

FANUC Robot **series**

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

***i*Pendant customization**

OPERATOR'S MANUAL

B-83594EN/01

- **Original Instructions**

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

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

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

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

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

In this manual we have tried as much as possible to describe all the various matters.

However, we cannot describe all the matters which must not be done, or which cannot be done, because there are so many possibilities.

Therefore, matters which are not especially described as possible in this manual should be regarded as "impossible".

SAFETY PRECAUTIONS

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

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

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

In addition, refer to the "FANUC Robot SAFETY HANDBOOK (B-80687EN)".

1 DEFINITION OF USER

The user can be defined as follows.

Operator:

- Turns ON/OFF power to the robot
- Starts the robot program from the operator's panel

Programmer:

- Operates the robot
- Teaches the robot inside the safety fence

Maintenance engineer:

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



- Operator is not allowed to work in the safety fence.
- Programmers and maintenance engineers are allowed to work in the safety fence. The work inside the safety fence includes lifting, setting, teaching, adjustment, maintenance, etc.
- To work inside the safety fence, the person must receive a professional training for the robot.

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

- Adequate clothes for the operation
- Safety shoes
- A helmet

2 DEFINITION OF SAFETY NOTATIONS

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

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

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

3 SAFETY OF THE USER

User 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 user safety.

- (1) Have the robot system users 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 user 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 user 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 (b).

- (4) Provide the peripheral equipment with appropriate earth (Class A, Class B, Class C, and Class D).
- (5) Try to install the peripheral equipment outside the robot operating space.
- (6) Draw an outline on the floor, clearly indicating the range of the robot operating space, 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 user enters the work area.
- (8) If necessary, install a safety lock so that no one except the user 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 equipment independently, be sure to turn off the power of the robot.
- (10) Operators should be ungloved while manipulating the operator 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. (refer to Controller OPERATOR'S MANUAL.)
- (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 inside 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.)
 - Outdoor
- (16) When connecting the peripheral equipment 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 footstep, please consider security for installation and maintenance work in high place according to Fig. 3 (c). Please consider footstep and safety belt mounting position.

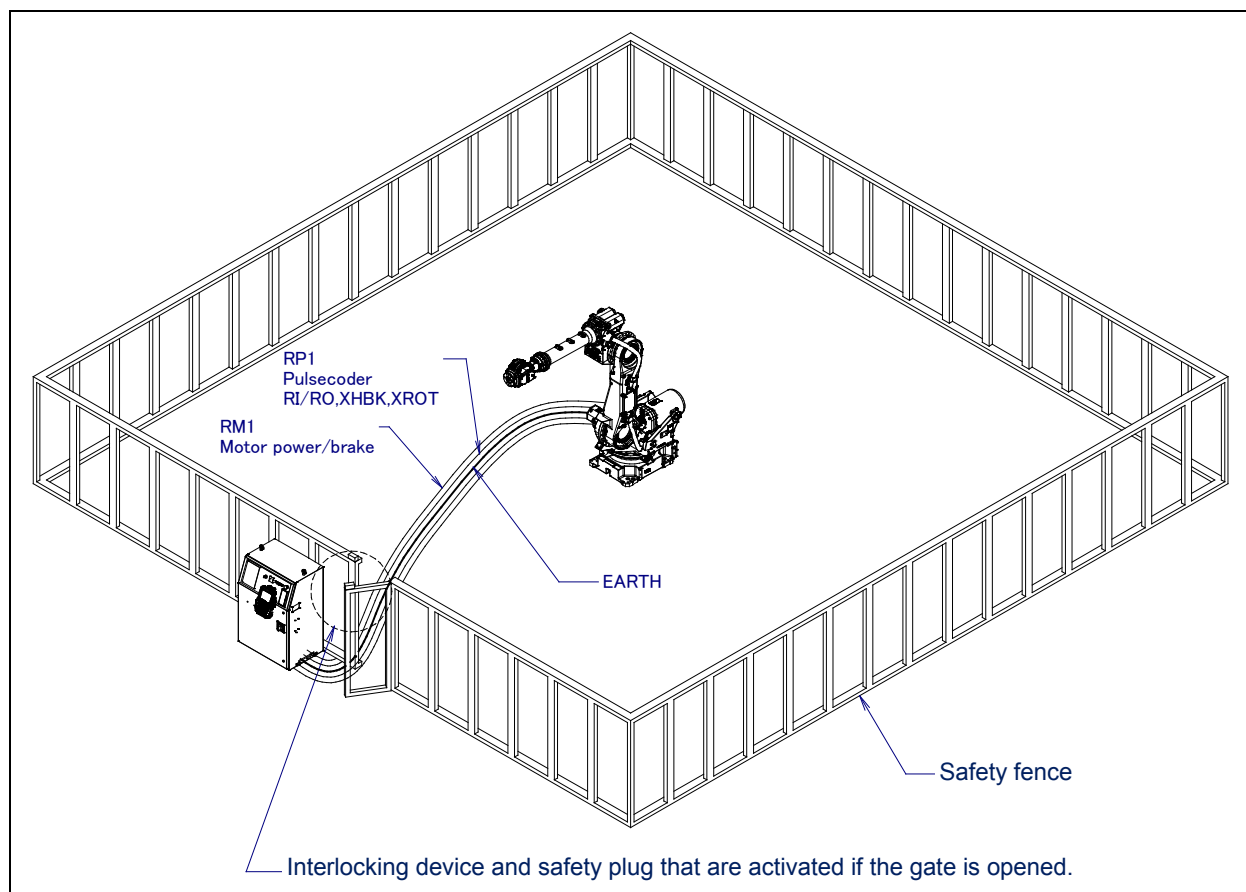


Fig. 3 (a) Safety fence and safety gate

⚠ WARNING

When you close a fence, please confirm that there is not a person from all directions of the robot.

