prevents you from getting into a trouble by forgetting to restore some configuration after modifying the robot system.

7.1 GET WRITE PROTECTION ATTRIBUTE

This function allows you to get write protection attribute.
It gets its attribute of TP program specified by the 1st parameter. If write protection is ON 0 is output to register specified by the 2nd parameter. If it is OFF, 1 is output to the register.

KAREL PROGRAM NAME: GTWRTPRT

SYNTAX

GTWRTPRT(prog_name, register_idx)
[in] prog_name :STRING
[out] register_idx : INTEGER

DETAIL

prog_name: TP program name (without extension) to be got write protection attribute.
register_idx: Register number to be output result.

SAMPLE

CALL GTWRTPRT(‘PNS0001’, 1) -> R[1] = 0 (If PNS0001.TP is write protected.)

7.2 CHECK DRY RUN

This function checks the dry run status. If dry run is ENABLE, 1 is output to the register specified by the 1st parameter. If DISABLE, 0 is output.

KAREL PROGRAM NAME: CKDRYRUN

SYNTAX

CKDRYRUN(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output result whether dry run mode or not.

SAMPLE

CALL CKDRYRUN(1) -> R[1] = 1 (If dry run is ENABLE.)

7.3 CHECK GROUP MOTION (MACHINE LOCK)

This function checks group motion. If group motion is ENABLE (machine lock is DISABLE), 0 is output to the register specified by the 1st parameter. If DISABLE (machine lock is ENABLE), 1 is output.

KAREL PROGRAM NAME: CKGRPMTN

SYNTAX

CKGRPMTN(group_idx, register_idx)
[in] group_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
register_idx: Register number to be output the group motion (machine lock) status.

SAMPLE

CALL CKGRPMTN(1, 1) -> R[1] = 1 (If group motion of G1 is DISABLE.)

7.4 CHECK SINGLE STEP

This function checks the single step status. If single step is ENABLE, 1 is output to the register specified by the 1st parameter. If DISABLE, 0 is output.

KAREL PROGRAM NAME: CKSGLSTP

SYNTAX

CKSGLSTP(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output the single step status.

SAMPLE

CALL CKSGLSTP(1) -> R[1] = 1 (If single step is ENABLE.)

7.5 CHECK IGNORE OFFSET COMMAND

This function checks the “Ignore Offset command” in GENERAL screen (Select [MENU] -> “6.SETUP” -> GENERAL). If it is ENABLE, 1 is output to the register specified by the 1st parameter. If DISABLE, 0 is output.

KAREL PROGRAM NAME: CKOFFSET

SYNTAX

CKOFFSET(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output “Ignore Offset command”.

SAMPLE

CALL CKOFFSET(1) -> R[1] = 1 (If Ignore Offset command is ENABLE.)

7.6 CHECK IGNORE TOOL_OFFSET

This function checks the “Ignore Tool_offset” in GENERAL screen. If it is ENABLE, 1 is output to the register specified by the 1st parameter. If DISABLE, 0 is output.

KAREL PROGRAM NAME: CKTLOFST

SYNTAX

CKTLOFST(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output “Ignore Tool_offset”.

SAMPLE

CALL CKTLOFST(1) -> R[1] = 1 (If “Ignore Tool_offset” is ENABLE.)

7.7 CHECK ENABLE VOFFSET

This function checks “Enable VOFFSET” in GENERAL screen. If it is ENABLE, 1 is output to the register specified by the 1st parameter. If DISABLE, 0 is output.

KAREL PROGRAM NAME: CKVOFFST

SYNTAX

CKVOFFST(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output “Enable VOFFSET”.

SAMPLE

CALL CKTLOFST(1) -> R[1] = 1 (If “Enable VOFFSET” is ENABLE.)

7.8 CHECK OVERRIDE

This function checks and outputs override to the register specified by the 1st parameter.
It is useful in the case that saving override before changing by override instruction and restoring it.

KAREL PROGRAM NAME: CKOVRIDE

SYNTAX

CKOVRIDE(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output override.

SAMPLE

CALL CKTLOFST(1) -> R[1] = 100 (If override is 100%.)

NOTE

If override is FINE or VFINE, 1 is output to the register.

7.9 CHECK ENABLE UI SIGNALS

This function checks “Enable UI signals” in CONFIG screen (Select [MENU] -> “0.-- NEXT --” -> “6.SYSTEM” -> Config). If it is ENABLE, 0 is output to the register specified by the 1st parameter. If DISABLE, 1 is output.

KAREL PROGRAM NAME: CKUOPDIS

SYNTAX

CKUOPDIS(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output “Enable UI signals”.

SAMPLE

CALL CKUOPDIS(1) -> R[1] = 0 (If “Enable UI signals” is ENABLE.)

7.10 GET MODE SWITCH

This function gets mode switch status. If it is AUTO, 0 is output to the register specified by the 1st parameter. If T1, 1 is output. If T2, 2 is output.

KAREL PROGRAM NAME: GTMODESW

SYNTAX

GTMODESW(register_idx)
[out] register_idx : INTEGER

DETAIL

register_idx: Register number to be output mode switch.

SAMPLE

CALL GTMODESW(1) -> R[1] = 1 (If mode switch is T1.)

7.11 SWITCH SINGLE STEP

> ⚠**WARNING**
> Verify no one works near the robot or inside the safety fence in case that switching step mode by remote control. The robot may run unexpected motion in such a case the operator remains oblivious to switch step mode by remotely.

> **NOTE**
> Switching single step is the same of pushing STEP key on teach pendant. Use “CHECK SINGLE STEP” in “7.4 CHECK SINGLESTEP” to confirm to switch single step surely.

7.11.1 Single Step ON

Set single step ON, and the warning, “SDTL-001 [STSTEPON] Single Step ON”, is displayed.

> **NOTE**
> CMDENBL UO[1] is turned off when single step mode is ON.

KAREL PROGRAM NAME: STSTEPON

SYNTAX

STSTEPON

DETAIL

There is no parameter.

SAMPLE

CALL STSTEPON

7.11.2 Single Step OFF

Set single step OFF, and the warning, “SDTL-002 [STSTEPOF] Single Step OFF”, is displayed.

KAREL PROGRAM NAME: STSTEPOF

SYNTAX

STSTEPOF

DETAIL

There is no parameter.

SAMPLE

CALL STSTEPOF

7.12 SET CURRENT POSITION

7.12.1 Set Current Position to Register

Set the current position of the specified group and axis as parameter to the register.

KAREL PROGRAM NAME: STPOSREG

SYNTAX

STPOSREG(group_idx, frame_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] frame_idx : INTEGER

[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
frame_idx: Coordinate system index. 1 is joint coordinate, 2 is Cartesian coordinate.
axis_idx: Axis number to set current position.
register_idx: Register number to output result.

SAMPLE

CALL STPOSREG (2,1,1,1) -> Output the joint value of J1 of G2 to R[1].

7.12.2 Set Current Position to Position Register

Set the current position of the specified group as parameter to the position register.

KAREL PROGRAM NAME: STPOSPRG

SYNTAX

STPOSPRG(group_idx, frame_idx, posreg_idx)
[in] group_idx : INTEGER
[in] frame_idx : INTEGER
[out] posreg_idx : INTEGER

DETAIL

group_idx: Group number.
frame_idx: Coordinate system index. 1 is joint coordinate, 2 is Cartesian coordinate.
posreg_idx: Position register number to output result.

SAMPLE

CALL STPOSPRG (1,2,3) -> Output the Cartesian coordinate of G1 to PR[3].

7.13 CHECK THE ROBOT AXES

It is allowed to check the status of the robot axes.

7.13.1 Get Current

This function gets current value of specified axis of specified group and output the value to specified register. The group axis is specified by the 1st and 2nd parameter respectively. The register is specified by the 3rd parameter. Unit of current is A.

NOTE
This function gets and outputs current at the time of it is called. Register value is not updated along with change of current automatically.

KAREL PROGRAM NAME: GTAXSCUR

SYNTAX

GTAXSCUR(group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number of specified group as 1st parameter.
register_idx: Register number.

SAMPLE

CALL GTAXSCUR(1, 1, 1) -> R[1] = 22.708 (If current of G1, J1 is 22.708A)

7.13.2 Get Torque

> **NOTE**
> This function gets and outputs torque at the time of it is called. Register value is not updated along with change of torque automatically.

This function gets and output following data to register.
・ "Current" value in DISTURBANCE TORQUE screen ([MENU] key -> 0-- NEXT -- -> 4 STATUS -> NEXT key -> F4[DISTURB]).
・ "Ave." and "Max" values in MONITOR screen ([MENU] key -> 0-- NEXT -- -> 4 STATUS -> NEXT key -> F2[MONITOR]).

7.13.2.1 Get "Current" of DISTURBANCE TORQUE screen

This function gets "Current" value of specified axis of specified group in DISTURBANCE TORQUE screen and outputs the value to specified register.

The group and axis are specified by the 1st and 2nd parameter respectively. The register is specified by the 3rd parameter. Unit of current is A.

KAREL PROGRAM NAME: GTDTBTRQ

SYNTAX

GTDTBTRQ (group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number of specified group as 1st parameter, group_idx.
register_idx: Register number.

SAMPLE

CALL GTDTBTRQ(1, 1, 1) -> R[1] = 1.01 (If current of G1, J1 is 1.01A)

7.13.2.2 Get "Ave." of MONITOR screen

This function gets "Ave." value of specified axis of specified group in MONITOR screen and outputs the value to specified register.
It outputs the average of torque monitor from starting the program to executing this function.
The group and axis are specified by the 1st and 2nd parameter respectively. The register is specified by the 3rd parameter. Unit of current is A.

KAREL PROGRAM NAME: GTTRQAVE

SYNTAX

GTTRQAVE (group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number of specified group as 1st parameter, group_idx.
register_idx: Register number.

SAMPLE

CALL GTTRQAVE(1, 1, 1) -> R[1] = 16.02 (If “Ave.” value of G1, J1 is 16.02A)

7.13.2.3 Get “Max.” of MONITOR screen

This function gets “Max.” value of specified axis of specified group in MONITOR screen and outputs the value to specified register.
It outputs the maximum value of torque monitor from starting the program to executing this function.
The group and axis are specified by the 1st and 2nd parameter respectively. The register is specified by the 3rd parameter. Unit of current is A.

KAREL PROGRAM NAME: GTTRQMAX

SYNTAX

GTTRQMAX (group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number of specified group as 1st parameter, group_idx.
register_idx: Register number.

SAMPLE

CALL GTTRQMAX(1, 1, 1) -> R[1] = 58.719 (If “Max.” value of G1, J1 is 58.719A)

7.13.2.4 “Ave.” and “Max.” in MONITOR screen

“Ave.” and “Max.” in MONITOR screen are the value of average and maximum while executing the program or jog motion. If the program or jog motion is paused, these values are output newly after the program or jog motion is restarted regardless of whether servo is OFF and ON at restarting.

7.13.3 Get Max Current

Get max current of the specified group and axis to the register.

KAREL PROGRAM NAME: GTMAXCUR

SYNTAX

GTMAXCUR(group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number to get max current.
register_idx: Register number to output result.

SAMPLE

CALL GTMAXCUR (1,2,3) -> Output max current of J2 of G1 to R[3].

7.13.4 Check Current

Check the current whether it is reached the specified value or not until time out.

KAREL PROGRAM NAME: GTCURCUR

SYNTAX

GTCURCUR(group_idx, axis_idx, cmp_reg_idx, timeout_reg_idx, register_idx)

[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[in] cmp_reg_idx : INTEGER
[in] timeout_reg_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number.
axis_idx: Axis number to get the value of current.
cmp_reg_idx: Register number that has the value of comparing current, unit is A.
timeout_reg_idx: Register number that has the time out value, unit is msec.
register_idx: Register number to output the result. If the current is over R[cmp_reg_idx] value, R[register_idx] is set 1. If time out, R[register_idx] is set -1.

SAMPLE

```
   R[3] = 1.5
   R[4] = 3000
CALL GTCURCUR (1,2,3,4,5)
```

Check whether the current of J2 of G1 reaches to R[3], 1.5A, value. Time out value is specified as 3 seconds, equal to 3000 msec, to R[4]. The result of checking is output to R[5].

7.14 SET STRING TO SYSTEM VARIABLE

⚠ **WARNING**
The operation of the robot and controller is controlled with system variables. Only a person who knows details of the influence of changes in system variables should set system variables. If a person without detailed knowledge attempts to set the system variables, the robot and controller would malfunction.