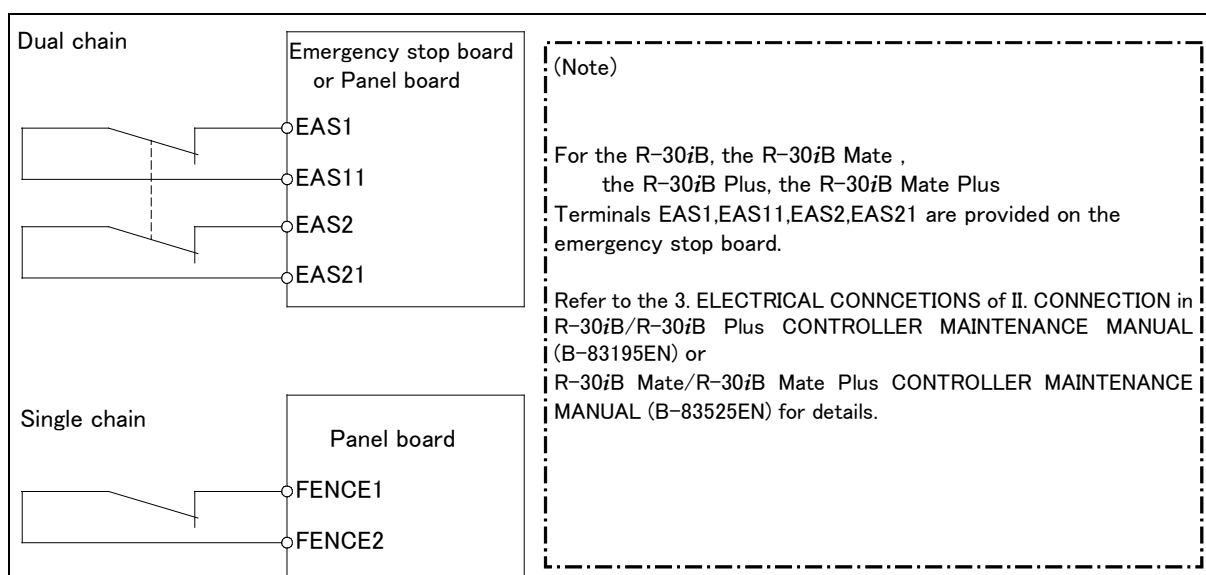


Fig. 3 (b) Connection diagram for the signal of safety fence

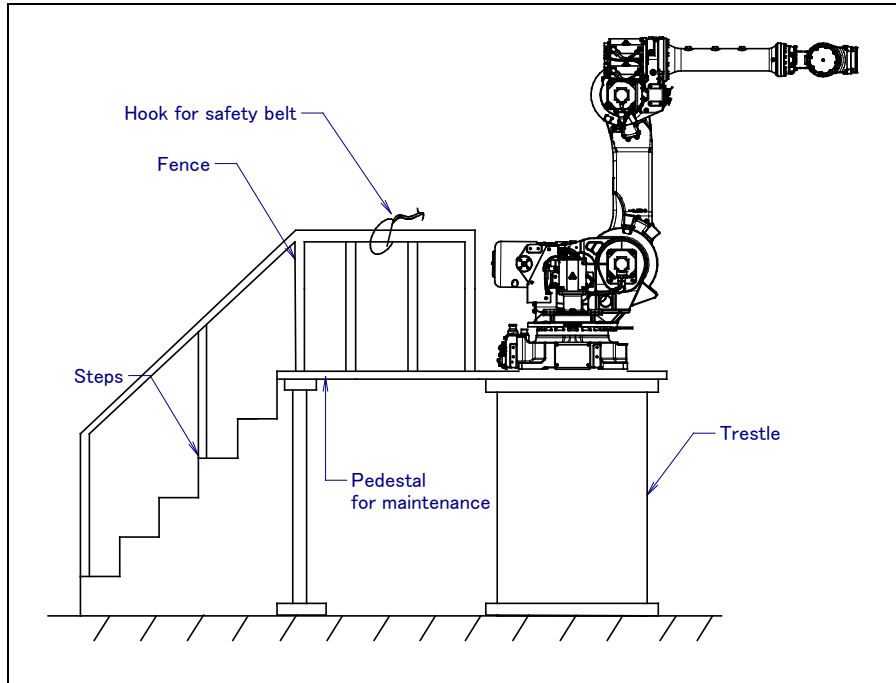


Fig. 3 (c) Pedestal for maintenance

3.1 SAFETY OF THE OPERATOR

An operator refers to a person who turns on and off the robot system and starts a robot program from, for example, the operator panel during daily operation.

Operators cannot work inside of the safety fence.

- (1) If the robot does not need to be operated, turn off the robot controller power or press the EMERGENCY STOP button during working.
- (2) Operate the robot system outside the operating space of the robot.
- (3) Install a safety fence or safety door to avoid the accidental entry of a person other than an operator in charge or keep operator out from the hazardous place.
- (4) Install one or more necessary quantity of EMERGENCY STOP button(s) within the operator's reach in appropriate location(s) based on the system layout.