Set string, 2nd parameter, to the system variable, 1st parameter. Each parameter is allowed to set string or string register number.

KAREL PROGRAM NAME: STSYSSTR

SYNTAX

STSYSSTR(sysvar_name, str_data)
[in] sysvar_name : STRING or INTEGER
[in] str_data : STRING or INTEGER

DETAIL

sysvar_name: System variable name or string register number that has system variable name.
str_data: String or string register number that has string to set to system variable.

SAMPLE

CALL STSYSSTR (1, 2)
Set the string data of SR[2] to the system variable saved in SR[1].
For example, if SR[1] and SR[2] is following, ALARM is set to $LOG_BUFF[3].$TITLE.
SR[1] = ‘$LOG_BUFF[3].$TITLE’
SR[2] = ‘ALARM’

Above sample is the same below.
CALL STSYSSTR (‘$LOG_BUFF[3].$TITLE’, ‘ALARM’)

7.15 GET NUMBER OF COLLISION DETECT

Get the number of collision detect to the register.

KAREL PROGRAM NAME: GTCDCNT

SYNTAX

GTCDCNT(group_idx, axis_idx, register_idx)
[in] group_idx : INTEGER
[in] axis_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

group_idx: Group number. If 0 is specified, all groups are intended.
axis_idx: Axis number to get collision detect. If 0 is specified, all axes are intended.
register_idx: Register number to output the result.

NOTE
If 0 is set to group_idx, axis_idx must be 0.

SAMPLE

CALL GTCDCNT(1, 6, 20)
Set the number of collision detect of J6 of G1 to R[20].

CALL GTCDCNT(1, 0, 20)
Set the max number of collision detect of all axes of G1 to R[20].

CALL GTCDCNT(0, 0, 20)
Set the max number of collision detect of all axes of all group to R[20].

7.16 TP Program to Change Payload Setting

KAREL program STPAYLOD can change payload setting at TP program. After execution of the program, PAYLOAD instruction of TP program must be executed.
There are 10 schedules of payload setting for each group. If you need more payload schedules, STPAYLOD can be useful. You can change payload setting dynamically by using STPAYLOD and PAYLOAD instruction.

STPAYLOD is included in system design tool (J738). It is available on 7DC1/12 and 7DD0/07 or later.

KAREL PROGRAM NAME: STPAYLOD

SYNTAX

STPAYLOD (group_idx, payload_sch, reg_1st_idx)
[in] group_idx: INTEGER
[in] payload_sch: INTEGER
[in] reg_1st_idx : INTEGER

DETAIL

group_idx : group number
payload_sch: schedule number of payload setting to change
reg_1st_idx: Register number of the first register that stores setting

Supposing reg_1st_idx is n, setting of payload schedule is specified by following registers.
R [n]: Payload [kg]

R [n+1]: Position of center of gravity, X [cm]
R [n+2]: Position of center of gravity, Y [cm]
R [n+3]: Position of center of gravity, Z [cm]
R [n+4]: Inertia about X [kgfcm^2]
R [n+5]: Inertia about Y [kgfcm^2]
R [n+6]: Inertia about Z [kgfcm^2]
Registers of 7 consecutive numbers are used by STPAYLOAD.

Payload should be more than equal to 0 and less than equal to 9999.99.
Position of center of gravity X, Y and Z should be more than equal to -999.99 and less than equal to 999.99.
Inertia about X, Y and Z should be more than equal to 0 and less than equal to 50000.

SAMPLE

This example changes payload setting of schedule number 10.
Data to be set are stored in 7 registers. The 1st one is specified by the 3rd argument of STPAYLOD.
R [21]: Payload [kg]
R [22]: Position of center of gravity, X [cm]
R [23]: Position of center of gravity, Y [cm]
R [24]: Position of center of gravity, Z [cm]
R [25]: Inertia about X [kgfcm^2]
R [26]: Inertia about Y [kgfcm^2]
R [27]: Inertia about Z [kgfcm^2]

```
CALL STPAYLOD(1, 10, 21)
PAYLOAD [10]
```

After call of STPAYLOD, PAYLOAD instruction must be executed for schedule 10.

NOTE
STPAYLOAD must be used with PAYLOAD instruction.

NOTE

Following table shows typical alarms that are displayed by execution of STPAYLOD.

Error	Cause and remedy
SDTL-074 Illegal group(%s)	The group number specified by the 1st argument doesn't exist. %s shows specified group number. Specify motion group that exists on the controller.
SDTL-075 Illegal Payload schedule number(%s)	The payload schedule number specified by the 2nd argument doesn't exist. %s shows specified schedule number. Specify payload schedule number that exists on the controller.
SDTL-076 Payload is out of range(%s)	Payload is less than 0 or larger than 9999.99. %s shows register number and value of the register. "194, -0.010" for example. Set proper payload value.
SDTL-077 Payload center position is out of range(%s)	Position of center of gravity (X, Y or Z) is less than -999.99 or larger than 999.99. %s shows coordinate(X, Y or Z), register number and its value. "Z, 197, -1000.000" for example. Set proper value.
SDTL-078 Payload inertia is out of range(%s)	Inertia about X, Y or Z is less than 0 or larger than 50000. %s shows coordinate (X, Y or Z), register number and value of the register. "X, 198, -0.010" for example. Set proper value.
SDTL-079 TPE parameter %s wrong data type SDTL-013 Not INTEGER	Argument is not integer. Teach integer.
SDTL-080 Failed to get TPE parameter %s	The Nth TPE parameter does not exist. N is shown in %s. It is 1, 2 or 3. STPAYLOD needs 3 arguments. Teach 3 arguments.
SDTL-081 Register for payload does not exist (%s). SDTL-082 Register for payload center (%s) does not exist. SDTL-083 Register for payload inertia (%s) does not exist.	Register to specify value of payload setting does not exist. Set proper value to the 3rd argument so that required registers exist. The 3rd argument is register number for payload. STPAYLOD uses 7 registers of consecutive numbers, which start with the 3rd argument. R [n]: Payload [kg] R [n+1]: Position of center of gravity, X [cm] R [n+2]: Position of center of gravity, Y [cm] R [n+3]: Position of center of gravity, Z [cm] R [n+4]: Inertia about X [kgfcm^2] R [n+5]: Inertia about Y [kgfcm^2] R [n+6]: Inertia about Z [kgfcm^2] For example, if there are 200 registers, the 3rd argument must be less than equal to 194. %s shows used register number. SDTL-082, SDTL-083 also shows coordinate (X, Y or Z). For example, SDTL-082 can be displayed as "SDTL-082 Register for payload center (X, 200) does not exist .

8 OTHERS

This chapter describes about functions classified as “OTHERS”.

8.1 USER SPECIFIED ALARM POST

This function allows you to display user own alarm such as user alarm. However this is different from user alarm in that it can change the alarm number. Alarm ID is KALM, ID = 122. It is allowed to change the message along with the status of running TP programs and to identify the alarm number remotely to output it to external.

Range	Usable alarm number	Description
Global	1~200	An alarm is issued to all programs.
Local	201~300	An alarm is issued only to the program that caused the alarm.

NOTE
Use local alarm in the case the alarm is desired to be applied to only one program while multi programs are executed simultaneously by multitasking function.

8.1.1 Global Alarm Post

KAREL PROGRAM NAME: STRBERGL

SYNTAX

STRBERGL(alarm_idx, comment, severity_idx)
[in] alarm_idx : INTEGER
[in] comment : STRING or 0 (INTEGER)
[in] severity_idx : INTEGER

DETAIL

alarm_idx: Alarm number to display. Integer from 1 to 200.
comment: String to display. If integer 0 is specified, NULL string is displayed.
severity_idx: Alarm severity. Correspondence table of severity and 3rd parameter is following.

Correspondence table of severity and severity_idx

severity_idx	Alarm severity	Program	Robot operation	Power to servo system
0	STOP.G	pause	decelerate the robot slowly until it stops	none
1	WARN	none	none	
2	PAUSE.G	pause	decelerate the robot slowly until it stops	
3	STOP.G			
4	SERVO		stop the robot immediately	off

Refer to the Subsection “ERROR CODE PROPERTIES” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Alarm Code List) (B-83284EN-1) to know detail of alarm severity.
It is the same behavior when 0 or 3 are specified by the 3rd parameter.

SAMPLE

CALL STRBERGL(1, ‘Body Error’, 2) -> “KALM - 001 Body Error” is displayed with PAUSE.G severity.

8.1.2 Local Alarm Post

KAREL PROGRAM NAME: STRBERLO

SYNTAX

STRBERLO(alarm_idx, comment, severity_idx)
[in] alarm_idx : INTEGER
[in] comment : STRING or 0 (INTEGER)
[in] severity_idx : INTEGER

DETAIL

alarm_idx: Alarm number to display. Integer from 201 to 300.
comment: String to display. If integer 0 is specified, NULL string is displayed.
severity_idx: Alarm severity. Correspondence table of severity and 3rd parameter is following.

Correspondence table of severity and severity_idx

severity_idx	Alarm severity	Program	Robot operation	Power to servo system
0	STOP.L	pause	decelerate the robot slowly until it stops	none
1	PAUSE.L			
2	STOP.L			

Refer to the Subsection “ERROR CODE PROPERTIES” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Alarm Code List) (B-83284EN-1) to know detail of alarm severity.
It is the same behavior when 0 or 2 are specified by the 3rd parameter.

SAMPLE

CALL STRBERLO(250, ‘Body Error’, 2) -> “KALM - 250 Body Error” is displayed with PAUSE.L severity.

8.2 GET DATE, TIME

Following is the cases that it is useful to use this function.
・ Change the light of vision system along with specified time.
・ Doing maintenance to the save date or time.

8.2.1 Get Date

This function is to get date and output it to specified registers.
Year is output XX part of 20XX.

KAREL PROGRAM NAME: GTRBDATE

SYNTAX

GTRBDATE(reg_year, reg_month, reg_day)
[out] reg_year : INTEGER
[out] reg_month : INTEGER
[out] reg_day : INTEGER

DETAIL

reg_year: Register number to output year. If 0 is specified no year is output.
reg_month: Register number to output month. If 0 is specified no month is output.
reg_day: Register number to output day. If 0 is specified no day is output.

SAMPLE

CALL GTRBDATE(1, 2, 3) -> R[1] = 10, R[2] = 12, R[3] = 14 (if it is 2010/12/14)

8.2.2 Get Time

This function is to get time and output it to specified registers. Second date is output at intervals of 2 seconds.

KAREL PROGRAM NAME: GTRBTIME

SYNTAX

GTRBTIME(reg_hour, reg_minute, reg_second)
[out] reg_hour : INTEGER
[out] reg_minute : INTEGER
[out] regi_second : INTEGER

DETAIL

reg_hour: Register number to output hour. If 0 is specified no hour is output.
reg_minute: Register number to output minute. If 0 is specified no minute is output.
reg_second: Register number to output second. If 0 is specified no second is output.

SAMPLE

CALL GTRBTIME(1, 2, 3) -> R[1] = 18, R[2] = 30, R[3] = 24 (if it is 18:30:24)

8.3 SET LANGUAGE

NOTE
Please cycle power after switching the language.
Refresh screen described below is useful after switching language, however, cycle power is needed.

8.3.1 Set English

Set language to English on teach pendant. “SDTL-005 Changed to English” is displayed after setting.
If English dictionary is not ordered, “SDTL-003 No English dictionary” is displayed.

KAREL PROGRAM NAME: STLANGEG

SYNTAX

STLANGEG

DETAIL

There is no parameter.

SAMPLE

CALL STLANGEG

8.3.2 Set Japanese

Set language to Japanese on teach pendant. “SDTL-006 Changed to Japanese” is displayed after setting.
If Japanese dictionary is not ordered, “SDTL-004 No Japanese dictionary” is displayed.

KAREL PROGRAM NAME: STLANGJP

SYNTAX

STLANGJP

DETAIL

There is no parameter.

SAMPLE

CALL STLANGJP

8.3.3 Refresh Screen of Teach Pendant

Refresh the screen of teach pendant. This function is called after Set Language. Please refer to the section 9.5 "REFRESH PANEL AFTER SETTING ENGLISH".

KAREL PROGRAM NAME: REDISP04

SYNTAX

REDISP04

DETAIL

There is no parameter.

SAMPLE

CALL REDISP04

8.4 SELECT TP PROGRAM

> **NOTE**
> TP program name to set string register must be used the allowed characters. Refer to the section “PROGRAM DETAIL INFORMATION” in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN).

Select the TP program specified the value of string register as 1st parameter. The result is output to the register specified as 2nd parameter. If the program is selected successfully, 0 is output to the register. –1 is output in the case the program is paused by some alarm. Over 1 is output if the program detects some alarm and it is finished.

Output	Description
-1	Stopped by alarm in KAREL program.
0	TP program selected successfully.
1	Illegal 1st parameter.
2	1st parameter is not integer.
3	Specified string register is NULL.
4	Failed to set program.
5	Failed to check status. TPP must not be selected. Probably specified program is paused or run.
6	Specified program name is too long.
7	Failed to get string register.
8	None of specified program
9	Time out is occurred when selecting program.
10	2 byte character is set in specified string register.
11	First character is number.
12	Invalid character is used as program name.
13	Program name could not get.

Set this program to the macro. Refer to the Section "MACRO INSTRUCTION" in the Chapter "UTILITY" in R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN).

KAREL PROGRAM NAME: STPRGNAM

SYNTAX

STPRGNAM(sreg_idx, register_idx)
[in] sreg_idx : INTEGER
[out] register_idx : INTEGER

DETAIL

sreg_idx: String register number that has the program name to be set.
register_idx: Register number to output the result of setting the program

SAMPLE

CALL STPRGNAM(1, 2)   (Select the program set in SR[1], result is output to R[2].)

8.5 Clear of position register

KAREL program CRPOSREG initializes data of specified group of specified position register. The program is part of system design tool (J738). The program is available on software 7DC1/06 or later.

KAREL PROGRAM NAME:CRPOSREG

SYNTAX

CRPOSREG (preg_idx, group_idx, cmt_clr)
[in] preg_idx: INTEGER
[in] group_idx: INTEGER
[in] cmt_clr: INTEGER

DETAIL

preg_idx specifies index of position register to clear.
group_idx specifies group to clear. If 0 is specified, data of all group of specified PR is cleared.
If cmt_clr is 1, comment is cleared. If it is 0, comment is not changed.

SAMPLE

Following example clears group 1 of position register 10. Comment is not changed.

CALL CRPOSREG(10, 1, 0)

NOTE
Suppose there is group with no axis, independent axis before addition of axes for example. CRPOSREG cannot initialize position register of such group if it is recorded in joint position.

8.6 OUTPUT ERROR HISTORY FILE

NOTE
The program is available on software 7DC1/07 or later.

KAREL PROGRAM NAME: ERFILECP

SYNTAX

ERFILECP (file_path)
[in] file_path : STRING or INTEGER

DETAIL

file_path: string that indicates device and directory or integer that indicates the number of the string register. File of error history is output to the path specified as file_path.

Target file to save is ERRACT.LS and ERRALL.LS.
If integer is specified, it is regarded as the index of string register. The value of specified string register is treated as device and directory.

Suffix is added to saved file name. The suffix represents year, month, date and time.
It is like ERRALL_yymmdd_hhnnss.LS or ERRACT_ yymmdd_hhnnss.LS.

Following is the meaning of the symbol of the suffix.

yy: year
mm: month
dd: day
hh: hour
nn: minute
ss: second (every 2 seconds)

For example, if it is the date is 12.15.2pm on August/22/2012, ERRALL_120822_121502.LS and ERRACT_120822_121502.LS are saved.

The same time is added to each file at the timing that saving ERRALL.LS is started for corresponding the file.

If file with the same already exists , it is overwritten.
The alarm is displayed when nonexistent subdirectory is specified. If you want to use subdirectory on Cn: , it must be supported on server side. If “:¥” is not found in file_path, the alarm is displayed.
In the case subdirectory is used, ¥ must be added at the end of file_path. MC:¥SUBDIR¥ is no problem, however, MC:¥SUBDIR causes alarm.

NOTE
1 Keep enough vacant space for the destination device.
2 FAT32 format is recommended in the case a lot of files are saved to root directory of MC:.

SAMPLE 1
CALL ERFILECP (‘MC:¥ERRINFO¥’)
Output the file that has error information to subdirectory, ERRINFO, of MC:.

SAMPLE 2
CALL ERFILECP (1)
Output the file that has error information to root of MC: when SR[1] = ‘MC:¥’.

8.7 WRITE PROTECTION ON

NOTE
The program is available on software 7DC1/07 or later.

KAREL PROGRAM NAME:STWRPRT
SYNTAX
STWRPRT (prog_name)
[in] prog_name : STRING or INTEGER

DETAIL
prog_name: TP program name to be set write protection to on, or index of the string register which has TP program name. Wild card (*) cannot be used for specifying TP program name.

NOTE

1. Turn on write protection of only TP program created by user. Do not turn on write protection of TP programs loaded as default. Some TP programs are created by executing some function. Do not turn on write protection of unfamiliar TP program that is not created by user.
2. If TP program is already used, write protection may not be set to on. Followings are examples.
   - Running TP program. TP program called from the running TP program.
   - Being used TP program by other function.

SAMPLE 1

CALL STWRPRT (‘PNS0001’)
Write protection of PNS0001.TP is set to on.

SAMPLE 2

CALL STWRPRT (3)
This example turns on write protection of TP program whose name is stored in SR[3].
If SR[3] = ‘PNS0003’, write protection of PNS0003.TP is turned on.

RELATED ALARM

Severity	Description	Alarm
STOP.G	The first parameter is REAL number.	SDTL-047 TPE parameter %s wrong data type
	Index of string register is wrong.	SDTL-068 Index of SR is wrong(%s)
WARN	Specified TP program does not exist on the controller.	SDTL-064 TP Program does not exist (%s)
	Program name was not specified.	SDTL-065 Program name is not specified %s
	Program name contains wild card, "*".	SDTL-066 * cannot be used (%s)
	Write protection could not be turned on.	SDTL-067 Failed to turn write protect on(%s)
	Specified program is not TP program.	SDTL-070 Specified program is not TP(%s)

- Long TP program name is not displayed correctly on %s in some cases. This is limitation.
- String register (SR[]) information is displayed on %s of SDTL-64, 65, 66, 67 and 70 if TP program is specified by string register like “SR[1]: TP program name”.

8.8 WRITE PROTECTION OFF

NOTE
The program is available on software 7DC1/07 or later.

KAREL PROGRAM NAME:CRWRPRT
SYNTAX

CRWRPRT (prog_name)
[in] prog_name : STRING or INTEGER

DETAIL

prog_name: TP program name to be set write protection to off, or index of the string register which has TP program name. Wild card (*) cannot be used for specifying TP program name.

SAMPLE 1

CALL CRWRPRT (‘PNS0001’)

Write protect of PNS0001.TP is set to off .

SAMPLE 2

CALL CRWRPRT (3)
This examples turns off write protection of TP program whose name is stored in SR[3].
If SR[3] = 'PNS0003', write protection of PNS0003.TP is turned off.

NOTE

1 Turn off write protect of only TP program created by user. Do not turn off write protection of TP programs loaded as default. Some TP programs are created by executing some function. Do not turn off write protection of unfamiliar TP program that is not created by user.
2 If TP program is already used, write protection may not be set to off. Followings are examples.
   - Running TP program. TP program called from the running TP program.
   - Being used TP program by other function.

RELATED ALARM

Severity	Description	Alarm
STOP.G	The first parameter is REAL number.	SDTL-047 TPE parameter %s wrong data type
	Index of string register is wrong.	SDTL-068 Index of SR is wrong(%s)
WARN	Specified TP program does not exist on the controller.	SDTL-064 TP Program does not exist (%s)
	TP Program name was not specified.	SDTL-065 Program name is not specified %s
	TP Program name contains wild card, "*".	SDTL-066 * cannot be used (%s)
	Write protection could not be turned off.	SDTL-069 Failed to turn write protect off(%s)
	Specified program is not TP program.	SDTL-070 Specified program is not TP(%s)

- Long TP program name is not displayed correctly on %s in some cases. This is limitation.
- String register (SR[]) information is displayed on %s of SDTL-64, 65, 66, 69 and 70 if TP program is specified by string register like "SR[1]: TP program name".

9 APPLICATION SAMPLE

This chapter shows the application sample with KAREL programs of System Design Tool.

9.1 SAMPLE TO CHECK AND OUTPUT STATUS TO DO

9.1.1 The Macro to Output Single Step Status to DO

This section shows the sample to output single step status to DO.

Create following TP program (CHK_STEP.TP). Set no group motion (set motion group [*,*,*,*,*,*,*,*] in program detail screen).

<CHK_STEP.TP>

```
CALL CKSGLSTP(1) ;
IF R[1]=1,JMP LBL[1] ;
 ;
DO[1]=OFF ;
END ;
 ;
LBL[1] ;
DO[1]=ON ;
END ;
```

Select [MENU] -> “6. SETUP” -> Macro. Set CHK_STEP.TP to execute when DI[1] is turned ON. Please refer to the section “MACRO INSTRUCTION” in the chapter “UTILITY” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to set the macro program.

CHK_STEP.TP is executed when DI[1] is turned ON. If it is single step mode DO[1] is ON. If it is not single step DO[1] is OFF.

Other status such as group motion (machine lock) is allowed to set macro by the same way above.

9.2 ALARM IS DISPLAYED IF I/O IS SIMULATED

9.2.1 Sample 1 of Check DI SIMULATED STATUS

Following is the sample to display warning if DI is simulated.

<CHK_DI.TP>

```
CALL CKDIOSIM(1,0,1) ;
IF R[1]=1,JMP LBL[1] ;
 ;
CALL STRBERGL(2,'No DI simulated',1) ;
END ;
 ;
LBL[1] ;
CALL STRBERGL(1,'DI simulated',2) ;
END ;
```

CKDIOSIM checks all of DI whether simulated or not and output result to R[1].
If one or more DI is simulated, “KALM - 001 DI simulated” is displayed with PAUSE.G severity.
If no DI is simulated, “KALM - 002 No DI simulated”, is displayed with WARN severity.

Other check status function such as group motion (machine lock) is allowed to create the same way above.

9.2.2 Sample 2 of Check DI SIMULATED STATUS

Following sample is to check from DI[1] to DI[10] and display alarm with STOP.G severity if simulated DI exists.

Set string register as follows.

```
SR[ 1:    ]=
SR[ 2:    ]=DI[
SR[ 3:    ]=] simulated
```

```
R[1]=0      ;
R[2]=0      ;
  ;
LBL[1] ;
R[1]=R[1]+1      ;
IF R[1]>10,JMP LBL[3] ;
CALL CKDIOSIM(1,R[1],2) ;
IF R[2]=1,JMP LBL[2] ;
JMP LBL[1] ;
  ;
LBL[2] ;
SR[1]=SR[2]+R[1]+SR[3]      ;
CALL STRBERGL(1,SR[1],0) ;
R[2]=0      ;
JMP LBL[1] ;
  ;
LBL[3] ;
END ;
```

If DI[3] is simulated, “KALM - 001 DI[3] simulated” is displayed with STOP.G severity.

9.3 PERIODIC MAINTENANCE

Following is the sample to call TP program (MAINTENANCE.TP) for maintenance if the date is 10.

```
CALL GTRBDATE(0,0,1) ;
IF R[1]=10,CALL MAINTENANCE ;
```

9.4 SELECT TP PROGRAMS BY WORKS

Following is the sample to select TP programs depending on works. Suppose R[1] has the work number. TP programs listed in MC:¥W1¥WORK1.DT or MC:¥W2¥WORK2.DT are loaded by the value of R[1].

```
SELECT R[1]=1,JMP LBL[1] ;
       =2,JMP LBL[2] ;
       ELSE,JMP LBL[3] ;
 ;
LBL[1] ;
CALL EXT_LOAD('MC:¥W1¥WORK1') ;
CALL PNS0011    ;
CALL PNS0012    ;
CALL PNS0013    ;
CALL EXT_DEL('MC:¥W1¥WORK1 ') ;
END ;
 ;
LBL[2] ;
CALL EXT_LOAD('MC:¥W2¥WORK2') ;
CALL PNS0021    ;
CALL PNS0022    ;
CALL PNS0023    ;
CALL EXT_DEL('MC:¥W2¥WORK2 ') ;
END ;
 ;
LBL[3] ;
END ;
```

Suppose each file list is as follows.

<MC:¥W1¥WORK1.DT>

```
PNS0011
PNS0012
PNS0013
```

<MC:¥W2¥WORK2.DT>

```
PNS0021
PNS0022
PNS0023
```

9.5 REFRESH PANEL AFTER SETTING ENGLISH

Following is an sample to switch the language to English and refresh the panel.

```
CALL STLANGEG    ;    Set English
CALL REDISP04    ;    Refresh panel
```

NOTE
Please cycle power after switching the language.
Refresh screen described above is useful after switching language, however, cycle power is needed.