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.

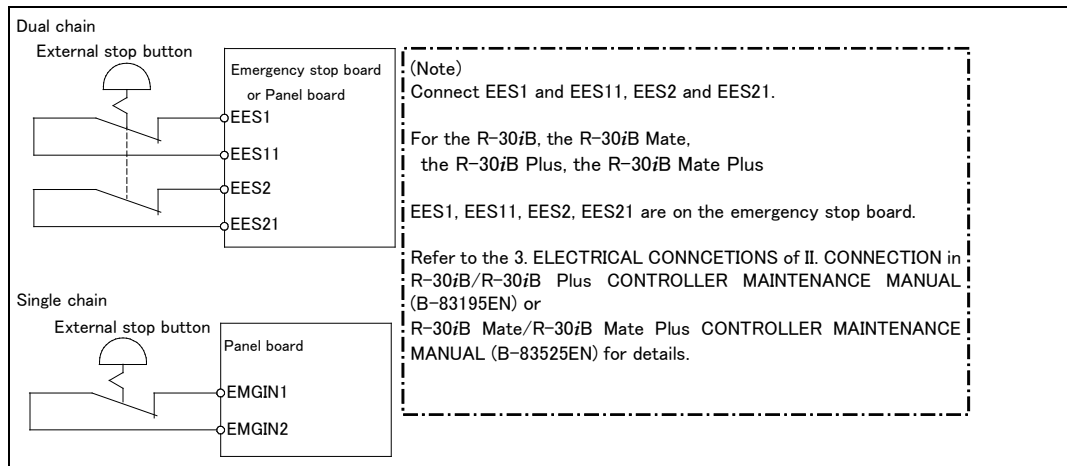


Fig. 3.1 Connection diagram for external emergency stop button

3.2 SAFETY OF THE PROGRAMMER

While teaching the robot, the operator may need to enter the robot operation area. The programmer must ensure the safety especially.

- (1) Unless it is specifically necessary to enter the robot operating space, carry out all tasks outside the operating space.
- (2) Before teaching the robot, check that the robot and its peripheral equipment are all in the normal operating condition.
- (3) If it is inevitable to enter the robot operating space 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 operating space.
- (5) Programming should be done outside the area of the safety fence as far as possible. If programming needs to be done inside 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 whole robot 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 (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 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 the stop of the robot (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) Enable: Servo power is turned off when the operator releases the DEADMAN switch or when the operator presses the switch strongly.
 - (b) Disable: The DEADMAN switch is disabled.
- (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-30iB/R-30iB Mate/R-30iB Plus/R-30iB Mate Plus 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 equipment 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.

Mode	Teach pendant enable switch	Software remote condition	Teach pendant	Operator panel	Peripheral equipment
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 box or operator panel, make certain that nobody is the robot operating space area and that there are no abnormalities in the robot operating space.
- (7) When a program is completed, be sure to carry out a test operation according to the following procedure.
 - (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 continuous operation at low speed.
 - (c) Run the program for one operation cycle in continuous operation 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 continuous operation 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.
- (8) While operating the system in the automatic operation, the programmer should leave the safety fence.

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 operating space.
- (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 as far as possible.
- (3) If it becomes necessary to enter the robot operating space while the power is on, press the emergency stop button on the operator box or operator panel, or the teach pendant before entering the range. The maintenance worker 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 worker must check the whole robot 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 whole robot 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 maintenance work, check that the robot and its peripheral equipment are all in the normal operating condition.
- (7) Do not operate the robot in the automatic operation while anybody is in the robot operating space.
- (8) When you maintain the robot alongside a wall or instrument, or when multiple users are working nearby, make certain that their escape path is not obstructed.
- (9) When a tool is mounted on the robot, or when any movable device other than the robot is installed, such as belt conveyor, pay careful attention to its motion.
- (10) If necessary, have a user who is familiar with the robot system stand beside the operator panel and observe the work being performed. If any danger arises, the user should be ready to press the EMERGENCY STOP button at any time.
- (11) When replacing a part, please contact your local FANUC representative. If a wrong procedure is followed, an accident may occur, causing damage to the robot and injury to the user.
- (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 operating space and that the robot and the peripheral equipment 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 user needs to touch such a part in the heated state, the user 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 robot 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 execution should be given for the robot according to a predetermined method. (See TESTING section of “Controller operator’s manual”.) During the test execution, the maintenance worker should work outside the safety fence.

4 SAFETY OF THE TOOLS AND PERIPHERAL EQUIPMENT

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 abnormality occurs in any other robots or peripheral equipment, even though the robot itself is normal.
- (3) For a system in which the robot and its peripheral equipment 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 equipment 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, operate the robot where insulated from the influence of oil, water, and dust.
- (2) Don't use unconfirmed liquid for cutting fluid and cleaning fluid.
- (3) Adopt limit switches or mechanical stoppers to limit the robot motion, and avoid the robot from collisions against peripheral equipment or tools.
- (4) Observe the following precautions about the mechanical unit cables. Failure to follow precautions may cause problems.
 - Use mechanical unit cable that have required user interface.