9.6 SELECT TP PROGRAM

Create SET_PROG.TP like following.
Set SET_PROG.TP as macro program to run if DI[1] is ON. Please refer to the section “MACRO INSTRUCTION” in the chapter “UTILITY” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to set the macro program.

```
SELECT R[10:WORK]=1,JMP LBL[1] ;     Suppose R[10] is the number
                                       of work.
          =2,JMP LBL[2] ;            After DI[1] is ON, this program is
                                     run respond to R[10].
          ELSE,JMP LBL[3] ;
 ;
LBL[1] ;
CALL STPRGNAM(1,1) ;             Select TP program set in SR[1].
JMP LBL[4] ;
 ;
LBL[2] ;
CALL STPRGNAM(2,1) ;             Select TP program set in SR[2].
JMP LBL[4] ;
 ;
LBL[3] ;
CALL STRBERGL(1,'WORK ERROR',1) ;     Display “KALM-001 WORK
                                      ERROR”.
END ;
 ;
LBL[4] ;
IF R[1]<>0,CALL STRBERGL(2,'FAILED TO SELECT PROGRAM',1) ;
                    Display “KALM-002 FAILED TO SELECT PROGRAM”.
END ;
```

9.7 CHECK COLLISION DETECT AND DISPLAY USER ALARM

Following is the sample to check the number of collision detect and display the user alarm if it is over the desired count (it suppose 3 in this sample).

In this sample, there are 5 registers, from R[1] to R[5], to user like following.

Resister	Description
1	Get the count of all collision detect in J6 axis.
2	The number of collision detect from displaying user alarm 1 at last in J6 axis.
3	All count of collision detect in the time of user alarm is displayed at last.
4	Tolerance of collision detect of J6 axis. It supposes 3 in this sample.
5	The axis number to be displayed alarm in group 1.

Because R[4] depends on each system, please set R[4] in adequate value. In the case that the user alarm would be displayed if collision detect is over 5, set R[4] = 5.

Create 2 TP programs below (G1J6_RST.TP, G1J6_CHK.TP). G1J6_RST.TP is executed at first. R[3] has the collision detect count of axis 1 in group 1.

<G1J6_RST.TP>

```
1:  CALL GTCDCNT(1,6,3) ;
```

G1J6_CHK.TP is executed to check the number of the collision detect.
The user alarm is displayed if the number of collision detect is over 3 (R[4]). And R[3] is refreshed.

<G1J6_CHK.TP>

```
1:  CALL GTCDCNT(1,6,1) ;
2:  R[2]=R[1]-R[3]     ;
3:  IF R[2]>R[4],JMP LBL[1] ;
4:  END ;
5:  LBL[1] ;
6:  R[5]=6     ;
7:  R[3]=R[1]     ;
8:  UALM[1] ;
```

NOTE
G1J6_RST.TP must be executed in some cases that the register is changed such as restoring the backup.

9.8 MONITOR THE CURRENT OF J1 AXIS

Following sample is to monitor the current of J1 axis of the robot (group 1) at a constant frequency, 0.1 second. In this sample TP program to monitor is set to “HOT START Autoexec program:”.

Teach programs to run the monitoring program, RUN_MONITOR.TP, and to monitor the current program, MONITOR_CURRENT.TP.

Set “Group Mask” to [*,*,*,*,*,*,*,*] and “Ignore pause” to ON in these TP programs. Refer to the Subsection “PROGRAM DETAIL INFORMATION” of R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know how to set these attribute.

<RUN_MONITOR.TP>

```
1:   RUN MONITOR_CURRENT ;
```

<MONITOR_CURRENT.TP>

```
1:   LBL[1] ;
2:   CALL GTAXSCUR(1,1,1) ;
3:   WAIT      .10(sec) ;
4:   JMP LBL[1] ;
```

NOTE
Do not set the WAIT time under 0.1 second. It affects the robot system in some cases. According to the software construction it also affects the robot system even if the WAIT time is over 0.1. Before running this sample or similar program in the system, careful confirmation must be needed. Please increate WAIT time if necessary.

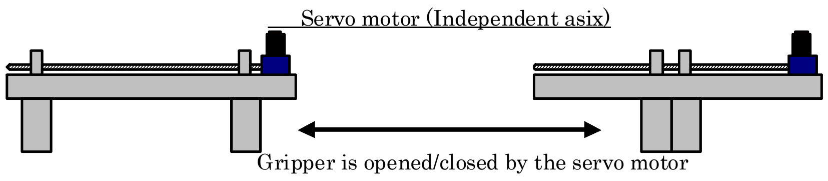
Set following in System/Config screen.
• “HOT START Autoexec program:” is RUN_MONITOR.TP
• “Multi Program Selection:” is TRUE

Refer to R-30*i*B/ R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) (B-83284EN) to know more about System/Config screen.

10 CHECK SERVO HAND GRIP STATUS

OVERVIEW

In this chapter, a servo hand is a servo driver gripper which moves by independent axis.


Servo motor (Independent asix)
Gripper is opened/closed by the servo motor

Example of servo hand

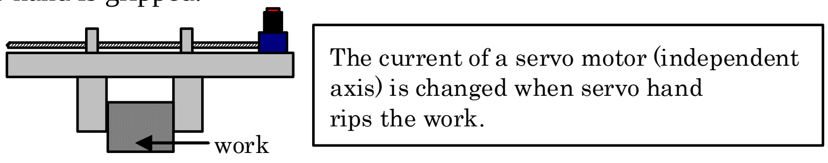
To confirm a servo hand grips the work correctly, following points are checked.

- The current of independent axis reaches specified value.
- The real position (*) of independent axis reaches gripper position of the work.

(*) Real position : Actual position of axis

Torque limit function option (J611) is usually used to control gripping force in the case of using servo hand. However, displayed axis position of current position screen is different from actual position of servo hand in some cases.

The current of independent axis is changed when it grips the work. The section 10.1 "CHECK THE SERVO HAND CURRENT" shows how to monitor the current to reach the specified value. It enables to confirm that servo hand is gripped.


work
The current of a servo motor (independent axis) is changed when servo hand rips the work.

Next the section 10.2 "GET SERVO HAND REAL POSITION" checks if the servo hand reach is gripper position.

NOTE

Refer to the chapter "TORQUE LIMIT FUNCTION" of R-30*i*B/ R-30*i*B Mate CONTROLLER Optional Function OPERATOR'S MANUAL (B-83284EN-2) to know torque limit function.

KAREL programs described below in this chapter are included in "Status Check/Set Function" category.

10.1 CHECK THE SERVO HAND CURRENT

NOTE

Torque limit function option (J611) is required. If it is not ordered, "SDTL-071 Torque Limit Function not installed" is displayed.

This function monitors whether current of the servo hand (independent axis) reached 95% (*) of the motor torque limited by the torque limit instruction or not until time out.

(*) Refer to the Section 10.5 "FOR CHECKING THE SERVO HAND CURRENT".

KAREL PROGRAM NAME: CKSVHCUR
SYNTAX

CKSVHCUR (group number, axis number, timeout_reg_idx, register number)
[in] group number : INTEGER
[in] axis number : INTEGER
[in] timeout_reg_idx : INTEGER
[out] register number : INTEGER

- group number: Set the group number including the servo motor using Torque Limit.
- axis number: Set the axis number of the servo motor using Torque Limit.
- timeout_reg_idx: Set the register number that has the time out value, unit is msec.
- register number: Set register number to output result.
  If the current is over 95% of the limited motor torque by torque limit instruction, R[register number] is set 1. If time out, R[register number] is set -1.

NOTE
"SDTL-072 Torque Limit disabled" is displayed if torque limit function is disabled in specified group or axis.

SAMPLE

```
R[3] = 3000;
TORQ_LIMIT 10%
RUN WORK_GRIP
CALL CKSVHCUR (2, 1,3,4);    (*)
```

If the maximum current of the amplifier is 40A, R[4] is set to 1 when the current value reaches to ±3.8A(40A×10%(torque limit)×95%= 3.8A) within time out (it is 3 seconds in this sample). If it takes over time, R[4] is set to -1.

(*)Refer to the section 10.6 "TO RESTART AFTER POWER OFF WHILE CHECKING", if power is off while checking the current.

10.2 GET SERVO HAND REAL POSITION

NOTE
GTRELPOS is only for independent axis. Never use the program to get position of the robot axes.

Displayed axis position of current position screen is different from actual position of servo hand in the case of using torque limit function. GTRELPOS sets the servo hand real position, unit is mm (translation axis) or deg (rotary axis), of the specified group and axis to the register.

KAREL PROGRAM NAME: GTRELPOS
SYNTAX

GTRELPOS (group number, axis number, register number)
[in] group number : INTEGER
[in] axis number : INTEGER
[out] register number : INTEGER

DETAIL

- group number: Set the group number to be got the real position.

- axis number: Set the axis number to be got the real position.
- register number: Register number to output the real position.

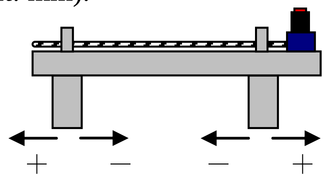
NOTE
"STDL-073 Group i not mastered" (i: group number) is displayed if the robot is not mastered on 7DC1/12, 7DDO/06 or later.

SAMPLE
CALL GTRELPOS (2,1,3) -> Output the real position of J1 of G2 to R[3].

10.3 SAMPLE OF CHECK SERVO HAND GRIP STATUS

⚠ **CAUTION**
Please confirm whether this sample could be applied or not to your system in advance. For example, some conditional branch instructions are reversed depending on the axis or frame configuration and so on.

This sample describes as group 1 is robot axes, group 2 is servo hand (independent axis), and servo hand is linear axis (unit: mm).



Open direction is plus, close direction is minus in this sample.

In this sample the closed up position of the servo hand is 0 mm.

10.3.1 Preparation for Sample TP Program

Load following KAREL programs from system design tool screen. Refer to the section 2.2 "LOAD KAREL PROGRAMS OF SYSTEM DESIGN TOOL" to know how to load.

- CHECK THE SERVO HAND CURRENT (CKSVHCUR)
- GET SERVO HAND REAL POSITION (GTRELPOS)
- USER SPECIFIED ALARM POST (STRBERGL)

10.3.1.1 How to Set up Torque Limit

To enable J1 axis in Group 2 uses Torque Limit, set TRUE to $G2[1], and set others FALSE.
$TORQUE_LMT.$GROUP[2] = TRUE
$TORQUE_LMT.$GA2[1] = TRUE
In this sample, Torque Limit function is not used for the other axes and groups. System variable for the other axes and groups are set to FALSE as shown below.

$TORQUE_LMT.$GROUP[1], [3]-[8] = FALSE (All groups but group 2)
$TORQUE_LMT.$GA1[1] - [9] = FALSE (All axes of group 1)
$TORQUE_LMT.$GA2[2] - [9] = FALSE (All axes of group 2 but axis 1)
$TORQUE_LMT.$GA3[1] - [9] = FALSE (All axes of group 3)
:
$TORQUE_LMT.$GA8[1] - [9] = FALSE (All axes of group 8)

NOTE

Refer to the chapter “TORQUE LIMIT FUNCTION” of R-30*i*B/ R-30*i*B Mate CONTROLLER Optional Function OPERATOR’S MANUAL (B-83284EN-2) to know torque limit function.

10.3.1.2 Register of Sample TP program

R[1] – R[8] are used in both the Subsection 10.3.3.1 and 10.3.3.2. R[9] is used only the Subsection 10.3.3.2 “Sample Program 2”.

Set the correct value in your system.

Register	Description
1	Sample program 1: This is used as skip condition instruction to terminate motion to grip the work when servo hand is about to open. If it is set to 1, the robot skips the motion.
	Sample program 2: Set 1 when it completes to grip. This is used as skip condition instruction when the servo hand grips the work.
2	Time out value to use monitoring the current of the servo hand. unit: msec
3	Result of monitoring the current of the servo hand.
4	Approach point that is near the grip point of the work. unit: mm
5	Grip point of the work. unit: mm
6	The value to press the servo hand into the work. unit :mm
7	The real position of the servo hand. unit: mm
8	Sample program 1: For interlock of the work gripping program(GRIP_WORK.TP) that is executed by multitasking function is done.
	Sample program 2: Set 1 when skip condition is not met and work gripping program(GRIP_WORK.TP) is over.
9	Interlock to confirm work gripping program(GRIP_WORK.TP) is done.

10.3.1.3 String Register of Sample TP program

SR[1] – SR[3] are used in this sample. SR[2] is only used in the Subsection 10.3.3.2 “Sample Program 2”.

The string registers are used to display alarm by KAREL program STRBERGL. The program is explained in section Section 8.1 “User Specified Alarm Post”. String for alarm should be set in the string registers in advance. Set each value like following.

String Register	Value
1	Time out
2	Miss to grip
3	Not to reach grip position

10.3.1.4 Position Register of Sample TP program

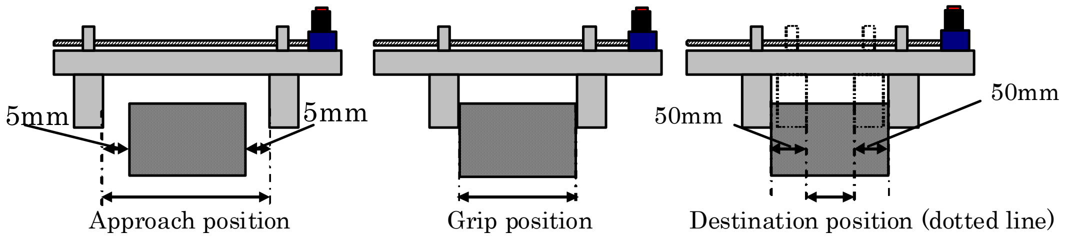
PR[1] – PR[2] are used for approach and destination position.

Position Register	Description
1	Approach point to grip work.
2	Destination position of the servo hand.

10.3.2 Each Position (Approach, Grip, Destination) of Sample TP program

In this sample approach point is taught 10 mm (5mm and 5mm from both end) away from grip position. To grip the work the servo hand moves faster (feed rate 100%) until approach point. It gets slower (feed rate 10%) from approach point to destination position.

The grip point is contact position with the work.
Destination position is to press the servo hand into the work R[6] value.(dotted line).
In following sample R[6] is set to 100 (50mm and 50mm from both end).



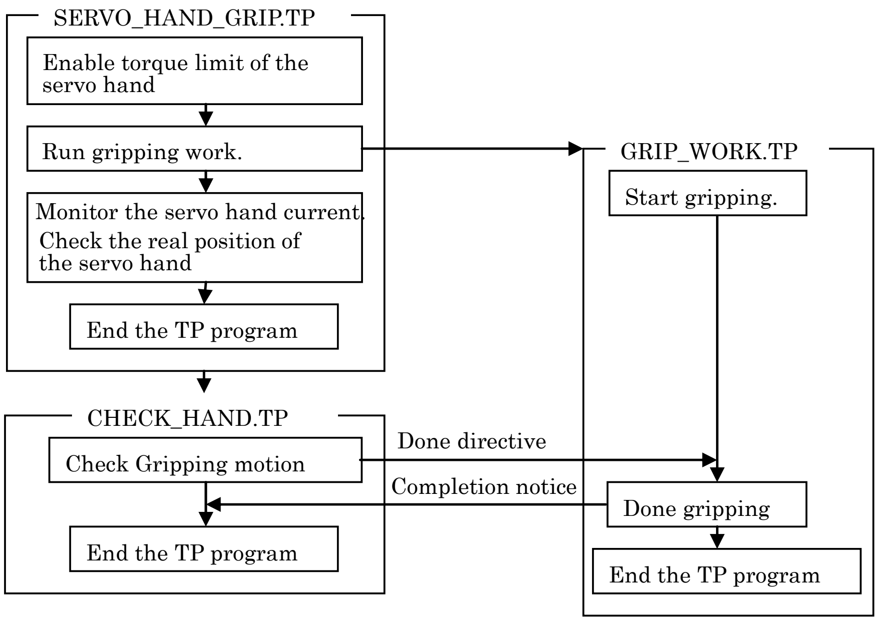
10.3.3 Sample TP Program

Two sample TP programs are prepared.

Sample TP program 1	To keep gripping force while gripping the work in following cases. (1) The work is easily change shape while it is gripped. (2) The servo hand is easily bent while it is gripping the work.
Sample TP program 2	To stop the servo hand motion after gripping the work. In the case it is not needed to keep gripping force after gripping the work.

10.3.3.1 Sample Program 1

Flow of sample TP program



```
                                   <MAIN.TP>                     Group Mask[*,*,*,*,*,*,*,*,]
   1:  !Move to eject position ;
   2:  CALL EJECT_POS      ;                (*1)
   3:   ;
   4:  !Set parameter for work ;
   5:  CALL WORK1_HAND_SET      ;      --(1)
   6:   ;
   7:  !Grip and check status ;
   8:  CALL SERVO_HAND_GRIP      ;      --(2)
   9:   ;
  10:  !Carry the work ;
  11:  CALL CARRY_WORK      ;              (*2)
  12:   ;
  13:  !Check the servo hand ;
  14:  CALL CHECK_HAND      ;               --(3)
  15:   ;
  16:  !Open the servo hand ;
  17:  CALL OPENHAND      ;                 --(4)
```

(*1): TP program to move to the eject position. In this sample EJECT_POS.TP is omitted.
(*2): TP program to carry the work. In this sample CARRY_WORK.TP is omitted.

```
(1)                         < WORK1_HAND_SET.TP>              Group Mask[*,*,*,*,*,*,*,*,]
   1:  !Set parameter ;
   2:  R[2:Time out(msec)]=3000      ;
   3:  R[4:Approach pos(mm)]=210      ;
   4:  R[5:Grip pos(mm)]=200      ;
   5:  R[6:Press val(mm)]=100      ;
```

(2) < SERVO_HAND_GRIP.TP> Group Mask[*,*,*,*,*,*,*,*]

```
LBL[100] ;
R[1:Done gripping]=0       ;
R[8:WORK_GRIP fin]=0       ;
!Start to grip ;
TORQ_LIMIT     10.0 %    ;
RUN GRIP_WORK ;     --(2-1) Run work gripping in multitasking
 ;
LBL[200] ;
!Monitor the current ;
CALL CKSVHCUR(2,1,2,3) ;
IF R[3:Result]<>1,JMP LBL[900] ;
 ;
!Check the real position ;
CALL GTRELPOS(2,1,7) ;
IF R[7:Real Pos(mm)]>R[5:Grip pos(mm)],JMP LBL[910] ;     (*3)
 ;
!Complete to grip ;                                       (*4)
END ;
 ;
!**Alarm** ;
LBL[900] ;
CALL STRBERGL(1,SR[1],0) ;
JMP LBL[200] ;
 ;
LBL[910] ;
CALL STRBERGL(2,SR[2],0) ;
JMP LBL[200] ;
```

(*3) The opposite of the condition of the sample can be appropriate for your system. It depends on the configuration of the servo hand and so on.
(*4) The program execution completes after the monitored current and the real position of the servo hand reach the specified value.

(2-1) < GRIP_WORK.TP> Group Mask[*,1,*,*,*,*,*,*,]

```
   1:  PR[1:Approach pos(mm)]=JPOS      ;
   2:  PR[2:Destination(mm)]=JPOS      ;
   3:  PR[GP2:1,1:Approach pos(mm)]=R[4:Approach pos(mm)]      ;
   4:  PR[GP2:2,1:Destination(mm)]=
         R[5:Grip pos(mm)]-R[6:Press val(,mm)]      ;              (*5)
   5:   ;
   6:  SKIP CONDITION R[1:Done gripping]=1      ;               (*6)
   7:J PR[1:Approach pos(mm)] 100% CNT0      ;
   8:J PR[2:Destination(mm)] 10% FINE Skip,LBL[300]      ;    (*6)
   9:   ;
  10:  LBL[300] ;                                             (*6)
  11:  R[8:WORK_GRIP fin]=1      ;                            (*6)
  12:  END ;                                                  (*6)
```

(*5) Depending on the configuration of the servo hand and other setting, plus and minus sign is different from this sample.

(*6) If there is enough time for the work gripping motion to complete before the servo hand is opened, you don't have to teach skip instruction. If there isn't, it is better to teach skip instruction like above sample program.

(3) <CHECK_HAND.TP> Group Mask[*,*,*,*,*,*,*,*,]

```
   1:  IF R[8:WORK_GRIP fin]<>1,JMP LBL[100] ;
   2:  END ;
   3:   ;
   4:  LBL[100] ;
   5:  R[1:Done gripping]=1      ;
   6:  WAIT R[8:WORK_GRIP fin]=1      ;
   7:  END ;
```

If the work gripping motion is still being executed when the servo hand should be opened, the TP program to open the servo hand (OPENHAND.TP) should not be called. Multi TP programs which have the same group mask are not executed simultaneously. "INTP-222 (program, i) Call program failed" and "PROG-040 Already locked by other task" are displayed.

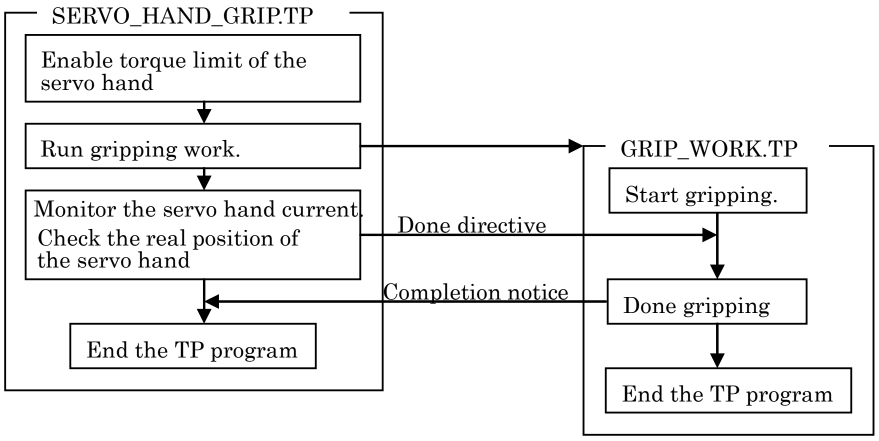
(4) <OPENHAND.TP> Group Mask[*,1,*,*,*,*,*,*,]

```
   1:  PR[1:Approach pos(mm)]=JPOS      ;
   2:  PR[GP2:1,1:Approach pos(mm)]=R[4:Approach pos(mm)]      ;
   3:  !Torque limit is 10% in next movement.   ;
   4:J PR[1:Approach pos(mm)] 100% CNT0      ;
   5:  TORQ_LIMIT   100.0 %   ;
   6:J P[1] 100% CNT0      ;
```

⚠ **CAUTION**
In the case the servo hand is opened, never to increase torque limit value until approach point. If torque limit is changed to 100% or is increased, the axis moves to destination position. Therefore work piece or servo hand may be broken.

10.3.3.2 Sample Program 2

Flow of sample program



<MAIN.TP> **Group Mask[*,*,*,*,*,*,*,*]**

```
 1:  !Move to eject position ;
 2:  CALL EJECT_POS      ;          (*1)
 3:    ;
 4:  !Set parameter for work ;
 5:  CALL WORK1_HAND_SET     ;   (*2)
 6:    ;
 7:  !Grip and check status ;
 8:  CALL SERVO_HAND_GRIP     ; --(1)
 9:    ;
10:  !Carry the work ;
11:  CALL CARRY_WORK      ;        (*3)
12:    ;
13:  !Open the servo hand ;
14:  CALL OPENHAND      ;          (*2)
```

(*1) TP program to move to the eject position. In this sample EJECT_POS.TP is omitted.
(*2) WORK1_HAND_SET.TP and OPENHAND.TP are the same as the Subsection "Sample Program 1". Refer to it.
(*3) TP program to carry the work. In this sample CARRY_WORK.TP is omitted.

(1) **< SERVO_HAND_GRIP.TP>** **Group Mask[*,*,*,*,*,*,*,*]**

```
LBL[100] ;
R[1:Done gripping]=0       ;
R[8:WORK_GRIP fin]=0       ;
R[9:Wait grip]=0       ;
!Start to grip ;
TORQ_LIMIT      10.0 %    ;
RUN GRIP_WORK ;      --(2-1) Run work gripping in multitasking
  ;
LBL[200] ;
!Monitor the current ;
CALL CKSVHCUR(2,1,2,3) ;
IF R[3:Result]<>1,JMP LBL[900] ;
  ;
!Check the real position ;
CALL GTRELPOS(2,1,7) ;
IF R[7:Real Pos(mm)]>R[5:Grip pos(mm)],JMP LBL[920] ; (*4)
  ;
!Complete to grip ;
R[1:Done gripping]=1       ;                  (*5)
WAIT R[9:Wait grip]=1       ;
END ;
  ;
!**Alarm** ;
LBL[900] ;
IF R[8:WORK_GRIP fin]=1,JMP LBL[910] ;
CALL STRBERGL(1,SR[1],0) ;
JMP LBL[200] ;
  ;
LBL[910] ;
CALL STRBERGL(2,SR[2],0) ;
CALL OPENHAND       ;
JMP LBL[100] ;
  ;
LBL[920] ;
CALL STRBERGL(3,SR[3],0) ;
JMP LBL[200] ;
```

(*4) The opposite of the condition of the sample can be appropriate for your system. It depends on the configuration of the servo hand and so on.