- Do not add user cable or hose to inside of the mechanical unit.
 - Please do not obstruct the movement of the mechanical unit 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 the device 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 perform 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 and power-off of the robot is incurred.
 - When alteration is necessary, safety switch is operated by opening safety fence and power-off stop is incurred 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 operates routinely and power-off stop is incurred for the robot.
 - Power-off stop is regularly incurred due to an inappropriate setting for Dual Check Safety (DCS).
- (6) Power-off stop of Robot is executed 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 MECHANICAL UNIT

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 operating spaces 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 robot operation area 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 pinched 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 (R-30iB, R-30iB Mate)

There are following four types of Stopping Robot.

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.

“**Power-Off stop**” performs following processing.

- An alarm is generated, and then the servo power turns off. Instantly the robot stops.
- Execution of the program is paused.

Frequent Power-Off stop of the robot during operation can cause mechanical problems of the robot.

Avoid system designs that require routine or frequent Power-Off stop conditions.

Controlled stop (Category 1 following IEC 60204-1)

The robot is decelerated until it stops, and servo power is turned off.

“**Controlled stop**” performs following processing.

- The alarm “**SRVO-199 Controlled stop**” occurs along with a decelerated stop. The program execution is paused.
- An alarm is generated, and then the servo power turns off.

Smooth stop (Category 1 following IEC 60204-1)

The robot is decelerated until it stops, and servo power is turned off.

“**Smooth stop**” performs following processing.

- The alarm “**SRVO-289 Smooth Stop**” occurs along with a decelerated stop. The program execution is paused.
- An alarm is generated, and then the servo power turns off.
- In Smooth stop, the robot decelerates until it stops with the deceleration time shorter than Controlled stop.

Hold (Category 2 following IEC 60204-1)

The robot is decelerated until it stops, and servo power remains on.

“Hold” performs following processing.

- The robot operation is decelerated until it stops. Execution of the program is paused.

⚠ WARNING

- 1 The stopping distance and time of Controlled stop and Smooth stop are longer than those 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 or Smooth Stop is used. Please refer to the operator's manual of a particular robot model for the data of stopping distance and time.
- 2 In multi arm system, the longest stopping distance and time of Controlled Stop or Smooth Stop among each robot are adopted as those for the system. A risk assessment for the whole robot system which takes into consideration a possibility that the stopping distance and time increase, is necessary on the multi arm system.
- 3 In the system which has extended axis, the longer stopping distance and time of Controlled Stop or Smooth Stop among robot and extended axis are adopted as those for the system. A risk assessment for the whole robot system which takes into consideration a possibility that the stopping distance and time increase, is necessary on the system which has extended axis. Please refer to the extended axis setup procedure of the controller operator's manual for considering the stopping distance and time of the extended axis.
- 4 When Smooth stop occurs during deceleration by Controlled stop, the stop type of robot is changed to Power-Off Stop.
When Smooth stop occurs during deceleration by Hold, the stop type of robot is changed to Power-Off Stop.
- 5 In case of Controlled stop or Smooth Stop, motor power shutdown is delayed for a maximum of 2 seconds. In this case, a risk assessment for the whole robot system is necessary, including the 2 seconds delay.

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

There are the following 3 Stop patterns.

Stop pattern	Mode	Emergency stop button	External Emergency stop	FENCE open	SVOFF input	Deadman switch (*)
A	AUTO	P-Stop	P-Stop	C-Stop	C-Stop	-
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop
C	AUTO	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
D	AUTO	S-Stop	S-Stop	C-Stop	C-Stop	-
	T1	S-Stop	S-Stop	-	C-Stop	S-Stop
	T2	S-Stop	S-Stop	-	C-Stop	S-Stop

P-Stop: Power-Off stop

C-Stop: Controlled stop

S-Stop: Smooth stop

-. Disable

(*) The stop pattern of NTED input is same as Deadman switch.

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

Option	R-30iB/ R-30iB Mate
Standard	A(**)
Controlled stop by E-Stop (A05B-2600-J570)	C(**)
Smooth E-Stop (A05B-2600-J651)	D(**)

(**)R-30iB 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 become 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 time of Controlled stop is longer than those of Power-Off stop, depending on the robot model and axis.

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 time of Controlled stop are longer than those 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.

"Smooth E-Stop Function" option

When "Smooth E-Stop Function" (A05B-2600-J651) option is specified, the stop type of the following alarms becomes Smooth stop in all operation modes (AUTO, T1 and T2 mode).

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-003 Deadman switch released	Both deadman switches on Teach pendant are released.

Alarm	Condition
SRVO-007 External emergency stops	External emergency stop input (EES1-EES11, EES2-EES21) is open.
SRVO-037 IMSTP input (Group: %d)	IMSTP input (*IMSTP signal for a peripheral device interface) is OFF.
SRVO-232 NTED input	NTED input (NTED1-NTED11, NTED2-NTED21) 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.
SRVO-410 DCS SSO NTED input	In DCS Safe I/O connect function, SSO[5] is OFF.
SRVO-419 DCS PROFIsafe comm. error	PROFINET Safety communication error occurs.

Smooth stop is different from **Power-Off stop** as follows:

- In Smooth stop, the robot is stopped along 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 Smooth 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 time of Smooth stop is longer than those of Power-Off stop, depending on the robot model and axis.

Smooth stop is different from **Controlled stop** as follows:

- The stopping distance and time of Smooth stop is normally shorter than those of Controlled stop, depending on the robot model and axis.

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 time of Smooth stop are longer than those 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.

8 STOP TYPE OF ROBOT (R-30iB Plus, R-30iB Mate Plus)

There are following three types of Stop Category.