(*5) After confirming gripping the work by monitoring that current and the real position of the servo hand reached the specified value, R[1:Done gripping] is set to 1. It causes skip of gripping motion of GRIP_WORK.TP, which has been run by multitasking.

(2-1) < GRIP_WORK.TP> Group Mask[*,1,*,*,*,*,*,*,]

```
  1:   PR[1:Approach pos(mm)]=JPOS       ;
  2:   PR[2:Destination(mm)]=JPOS       ;
  3:   PR[GP2:1,1:Approach pos(mm)]=R[4:Approach pos(mm)]       ;
  4:   PR[GP2:2,1:Destination(mm)]=
     R[5:Grip pos(mm)]-R[6:Press val(mm)]     ;     (*6)
  5:     ;
  6:   SKIP CONDITION R[1:Done gripping]=1       ;           (*7)
  7:J PR[1:Approach pos(mm)] 100% CNT0       ;
  8:J PR[2:Destination(mm)] 10% FINE Skip,LBL[300]       ; (*7)
  9:   R[9:Wait grip]=1       ;
 10:   END ;
 11:     ;
 12:   LBL[300] ;                             (*7)
 13:   R[8:WORK_GRIP fin]=1       ;           (*7)
 14:   R[9:Wait grip]=1       ;               (*7)
 15:   END ;                                  (*7)
```

(*6) Depending on the configuration of the servo hand and other setting, plus and minus sign is different from this sample.

(*7): Skip condition instruction should be deleted in some cases that work is very soft or easily to become deformed while gripping.

10.4 STOP/MOVE ERROR EXCESS, INPOS TIME OVER

It varies according to the work, servo hand or target position, following alarms are tended to display when torque limit function is used.

- SRVO-023 Stop error excess(G:%d A:%d)
- SRVO-024 Move error excess(G:%d A:%d)
- SRVO-036 Inpos time over (G:%d A:%d)

It is necessary to set following system variables to large value in advance.
$PARAM_GROUP[group].$STOPTOL[axis]
$PARAM_GROUP[group].$STOPERLIM[axis]
$PARAM_GROUP[group].$MOVER_OFFST[axis]

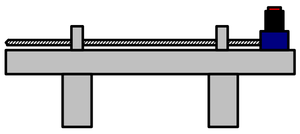
However, if too large value is set, these alarms are not displayed when they should be displayed. Next subsection shows the sample to change them.

10.4.1 Change the system variable of axis in torque limit

This subsection shows example of adjustment of the system variables. It is assumed that axis 1 of group 2 is used for servo hand.

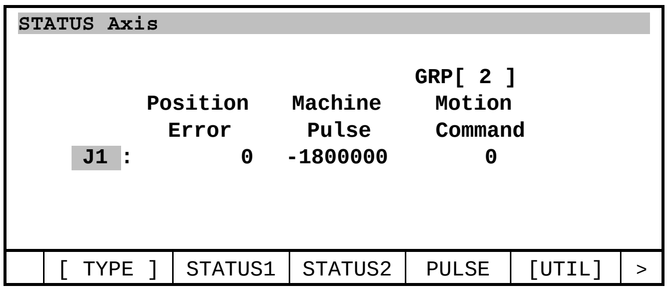
Step

1. Set torque limit 100%
2. Move to grip position without the work.



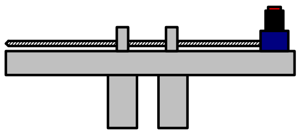
Grip position

3. Select [MENU] → 0.--Next-- → 4 Status→ F4[PULSE].
4. Select F5[UTIL] → GROUP. Specify group 2.



```
STATUS Axis

                                    GRP[ 2 ]
           Position    Machine     Motion
              Error      Pulse     Command
    J1 :          0   -1800000          0

  [ TYPE ] STATUS1 STATUS2 PULSE [UTIL] >
```

5. Record the machine pulse of group 2, J1 of the grip position displayed on TP.
6. Keep torque limit 100% and move the servo hand to destination position without gripping work.



Destination position

7. Record the machine pulse of destination position after moving.

8. Subtract the machine pulse of the grip position from the destination one. Calculate the absolute value, and multiply it by 1.3 (only as a guide) as a margin. Calculating formula is as follows.
   |(Destination position machine pulse – Grip position machine pulse)| × 1.3

9. Change the following system variables to the value calculated in step 8. Record the original value to be able to turn back the value.
   Select [MENU] → 0.--NEXT-- → 6 SYSTEM → F1[TYPE] → Variables. Set the calculated value to following variables.
   $ PARAM_GROUP[2].$STOPTOL[1]
   $ PARAM_GROUP[2].$STOPERLIM[1]
   $ PARAM_GROUP[2].$MOVER_OFFST[1]

10. Cycle power the robot controller.

NOTE
To enable the changed value above, the robot controller must be cycle power.

Calculation sample

In the case the machine pulse of destination position is -50000 and the grip one is 60000,

| (-50000) - 60000)| ×1.3 = 143000 is set to the system variables described above.

When the alarms, stop excess error and so on, tend to be displayed even though cycle power after setting them, set the larger value to the system variables. For example, multiply 1.3 and cycle power again.

128000 ×1.3 = 166400

If the alarms tend to be displayed, repeat these step to set a little bigger value and check the operation.

10.5 FOR CHECKING THE SERVO HAND CURRENT

Servo hand (independent axis) current is monitored if it reaches 95% of the maximum toque that is limited by torque limit instruction until time out in the Section 10.1 “CHECK THE SERVO HAND CURRENT”

This 95 value is saved in the KAREL variable, TQ_TOLERANCE, of CKSVHCUR

NOTE
Use as 95% except special cases.

NOTE
If CKSVHCUR is deleted and reloaded from system design tool screen once, TQ_TORELANCE is reset to 95.

NOTE
If TQ_TORELANCE is set to 100, it is not detected that the current is reached to specified value by internal software calculation error in some cases. Not over 95 is recommended.

10.6 TO RESTART AFTER POWER OFF WHILE CHECKING

In the case power off while checking the servo hand current, set the cursor to one upper line and execute CKSVHCUR.PC after power on the robot controller.

⚠ **CAUTION**
In some cases additional treatment is needed such as once opening the servo hand and removing the work from it. Please confirm the system and restart the system after additional treatment.

In the sample described above, set the cursor onto line 10 of SERVO_HAND_GRIP.TP.

```
<SERVO_HAND_GRIP.TP>
  10:   !Monitor the current ;           ← Move the cursor to this line and restart the TP program.
  11:   CALL CKSVHCUR(2,1,2,3) ;
```

11 APPENDIX

11.1 ERROR CODES

Associated with System Design Tool is copied from B-83284EN-1 to below.

11.1.1 SYST Alarm Code

SYST-298 Trigonometric Function not installed

Cause: No Trigonometric Function (J736) option.
Remedy: This message is reference only. No action is required.

SYST-299 Robot Setup Assistance not installed

Cause: No Robot Setup Assistance (J737) option.
Remedy: This message is reference only. No action is required.

SYST-300 System Design Tool not installed

Cause: No System Design Tool (J738) option.
Remedy: This message is reference only. No action is required.

11.1.2 KALM Alarm Code

NOTE
KALM is for User Specified Alarm Post included System Design Tool. This alarm code is for USER ONLY. If FANUC software detects some abnormal status in the robot controller, KALM alarm code is never displayed. In case that KALM is displayed, verify your robot system, and exercise proper care.

KALM-001 ~ KALM-200 are for global alarm.
KALM-201 ~ KALM-300 are for local alarm.

Refer to 8.1 "USER SPECIFIED ALARM POST" to know KALM.

11.1.3 SDTL Alarm Code

NOTE
SDTL is for System Design Tool (J738), Robot Setup Assistance (J737) and Trigonometric Function (J736).

SDTL-001 [STSTEPON] Single Step ON

Cause: Switch Single Step KAREL program (STSTEPON.PC) was executed, then step mode turns ON.
Remedy: This message is reference only. No action is required.

SDTL-002 [STSTEPOF] Single Step OFF

Cause: Switch Single Step KAREL program (STSTEPOF.PC) was executed, then step mode turns OFF.
Remedy: This message is reference only. No action is required.

SDTL-003 No English dictionary

Cause: Set English KAREL program (STLANGEG.PC) was executed, however, it could not set English dictionary.
Remedy: Add the order of English dictionary.

SDTL-004 No Japanese dictionary

Cause: Set Japanese KAREL program (STLANGJP.PC) was executed, however, it could not set Japanese dictionary.
Remedy: **A**dd the order of Japanese dictionary.

SDTL-005 Changed to English

Cause: Set English KAREL program (STLANGEG.PC) was successfully executed. Dictionary switched to English.
Remedy: This message is reference only. No action is required.

SDTL-006 Changed to Japanese

Cause: Set Japanese KAREL program (STLANGJP.PC) was successfully executed. Dictionary switched to Japanese.
Remedy: This message is reference only. No action is required.

SDTL-007 Invalid file list

Cause: TP Program Memory Assist (EXT_DEL.PC, EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, file list is invalid format.
Remedy: Refer to cause code.

SDTL-008 :¥ not found

Cause: There is no :¥ in the parameter, then external device is not recognized. This is cause code of SDTL-007.
Remedy: Insert :¥ after external device like MC:¥.

SDTL-009 Too long name

Cause: The parameter is too long. This is cause code of SDTL-007.
Remedy: Set string parameter length to under 24.

SDTL-010 No name

Cause: Specified parameter is NULL. This is cause code of SDTL-007.
Remedy: Input the parameter correctly.

SDTL-011 TPE parameter %s wrong data type

Cause: TP Program Memory Assist (EXT_DEL.PC, EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, the type of specified parameter was wrong.
Remedy: Refer to cause code.

SDTL-012 Not STRING

Cause: The parameter type is not string. This is cause code of SDTL-011.
Remedy: Input string to the parameter.

SDTL-013 Not INTEGER

Cause: The parameter type is not integer. This is cause code of SDTL-011.
Remedy: Input integer to the parameter.

SDTL-014 Not REAL

Cause: The parameter type is not real. This is cause code of SDTL-011.
Remedy: Input real to the parameter.

SDTL-015 %s not found

Cause: TP Program Memory Assist (EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, specified TP program in the file list was not found.
Remedy: Confirm if TP program exist or not.

SDTL-016 Failed to save %s

Cause: TP Program Memory Assist (EXT_SAVE.PC) was executed, however, specified TP program in the file list was not saved.
Remedy: Refer to cause code.

SDTL-017 Failed to load %s

Cause: TP Program Memory Assist (EXT_LOAD.PC) was executed, however, specified TP program in the file list was not loaded.
Remedy: Refer to cause code.

SDTL-018 Failed to delete %s

Cause: TP Program Memory Assist (EXT_DEL.PC) was executed, however, specified TP program in the file list was not deleted.
Remedy: Refer to cause code.

SDTL-019 %s not found

Cause: TP Program Memory Assist (EXT_DEL.PC, EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, specified file list was not found.
Remedy: Refer to cause code.

SDTL-020 File reading error

Cause: TP Program Memory Assist (EXT_DEL.PC, EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, specified file list could not read.
Remedy: Refer to cause code.

SDTL-021 TPE parameter not exist

Cause: TP Program Memory Assist (EXT_DEL.PC, EXT_SAVE.PC, EXT_LOAD.PC) was executed, however, specified parameter could not be got
Remedy: Confirm whether the parameter is correct.

SDTL-022 Failed to load %s

Cause: Failed to load specified program.
Remedy: Refer to cause code and remove cause of it.

SDTL-023 Specified program is in use

Cause: The program you specified for load or delete is opened for another function.
Remedy: Abort the program. If teach pendant displays 2 or more screens and one of them displays the program, display another screen.

SDTL-024 Paused

Cause: The program you specified for load or delete is paused.
Remedy: Abort the program.

SDTL-025 %d

Cause: The program you specified for load or delete in system design tool screen cannot be loaded or deleted. It was not because the program was used by another function. It was not because the program was paused. This alarm is displayed in this situation.
Remedy: Abort the program. If teach pendant displays 2 or more screens and one of them displays the program, display another screen.

SDTL-026 Failed to delete %s

Cause: Specified program cannot be deleted.
Remedy: Refer to cause code and remove cause of it.

SDTL-028 Variable used by other program

Cause: Variable of the program you specified for load or delete is used by another program.
Remedy: Delete the programs which refer to the variable.

SDTL-029 Failed to set to register

Cause: Check Real Position (GTCURCUR.PC), Get Max Current (GTMAXCUR.PC), Get Real Position (GTRELPOS.PC) or Set Current Position to Register (STPOSREG.PC) could not set value to register specified by argument.
Remedy: Refer to cause code and remove cause of it.

SDTL-030 Failed to set to position register

Cause: Set Current Position to String Register (STPOSPRG.PC) could not set value to position register.
Remedy: Refer to cause code and remove cause of it.

SDTL-031 Failed to get the robot position

Cause: Get Real Position (GTRELPOS.PC) could not get current position of the robot.
Remedy: Refer to cause code and remove cause of it. If cause code is not displayed, gear ratio of extended axis might be wrong.

SDTL-032 Failed to get max current

Cause: Get Max Current (GTMAXCUR.PC) cannot get the maximum current.
Remedy: Refer to cause code and remove cause of it.

SDTL-033 Failed converting to REAL value

Cause: Set Current Position to Register (STPOSREG.PC) could not perform conversion to real value.
Remedy: Refer to cause code and remove cause of it.

SDTL-034 Failed to get TPE parameter %s

Cause: Check Real Position (GTCURCUR.PC), Get Max Current (GTMAXCUR.PC), Get Real Position (GTRELPOS.PC), Set Current Position to Register (STPOSREG.PC), Set Current Position to String Register (STPOSPRG.PC), COPY ERROR FILE (ERFILECP.PC), WRITE PROTECTION ON (STWRPRT.PC) or WRITE PROTECTION OFF (CRWRPRT.PC) could not get parameter.
Remedy: Refer to cause code and remove cause of it.

SDTL-035 TPE parameter %s wrong data type

Cause: Check Real Position (GTCURCUR.PC), Get Max Current (GTMAXCUR.PC), Get Real Position (GTRELPOS.PC), Set Current Position to Register (STPOSREG.PC) or Set Current Position to String Register (STPOSPRG.PC) could not get parameter.
Remedy: Refer to cause code and remove cause of it.

SDTL-036 Not STRING

Cause: Specified argument is not string. This alarm is cause code of SDTL-035.
Remedy: Input string as argument.

SDTL-037 Not INTEGER

Cause: Specified argument is not integer. This alarm is cause code of SDTL-035.
Remedy: Input integer as argument.

SDTL-038 Not REAL

Cause: Specified argument is not real number. This alarm is cause code of SDTL-035.
Remedy: Input real number as argument.

SDTL-039 TPE parameter %s is out of range

Cause: Specified parameter of Check Real Position (GTCURCUR.PC), Get Max Current (GTMAXCUR.PC), Get Real Position (GTRELPOS.PC), Set Current Position to Register (STPOSREG.PC) and Set Current Position to String Register (STPOSPRG.PC) was out of range.
Remedy: Input correct parameter.

SDTL-040 TPE parameter %s wrong data type

Cause: Specified parameter of Get Number of Collision Detect (GTCDCNT.PC) is wrong data type.
Remedy: Input correct parameter.

SDTL-041 Not INTEGER

Cause: Specified argument is not integer. This alarm is cause code of SDTL-040.
Remedy: Input integer to the parameter.

SDTL-042 Illegal group number

Cause: Illegal parameter was specified to the first parameter of Get Number of Collision Detect (GTCDCNT.PC) as the group.
Remedy: Input correct group number to the first parameter.

SDTL-043 Illegal axis number

Cause: Illegal parameter was specified as the second parameter of Get Number of Collision Detect (GTCDCNT.PC) as the axis.

Remedy: Input correct axis number to the second parameter.

SDTL-044 String size not big enough

Cause: Too big size of string is going to set in Set String to System Variable (STSYSSTR.PC). A part of string of second parameter is set to the system variable specified as first parameter.

Remedy: The capacity length to set string is vary with system variables. Set the shorter string.

SDTL-045 Variable type is not compatible

Cause: The type of system variable is different from the parameter to be set in Set String to System Variable (STSYSSTR.PC).

Remedy: Set the correct type of string.

SDTL-046 Illegal variable

Cause: Incorrect system variable is set as the first parameter in Set String to System Variable (STSYSSTR.PC).

Remedy: Set the correct system variable. Verify that miss spelling, inputting space, nonexistence variable is set and so on.

SDTL-047 TPE parameter %s wrong data type

Cause: Specified parameter of Set String to System Variable (STSYSSTR.PC) is wrong data type.

Remedy: Set correct string or string register index to the parameter.

SDTL-048 Wrong data type(Line %s)

Cause: Execution of X_DYNDAT causes this error. Wrong data type (identifier) is specified in CSV file.

Remedy: Displayed line of specified file has problem. String at which X_DYNDAT detected error is displayed, too. Even if the string is valid identifier for X_DTCOPY, it may not be supported by X_DYNDAT. Actually most of valid identifier of X_DTCOPY are not supported by X_DYNDAT. Please specify identifier that is supported by X_DYNDAT.

SDTL-049 Illegal terminator(Line %s)

Cause: Execution of X_DYNDAT causes this error. Terminator(!!!!) is expected at the end of a line of CSV file but it was not found.

Remedy: Check displayed line of CSV file. Confirm data format and add terminator according to data format.

SDTL-050 Illegal index(Line %s)

Cause: Execution of X_DYNDAT causes this error. Wrong index is specified in CSV file.

Remedy: Check displayed line of CSV file. String at which X_DYNDAT detected error is displayed, too. Check the string, data type and its data format. If index is out of range, specify proper value.

SDTL-051 Illegal group(Line %s)

Cause: Execution of X_DYNDAT causes this error. Wrong formatted data is found where group number should be specified.

Remedy: Check displayed line of CSV file. String at which X_DYNDAT detected error is displayed, too. Check the string, data type and its data format.

SDTL-052 Illegal data(Line %s)

Cause: Execution of X_DYNDAT causes this error. Unexpected data is found at displayed line of specified CSV file.

Remedy: Check displayed line of CSV file. Check data type and its data format.

SDTL-053 Illegal config(Line %s)

Cause: Execution of X_DYNDAT causes this error. Position register of Cartesian data type needs configuration data. Wrong data is specified in CSV file.

Remedy: Check displayed line of CSV file. String at which X_DYNDAT detected error is displayed, too. Specify configuration data properly.

SDTL-054 Out of range(Line %s)

Cause: Execution of X_DYNDAT causes this error. Integer value in CSV file was out of range.

Remedy: Check displayed line of CSV file. Wrong value, minimum value and maximum value are displayed, too. Write proper data in CSV file.

SDTL-055 Empty field does not exist(Line %s)

Cause: Execution of X_DYNDAT causes this error. CSV file doesn't have empty field (field that doesn't have character) where it is expected. If data for an identifier is composed of 2 or more lines, the 2nd or later lines must start with empty field.

Remedy: Check displayed line of CSV file. String at which X_DYNDAT detected error is displayed, too. Check data type and its data format. Input data in proper format.

SDTL-056 Converted to real of out of range(Line %s)

Cause: Execution of X_DYNDAT causes this error. Very large (or small) value is specified as real value of register. It was interpreted as value of out of range. Data should be larger than -1.0E16 and less than 1.0E16. Scientific notation cannot be used in CSV file.

Remedy: Make absolute value smaller.

SDTL-057 Index of SR is wrong(%s)

Cause: Execution of X_DYNDAT without parameter causes this error. In this case, path of CSV file is decided by string register. Index of string register is decided by KAREL variable, [X_DYNDAT]IDEFSRIDX (KAREL variable IDEFSRIDX of KAREL program X_DYNDAT). String register of specified index doesn’t exist.

Remedy: Check value of [X_DYNDAT]IDEFSRIDX. Change its value to valid index of string register.

SDTL-058 Cannot open specified file(%s)

Cause: Execution of X_DYNDAT causes this error. %s shows specified path of CSV file. If X_DYNDAT is used without parameter, string register used to decide file path is displayed, too. Following table shows examples. In all cases, specified file could not be opened.

Example of error	Description
SDTL-058 Cannot open specified file (MC:¥CP_0100.CSV)	X_DYNDAT was executed by CALL instruction. The 1st parameter corresponded to ‘MC:¥CP_0100.CSV’.
SDTL-058 Cannot open specified file ()	X_DYNDAT was executed by CALL instruction. The 1st parameter corresponded to ‘’(null string). If CALL X_DYNDAT(SR[3]) is executed and SR[3] is ‘’ (null string), this can happen.
SDTL-058 Cannot open specified file (SR[1]:MC:¥CP_0100.CSV)	X_DYNDAT was directly RUN or called without parameter. Because KAREL variable IDEFSRIDX of X_DYNDAT was 1, value of SR[1] was used as file path of CSV file. SR[1] was ‘MC:¥CP_0100.CSV’.
SDTL-058 Cannot open specified file (SR[1]:)	This is similar to the case above, but SR[1] was ‘’ (null string).

Please refer to cause code, too. For example, if error is displayed like following, please check if specified file exists or not.

SDTL-058 Cannot open specified file (MC:¥CP_0100.CSV)
FILE-014 File not found.

Remedy: Check displayed file path. Check if specified file exists. Please refer to cause code, too.

SDTL-060 :¥ not found (%s)

Cause: String of file path doesn’t include :¥. %s shows specified file path.

Remedy: Include :¥ in file path string.

SDTL-061 Illegal file path (%s)

Cause: Specified file path is not proper. %s shows specified file path.

Remedy: Correct file string path.

SDTL-062 Last char is not ¥ (%s)

Cause: The last character of file path string is not ¥. %s shows specified file path.
Remedy: Correct file path string.

SDTL-063 Failed to copy (%s)

Cause: File copy failed. %s shows destination file name.
Remedy: Check whether specified file device and sub directory can be used. Please refer to cause code, too.

SDTL-064 TP Program does not exist (%s)

Cause: Specified TP program does not exist on the controller. %s shows specified TP program name.
Remedy: Please specify program that exists on controller.

SDTL-065 Program name is not specified %s

Cause: Program name was not specified. If string register is specified, information of the string register is displayed. SR[1] for example.
Remedy: Check parameter. If integer value is used, value of SR[the integer] is used as program name.

SDTL-066 * cannot be used (%s)

Cause: Program name for STWRPRT and CRWRPRT contains "*". You cannot use the character to specify program name.
Remedy: Correct program name.

SDTL-067 Failed to turn write protect on(%s)

Cause: Write protection could not be turned on. %s shows program name. If string register is specified, information of the string register is also displayed.
Remedy: Displayed program is used. Try again in a state that displayed program is not used. Please refer to cause code, too.

SDTL-068 Index of SR is wrong(%s)

Cause: Index of string register is wrong. %s shows specified index is displayed.
Remedy: Specify string register that exist on the controller. KAREL variable can be used to specify index. If it is the case, KAREL variable should be modified to proper value. For example, when STWRPRT or CRWRPRT was called without parameter, IDEFSRIDX of each program is used. The value should be checked.

SDTL-069 Failed to turn write protect off(%s)

Cause: Write protection could not be turned off. %s shows program name. If string register is specified, information of the string register is also displayed.
Remedy: Displayed program is used. Try again in a state that displayed program is not used. Please refer to cause code, too.

SDTL-070 Specified program is not TP(%s)

Cause: Specified program is not TP program.
Remedy: Specify TP program.

SDTL-071 Torque Limit Function not installed

Cause: Torque Limit Function Option (J611) is not ordered. It is necessary to execute Check Servo Hand Current (CKSVHCUR.PC).
Remedy: Contact your FANUC Technical Representative to add Torque Limit Function Option (J611).

SDTL-072 Torque Limit disabled

Cause: Torque limit function of specified group or axis is disabled.
Remedy: See the Chapter, “Torque Limit Function” of R-30*i*B/R-30*i*B Mate CONTROLLER Optional Function OPERATOR’S MANUAL and the Subsection 10.3.1.1 ” How to Set up Torque Limit” of R-30*i*B/R-30*i*B Mate System Design Tool OPERATOR’S MANUAL (B-83274EN). Enable the torque limit of specified group and axis.