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

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.

“**Stop Category 0**” performs following processing.

- An alarm is generated, and then the servo power turns off. Instantly the robot stops.
- Execution of the program is paused.

Frequent Category 0 Stop of the robot during operation can cause mechanical problems of the robot. Avoid system designs that require routine or frequent Category 0 Stop conditions.

Stop Category 1 following IEC 60204-1 (Controlled Stop, Smooth Stop)

The robot is decelerated until it stops, and servo power is turned off.

“**Stop Category 1**” performs following processing.

- The alarm "SRVO-199 Controlled stop" or "SRVO-289 Smooth Stop" occurs along with a decelerated stop. The program execution is paused.
- An alarm is generated, and then the servo power turns off.

In Smooth stop, the robot decelerates until it stops with the deceleration time shorter than Controlled stop. The stop type of Stop Category 1 is different according to the robot model or option configuration. Please refer to the operator's manual of a particular robot model.

Stop Category 2 following IEC 60204-1 (Hold)

The robot is decelerated until it stops, and servo power remains on.

"Stop Category 2" performs following processing.

- The robot operation is decelerated until it stops. Execution of the program is paused.



WARNING

- 1 The stopping distance and time of Stop Category 1 are longer than those of Stop Category 0. A risk assessment for the whole robot system which takes into consideration the increased stopping distance and stopping time is necessary when Stop Category 1 is used. Please refer to the operator's manual of a particular robot model for the data of stopping distance and time.
- 2 In multi arm system, the longest stopping distance and time of Stop Category 1 among each robot are adopted as those for the system. A risk assessment for the whole robot system which takes into consideration a possibility that the stopping distance and time increase, is necessary on the multi arm system.
- 3 In the system which has extended axis, the longer stopping distance and time of Stop Category 1 among robot and extended axis are adopted as those for the system. A risk assessment for the whole robot system which takes into consideration a possibility that the stopping distance and time increase, is necessary on the system which has extended axis. Please refer to the extended axis setup procedure of the controller operator's manual for considering the stopping distance and time of the extended axis.
- 4 When Stop Category 1 occurs during deceleration by Stop Category 2, the stop type of robot is changed to Stop Category 0.
- 5 In case of Stop Category 1, motor power shutdown is delayed for a maximum of 2 seconds. In this case, a risk assessment for the whole robot system is necessary, including the 2 seconds delay.

When the emergency stop button is pressed or the FENCE is open, the stop type of robot is Stop Category 0 or Stop Category 1. The configuration of stop type for each situation is called *stop pattern*. The stop pattern is different according to the option configuration.

There are the following 3 Stop patterns.

Stop pattern	Mode	Emergency stop button	External Emergency stop	FENCE open	SVOFF input	Deadman switch (*)
A	AUTO	Category 0	Category 0	Category 1	Category 1	-
	T1	Category 0	Category 0	-	Category 1	Category 0
	T2	Category 0	Category 0	-	Category 1	Category 0
C	AUTO	Category 1	Category 1	Category 1	Category 1	-
	T1	Category 0	Category 0	-	Category 1	Category 0
	T2	Category 0	Category 0	-	Category 1	Category 0
D	AUTO	Category 1	Category 1	Category 1	Category 1	-
	T1	Category 1	Category 1	-	Category 1	Category 1
	T2	Category 1	Category 1	-	Category 1	Category 1

Category 0: Stop Category 0

Category 1: Stop Category 1

-: Disable

(*) The stop pattern of NTED input is same as Deadman switch.

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

The case R651 is specified.

Option	R-30iB Plus/ R-30iB Mate Plus
Standard	C(**)
Old Stop Function (A05B-2670-J680)	A(**)
All Smooth Stop Function (A05B-2670-J651)	D(**)

The case R650 is specified.

Option	R-30iB Plus/ R-30iB Mate Plus
Standard	A(**)
Stop Category 1 by E-Stop (A05B-2670-J521)	C(**)
All Smooth Stop Function (A05B-2670-J651)	D(**)

(**)R-30iB Mate Plus 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.

"Old Stop Function" option

When "Old Stop Function" (A05B-2670-J680) option is specified, the stop type of the following alarms becomes Stop Category 0 in AUTO mode.

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.

Stop Category 0 is different from **Stop Category 1** as follows:

- In Stop Category 0, 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 stopping distance and time of Stop Category 0 is shorter than those of Stop Category 1, depending on the robot model and axis.

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.

"All Smooth Stop Function" option

When "All Smooth Stop Function" (A05B-2670-J651) option is specified, the stop type of the following alarms becomes Stop Category 1 in all operation modes (AUTO, T1 and T2 mode).

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-003 Deadman switch released	Both deadman switches on Teach pendant are released.
SRVO-007 External emergency stops	External emergency stop input (EES1-EES11, EES2-EES21) is open.
SRVO-037 IMSTP input (Group: %d)	IMSTP input (*IMSTP signal for a peripheral device interface) is ON.
SRVO-232 NTED input	NTED input (NTED1-NTED11, NTED2-NTED21) 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.
SRVO-410 DCS SSO Ext Emergency Stop	In DCS Safe I/O connect function, SSO[5] is OFF.
SRVO-419 DCS PROFIsafe comm. error	PROFINET Safety communication error occurs.