SDTL-073 Group %s not mastered

Cause: Specified group is not mastered in Get Servo Hand Real Position (GTRELPOS.PC).
Remedy: Perform the mastering from the calibration screen [6 SYSTEM Master/Cal].

SDTL-074 Illegal group (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. The 1st argument is group number. Specified group doesn't exist. %s shows specified group number.

［Remedy］ Check the 1st argument of STPAYLOD. Specify motion group that exists on the controller.

SDTL-075 Illegal Payload schedule number (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. The 2nd argument is payload schedule number. Specified payload schedule number doesn't exist. %s shows specified schedule number.

［Remedy］ Check the 2nd argument of STPAYLOD. Specify payload schedule number that exists on the controller.

SDTL-076 Payload is out of range (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Payload that is set in specified register is out of range. Index of register that is used to specify payload is displayed. Its value, which should be payload, is displayed, too. Suppose R [5] is specified and its value is -10.5. SDTL-076 is displayed as “Payload is out of range (5, -10.500)”.

［Remedy］ Check index of register and value of the register. Set proper payload.

SDTL-077 Payload center position is out of range (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Payload center is out of range. It is specified by register. %s shows followings.

- X, Y or Z
- Index of register that is used to specify payload center.
- Specified value by the register.

Suppose the 3rd argument of STPAYLOD is 101. Center of gravity of payload is specified by R [102]. If its value is -20.5, SDTL-077 is displayed as “Payload center position is out of range (X, 105, -20.500)”

［Remedy］ Check index of register and value of the register. Set proper value.

SDTL-078 Payload inertia is out of range (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Inertia of payload is out of range. It is specified by register. %s shows followings.

- X, Y or Z
- Index of register that is used to specify inertia.
- Specified value by the register.

Suppose the 3rd argument of STPAYLOD is 101. Inertia of payload about X axis is specified by R [105]. If its value is -20.5, SDTL-078 is displayed as “Payload inertia is out of range (X, 105, -20.500)”

［Remedy］ Check index of register and value of the register. Set proper value.

SDTL-079 TPE parameter %s wrong data type

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Data type of shown parameter is wrong.

［Remedy］ Use argument of proper data type.

SDTL-080 Failed to get TPE parameter %s

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. The Nth TPE parameter does not exist. N is shown in %s. It is 1, 2 or 3.

［Remedy］ STPAYLOD needs 3 argument. Teach required argument.

SDTL-081 Register for payload does not exist (%s)

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Register for payload does not exist.

［Remedy］ The 3rd argument of STPAYLOD is index of register for payload. Set proper value.

SDTL-082 Register for payload center (%s) does not exist.

［Cause］ This alarm is displayed by execution of KAREL program, STPAYLOD. Register for payload center does not exist. %s shows coordinate (X or Y or Z) and index of the register.

Suppose the 3rd argument of STPAYLOD is 200. Center of gravity of payload is specified by R [201]. If R[201] doesn’t exist, “SDTL-082 Register for payload center (X, 201) does not exist “ can be displayed.

［Remedy］ Set proper value to the 3rd argument so that required registers exist.

Suppose the 3rd argument of STPAYLOD is n. Center of gravity of payload is specified by following registers.

R [n+1]: Position of center of gravity, X [cm]
R [n+2]: Position of center of gravity, Y [cm]

R [n+3]: Position of center of gravity, Z [cm]

SDTL-083 Register for payload inertia (%s) does not exist.

〔Cause〕 This alarm is displayed by execution of KAREL program, STPAYLOD. Register for payload inertia does not exist. %s shows coordinate (X or Y or Z) and register number.
Suppose the 3rd argument of STPAYLOD is 197. Inertia about X axis is specified by R [201]. If R [201] doesn't exist, "SDTL-083 Register for payload inertia (X, 201) does not exist" can be displayed.

〔Remedy〕 Set proper value to the 3rd argument so that registers for inertia exist.
Suppose the 3rd argument of STPAYLOD is n. Inertia of payload is specified by following registers.

R [n+4]: Inertia about X [cm]
R [n+5]: Inertia about Y [cm]
R [n+6]: Inertia about Z [cm]

11.1.4 Added History of System Design Tool

No.	KAREL	Description	Series and Version			Option		
			R-30*i*B	R-30*i*B Mate	R-30*i*B/ R-30*i*B Mate			
			7DC1	7DD0	7DC2	J736	J737	J738
1	PSIN	SIN function	1	1	1	○		○
2	PCOS	COS function	1	1	1	○		○
3	PTAN	TAN function	1	1	1	○		○
4	PASIN	arc SIN function	1	1	1	○		○
5	PACOS	arc COS function	1	1	1	○		○
6	PATAN	arc TAN function	1	1	1	○		○
7	PABS	Absolute value	1	1	1	○		○
8	PEXP	Exponential	1	1	1	○		○
9	PLN	logarithm	1	1	1	○		○
10	PROUND	Truncation	1	1	1	○		○
11	PTRUNC	Round off	1	1	1	○		○
12	PSQRT	Square root	1	1	1	○		○
13	X_DTCOPY	Robot setup assistance, main program	1	1	1		○	○
14	CSV_UTIL	Robot setup assistance, sub program. CSV utility	1	1	1		○	○
15	X_CPBKUP	Robot setup assistance, sub program. Automatic backup	1	1	1		○	○
16	X_CPBLAL	Robot setup assistance, sub program. Battery alarm	1	1	1		○	○
17	X_CPBLNK	Robot setup assistance, sub program. Back light automatic blanking	1	1	1		○	○
18	X_CPDSET	Robot setup assistance, sub program	1	1	1		○	○
19	X_CPERNA	Robot setup assistance, sub program. Alarms that do not turn on FAULT signal	1	1	1		○	○
20	X_CPERTB	Robot setup assistance, sub program. Error table	1	1	1		○	○
21	X_CPFRM	Robot setup assistance, sub program. Frame setup	1	1	1		○	○
22	X_CPIO	Robot setup assistance, sub program. I/O	1	1	1		○	○
23	X_CPLD	Robot setup assistance, sub program. Payload	1	1	1		○	○
24	X_CPMCR	Robot setup assistance, sub program. Macro setup	1	1	1		○	○
25	X_CPNOHI	Robot setup assistance, sub program. Warn/Reset in alarm history	1	1	1		○	○
26	X_CPOVSL	Robot setup assistance, sub program. Override select	1	1	1		○	○
27	X_CPPREG	Robot setup assistance, sub program. Position register	1	1	1		○	○
28	X_CPPROG	Robot setup assistance, sub program. Program comment	1	1	1		○	○
29	X_CPPSEL	Robot setup assistance, sub program. Program select	1	1	1		○	○
30	X_CPPTIM	Robot setup assistance, sub program. Program timer	1	1	1		○	○
31	X_CPREF	Robot setup assistance, sub program. Reference position	1	1	1		○	○

No.	KAREL	Description	Series and Version			Option		
			R-30*i*B	R-30*i*B Mate	R-30*i*B/ R-30*i*B Mate			
			7DC1	7DD0	7DC2	J736	J737	J738
32	X_CPREG	Robot setup assistance, sub program. Register	1	1	1		○	○
33	X_CPSREG	Robot setup assistance, sub program. String register	1	1	1		○	○
34	X_CPSSP	Robot setup assistance, sub program. Space check function	1	1	1		○	○
35	X_CPSYSC	Robot setup assistance, sub program. System configuration	1	1	1		○	○
36	X_CPTCPI	Robot setup assistance, sub program. TCP/IP	1	1	1		○	○
37	X_CPUALM	Robot setup assistance, sub program. User alarm	1	1	1		○	○
38	X_DYNDAT	Robot setup assistance. Data set from CSV to position register.	6	1	1		○	○
39	XCSVUTIL	Robot setup assistance, sub program. CSV utility	15	7	1		○	○
40	EXT_LOAD	Load TP program from external device (MC:)	1	1	1			○
41	EXT_SAVE	Save TP program to external device (MC:)	1	1	1			○
42	EXT_DEL	Delete TP program	1	1	1			○
43	CKDIOSIM	Check DI/O simulated status	1	1	1			○
44	CKAIOSIM	Check AI/O simulated status	1	1	1			○
45	CKGIOSIM	Check GI/O simulated status	1	1	1			○
46	CKRIOSIM	Check RI/O simulated status	1	1	1			○
47	CKWIOSIM	Check WI/O simulated status	1	1	1			○
48	CRDIOSIM	Unsimulate DI/O	1	1	1			○
49	CRAIOSIM	Unsimulate AI/O	1	1	1			○
50	CRGIOSIM	Unsimulate GI/O	1	1	1			○
51	CRRIOSIM	Unsimulate RI/O	1	1	1			○
52	CRWIOSIM	Unsimulate WI/O	1	1	1			○
53	CRIOSIM	Unsimulate all I/O	1	1	1			○
54	GTWRTPRT	Get write protection attribute.	1	1	1			○
55	CKDRYRUN	Check dry run	1	1	1			○
56	CKGRPMTN	Check group motion (machine lock)	1	1	1			○
57	CKSGLSTP	Check single step	1	1	1			○
58	CKOFFSET	Check ignore offset command	1	1	1			○
59	CKTLOFST	Check ignore tool offset	1	1	1			○
60	CKVOFFST	Check enable voffset	1	1	1			○
61	CKOVRIDE	Check override	1	1	1			○
62	CKUOPDIS	Check enable UI signals	1	1	1			○
63	GTMODESW	Get mode switch	1	1	1			○
64	STSTEPON	Single step ON	1	1	1			○
65	STSTEPOF	Single step OFF	1	1	1			○
66	STPOSREG	Set current position to register	1	1	1			○
67	STPOSPRG	Set current position to position register	1	1	1			○
68	GTAXSCUR	Get current of the robot axis	1	1	1			○
69	GTDTBTRQ	Get disturbance torque of the robot axis	1	1	1			○
70	GTTRQAVE	Get average torque of the robot axis	1	1	1			○
71	GTTRQMAX	Get maximum torque of the robot axis	1	1	1			○

No.	KAREL	Description	Series and Version			Option		
			R-30*i*B	R-30*i*B Mate	R-30*i*B/ R-30*i*B Mate			
			7DC1	7DD0	7DC2	J736	J737	J738
72	GTMAXCUR	Get max current of the robot axis	1	1	1			○
73	GTCURCUR	Check current of the robot axis(*1)	1	1	1			○
74	STSYSSTR	Set string to system variable	2	1	1			○
75	GTCDCNT	Get number of collision detect	2	1	1			○
76	STRBERGL	Global alarm post	1	1	1			○
77	STRBERLO	Local alarm post	1	1	1			○
78	GTRBDATE	Get date	1	1	1			○
79	GTRBTIME	Get time	1	1	1			○
80	STLANGEG	Set English	1	1	1			○
81	STLANGJP	Set Japanese	1	1	1			○
82	REDISP04	Refresh screen of teach pendant	1	1	1			○
83	STPRGNAM	Set program	1	1	1			○
84	CKSVHCUR	Check the servo hand current	12	6	1			○
85	STPAYLOD	Set payload setting	12	7	1			○
86	GTRELPOS	Get servo hand real position	1	1	1			○
87	CRPOSREG	Clear of position register.	6	1	1			○
88	ERFILECP	Copy of error file.	7	1	1			○
89	STWRPRT	Turn on write protect.	7	1	1			○
90	CRWRPRT	Turn off write protect.	7	1	1			○

No	Description
1	Use CKSVHCUR in the case of the servo hand.

11.1.5 Changed History of System Design Tool

KAREL	History	Series and version		
		R-30*i*B	R-30*i*B Mate	R-30*i*B R-30*i*B Mate
		7DC1	7DD0	7DC2
GTRELPOS	“STDL-073 Group i not mastered” (i: group number) is displayed if the robot is not mastered.	12	6	1

INDEX

REVISION RECORD

Edition	Date	Contents
02	Oct., 2013	• Applied to R-30*i*B Mate • Update Chapter 3, “TRIGONOMETRIC FUNCTION”, Chapter 4, “ROBOT SETUP ASSISTANCE TOOL”, Chapter 7, “STATUS CHECK/SET FUNCTION”, Chapter 8 “OTHERS”. • Add Chapter 10,”CHECK SERVO HAND GRIP STATUS”. • Add subsection to APPENDIX. 11.1.4 “Added History of System Design Tool”, 11.1.5 “Changed History of System Design Tool”.
01	June, 2012	

B-83274EN/02

* B - 8 3 2 7 4 E N / 0 2 *