Stop Category 1 is different from **Stop Category 0** as follows:

- In Stop Category 1, the robot is stopped along 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 Stop Category 1, physical impact is less than Stop Category 0. 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 time of Stop Category 1 is longer than those of Stop Category 0, depending on the robot model and axis.

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 time of Stop Category 1 are longer than those of Stop Category 0. 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.

"Stop Category 1 by E-Stop" option

When "Stop Category 1 by E-Stop" (A05B-2670-J521) option is specified, the stop type of the following alarms become Category 1 Stop but only in AUTO mode. In T1 or T2 mode, the stop type is Category 0 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.

Stop Category 1 is different from **Stop Category 0** as follows:

- In Stop Category 1, the robot is stopped along 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 Stop Category 1, physical impact is less than Stop Category 0. 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 time of Stop Category 1 is longer than those of Stop Category 0, depending on the robot model and axis.

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 time of Stop Category 1 are longer than those of Stop Category 0. 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.

170406

TABLE OF CONTENTS

SAFETY PRECAUTIONS	s-1
1 INTRODUCTION	1
2 iPendant WEB BROWSER	2
2.1 OVERVIEW	2
2.2 BROWSER MENUS	2
2.3 HELP FOR BROWSER MENUS	3
2.4 EXTENDED STATUS WINDOW	3
2.5 BACKUP AND RESTORE OF BROWSER FILES	4
3 USING FANUC iPendant CONTROLS	5
3.1 iPendant CONTROLS SUMMARY	5
3.2 RECOMMENDED ENVIRONMENT	5
3.3 INSTALLATION	6
3.3.1 ActiveX Control Shortcut	7
3.4 CONTROL FEATURES SUMMARY	7
4 SharePoint Designer 2007	9
4.1 WORKING WITH WEBS	9
4.2 WORKING WITH PAGES	9
4.2.1 File Names	10
4.2.2 Meta Tags	11
4.2.3 Page Properties	11
4.2.4 Window Size	11
4.2.5 Positioning	13
4.2.6 HTML Editing	13
4.2.7 Font Size	13
4.2.8 Font Name	14
4.2.9 Images	14
4.2.10 Links	14
4.2.11 Forms	15
4.2.12 Frames	16
4.2.13 Themes and Styles	16
4.2.14 Ajax	16
4.2.15 jQuery	17
4.2.16 Scripting Elements	17
4.2.16.1 Adding script	18
4.2.16.2 Debugging in source view	19
4.2.16.3 Debugging in internet explorer	19
4.2.16.4 Scripting iPendant controls	19
4.2.17 DOM Elements	20
4.3 PUBLISHING YOUR WEB	21
5 MAKING A CUSTOM iPendant SCREEN USING THE iPendant CONTROLS	23
5.1 CONTROL ARRANGEMENT	23
5.2 COMMON CONTROL PROPERTIES	24
5.2.1 Object Tag	24

5.2.2	DataType and DataIndex	24
5.2.3	Images	25
5.2.4	Border	26
5.2.5	Colors	26
5.2.6	Fonts	26
5.2.7	Alignment.....	27
5.2.8	Monitor.....	27
5.2.9	Function Key ViewType	27
5.2.10	Caption	28
5.2.11	Pulse DO.....	29
5.2.12	Port Simulation.....	29
5.2.13	Positions	30
5.2.14	Indirect DataType and DataIndex	32
5.2.15	PANEID	33
5.2.16	Pipe.....	33
5.2.17	SetFocus	33
5.2.18	ClickMe	34
5.3	CHECK CONDITION	34
5.3.1	Supported Controls.....	35
5.3.2	User Interface	35
5.3.3	Error Message 1.....	36
5.3.4	Error Message 2.....	37
5.3.5	Display other than iPendant	38
5.3.6	Caution and Limitations	40
5.3.7	ComboBox Control Limitation.....	41
5.3.8	Execution Control Limitation.....	41
5.3.9	Help Control Limitation	41
5.4	CONTROL DESCRIPTION.....	41
5.4.1	Label Control.....	41
5.4.2	EditText Control	43
5.4.3	ToggleLamp Control	44
5.4.4	CommandButton Control	47
5.4.5	ToggleButton Control.....	48
5.4.6	Multi Control	51
5.4.7	AutoChange Control.....	52
5.4.8	MenuChange Control	54
5.4.9	ButtonChange Control.....	55
5.4.10	Help Control.....	56
5.4.11	ComboBox Control	56
5.4.12	Execution Control.....	59
5.5	CONTROL DESIGN ADVICE	60
5.5.1	Error Code Dialog	60
5.5.2	Error Code Messages.....	60
6	USING THE CHART CONTROL	63
6.1	CONTROL ARRANGEMENT	65
6.2	COMMON CHART CONTROL PROPERTIES	66
6.2.1	Object Tag	66
6.2.2	Fonts	66
6.2.3	General Chart Properties	67
6.2.3.1	Caption & CaptionFontSize.....	68
6.2.3.2	Annotation & AnnotateFontSize	68
6.2.3.3	Name.....	68
6.2.3.4	Border	68

6.2.3.5	Foreground & Background Colors	68
6.2.3.6	Pipe	69
6.2.3.7	PipeMonRate	69
6.2.3.8	FastLoad	69
6.2.4	Chart config tab	69
6.2.4.1	ChartType	70
6.2.4.2	Orientation	70
6.2.4.3	LabelFontSize	71
6.2.4.4	LineScaleActive	71
6.2.4.5	DataFontSize	71
6.2.4.6	DataFormat	71
6.2.4.7	DataShowValues	72
6.2.4.8	SampleScaleAspect	72
6.2.4.9	SampleScale	72
6.2.4.10	SampleScaleFormat	73
6.2.4.11	SampleGrid	73
6.2.4.12	DataScaleFormat	73
6.2.4.13	DataScale	73
6.2.4.14	DataScaleFormat	74
6.2.4.15	DataGrid	74
6.2.4.16	GridColor	74
6.2.4.17	GridType	74
6.2.5	Channel config tab	74
6.2.5.1	ChN_Name	75
6.2.5.2	ChN_Color	75
6.2.5.3	ChN_Source	75
6.2.5.4	ChN_Data	76
6.2.5.5	ChN_Digital	76
6.2.5.6	ChN_State	76
6.2.5.7	ChN_Rate	77
6.2.5.8	ChN_DataScale	77
6.2.5.9	ChN_DataGrid	77
6.2.5.10	ChN_AutoRange	77
6.2.6	Miscellaneous	78
6.2.6.1	SampleMarkerN	78
6.2.6.2	SampleMarkerColor	78
6.2.6.3	ChN_DataMarkerN	78
6.2.6.4	Ch_Data_N	78
6.2.6.5	ChN_Clear	79
6.2.6.6	ChartClear	79
6.3	CHART CONTROL DESCRIPTION	79
6.3.1	Bar Chart Control	79
6.3.2	Line Chart Control	81
6.4	PROPERTIES WITH ADDITIONAL OPTIONAL VALUES	83
6.5	CHARTING CONTROL DESIGN ADVICE	83
6.5.1	Error Handling	83
7	USING THE DRAWING CONTROL	84
7.1	CONTROL ARRANGEMENT	85
7.2	COMMON DRAWING CONTROL PROPERTIES	86
7.2.1	Object Tag	86
7.2.2	Fonts	86
7.2.3	General tab	87
7.2.3.1	Caption	87
7.2.3.2	Name	88
7.2.3.3	Foreground & Background Colors	88

	7.2.3.4	Pipe	88
	7.2.3.5	PipeMonRate	89
	7.2.3.6	Border	89
	7.2.3.7	FastLoad	89
7.2.4		General 2 tab	89
	7.2.4.1	Layer control.....	89
	7.2.4.2	Scale.....	90
	7.2.4.3	InvertY	90
	7.2.4.4	Data.....	90
7.2.5		The Entities.....	90
	7.2.5.1	Text	90
	7.2.5.2	Line	91
	7.2.5.3	Path	91
	7.2.5.4	Circ	91
	7.2.5.5	Rect.....	92
	7.2.5.6	Diam	92
	7.2.5.7	Imag	92
7.2.6		Miscellaneous Properties.....	93
	7.2.6.1	Clear.....	93
	7.2.6.2	Delete.....	93
	7.2.6.3	LayerOn	93
	7.2.6.4	LayerOff	93
7.2.7		Dynamic Entity modifications.....	93
8		USING THE GRID CONTROL.....	94
8.1		INSERTING A GRID CONTROL ON A WEB PAGE.....	95
8.2		COMMON GRID CONTROL PROPERTIES	96
	8.2.1	Object Tag	96
	8.2.2	Fonts	96
	8.2.3	General tab	97
		8.2.3.1 Caption.....	98
		8.2.3.2 Name.....	98
		8.2.3.3 Foreground & Background Colors	98
		8.2.3.4 Pipe	98
		8.2.3.5 Pipe Mon Rate	99
		8.2.3.6 Border	99
		8.2.3.7 FastLoad	99
8.3		DISPLAY CONCEPTS.....	99
	8.3.1	Conventions.....	99
	8.3.2	Alignment.....	101
	8.3.3	Display, Pan and Zoom	101
	8.3.4	Dynamic Display	102
	8.3.5	Rendering	103
8.4		XML TO THE GRID	103
	8.4.1	XML tag	105
	8.4.2	GRID tag	105
	8.4.3	TILE tag	106
	8.4.4	TEXT tag.....	107
	8.4.5	LINE tag.....	109
	8.4.6	RECTANGLE tag	109
	8.4.7	POLYGON tag	110
	8.4.8	CIRCLE tag	111
	8.4.9	IMAGE tag.....	112
	8.4.10	BUTTON tag	113
	8.4.11	DISPLAY tag	115

8.4.11.1	Scroll bars with the DISPLAY tag	116
8.4.12	DEL TILE tag	118
8.4.13	Modifying SHAPES	119
8.4.14	Modifying TILES	119
8.4.15	Dynamic data	120
8.4.15.1	Using the Name Parameter	120
8.4.15.2	Using the Pipe parameter	120
8.4.15.3	Using the Data parameter	120
9	PANE LINKING	122
9.1	INTRODUCTION	122
9.2	DISPLAY MENU FUNCTIONS	122
9.3	COMMANDING LINKS FROM KAREL	123
9.4	GENERIC LINKING FUNCTIONALITY	123
9.5	CONTEXT SENSITIVE HELP	123
9.6	ABOUT IDENTIFIERS	124
9.7	NEW/MODIFIED SYSTEM VARIABLES	125
9.7.1	\$TX	125
9.7.2	\$ALM_IF	125
9.7.3	\$UI_USERVIEW	125
9.7.4	\$UI_CONFIG	126
9.7.5	\$UI_MENHIST[5]	126
9.7.6	\$UI_PANEDATA[9]	126
9.7.7	\$UI_PANELINK[5]	126
9.8	GENERIC LINKING DETAILED INFORMATION	127
9.9	RELATED VIEWS	128
9.10	EXAMPLES	129
 APPENDIX		
A	EXTENDED STATUS TEMPLATE	133
B	CUSTOM SCREEN EXAMPLES	134
B.1	USING TABLES TO SET SIZE	134
B.2	SIMPLE HMI EXAMPLE	135
B.3	FORM EXAMPLE	136
B.3.1	Overview	136
B.3.2	Web Page	136
B.3.3	KAREL Program	137
B.4	AJAX EXAMPLE	139
B.4.1	Overview	139
B.4.2	Web Page	139
B.4.3	KAREL Program	140
B.4.4	Web Page using iPendant Control Instead	141
C	USING DISCTRL_FORM TO DISPLAY A WEB PAGE	142
C.1	OVERVIEW	142
C.2	WEB PAGE EXAMPLE 1	143
C.3	KAREL PROGRAM EXAMPLE 1	145
C.4	WEB PAGE EXAMPLE 2	146
C.5	KAREL PROGRAM EXAMPLE 2	147

D SYSTEM VARIABLES 150

E KAREL BUILTINS..... 151

 E.1 FORCE_LINK BUILT-IN 151

 E.2 GET_DEV_INFO BUILT-IN 151

 E.3 DISPLAY WEB PAGE MACRO 152

 E.4 DISCTRL_DIAG..... 153

 E.4.1 Dialog Box XML File 154

 E.4.1.1 Tags and Attributes..... 155

 E.4.1.2 XML content example 158

 E.4.1.3 KAREL program example..... 159

1 INTRODUCTION

This document describes how to customize the FANUC Robotics *i*Pendant. Please refer to the FANUC Robotics SYSTEM R-30*i*B Controller Setup and Operations Manual for information relating to *i*Pendant screen navigation and *i*Pendant-specific functions.

The FANUC Robotics *i*Pendant provides the capability for the user to easily develop custom screens using the EasyPanel development environment and the custom *i*Pendant components supplied by FANUC Robotics America Corporation. This document is meant to provide detailed information on how to create and use these custom screens on an *i*Pendant.

Please refer to the FANUC Robotics SYSTEM R-30*i*B Controller Internet Options Setup and Operation Manual for information relating to the Web Server and Server Side Includes (SSI). It also contains information about connecting your PC to the robot home page and using Monitor *i*Pendant (ECHO) and Navigate *i*Pendant (CGTP). Steps for troubleshooting your PC connection to the robot can be found in that manual.

2 iPendant WEB BROWSER

2.1 OVERVIEW

The *iPendant* uses Internet Explorer Embedded, a web browser developed by Microsoft ®. The Internet Explorer Embedded browser is a version of Internet Explorer 7 with additional features, such as hardware acceleration for graphics and support for touch gestures.

Microsoft is registered trademark of Microsoft Corporation.

2.2 BROWSER MENUS

The *iPendant* BROWSER is available from the BROWSER entry on the second page of the Main Menu.

The *iPendant* BROWSER screen allows you to access web pages on the robot or web pages on any device on the network with the robot. Please refer to the FANUC Robot series R-30*iA*/R-30*iA* Mate/R-30*iB* CONTROLLER Ethernet Function OPERATOR'S MANUAL for information on using the robot's web server.

You can add your own links to the BROWSER [TYPE] menu, which is displayed by the F1 key. Ten links are available in this [TYPE] menu. They are defined in the system variable \$TX_SCREEN[n] where:

\$TX_SCREEN[n].\$DESTINATION specifies the browser link/URL.

\$TX_SCREEN[n].\$SCREEN_NAME specifies the screen name to display in the menu.

Where n is 1 to 10

The screen name must be provided or the entry will not be shown.

Some example links are shown below:

Link	Description
"/fr/example.htm"	example.htm which is on fr: device, uses relative address. Always use relative addressing for your local robot.
"http://pderob011"	Default web page for remote robot pderob011
"http://pderob011/fr/example.htm"	fr:example.htm on remote robot pderob011
"http://localhost"	Default web page for a Virtual Robot on a PC
"http://IP address of the robot"	Robot Home Page

For links other than the connected robot (/fr/example.htm), the "Internet Connectivity Option" must be loaded and the Proxy Server properly configured. Please refer to the FANUC Robot series R-30*iA*/R-30*iA* Mate/R-30*iB* CONTROLLER Ethernet Function OPERATOR'S MANUAL for more information.

For example, if you set the following:

\$TX_SCREEN[1].\$DESTINATION = 'http://pderob011'

\$TX_SCREEN[1].\$SCREEN_NAME = 'pderob011'

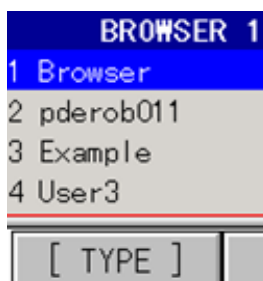
\$TX_SCREEN[2].\$DESTINATION = '/fr/example.htm'

```
$TX_SCREEN[2].$SCREEN_NAME = 'Example'
```

```
$TX_SCREEN[3].$DESTINATION = '/fr/status.htm'
```

```
$TX_SCREEN[3].$SCREEN_NAME = 'User3'
```

The BROWSER [TYPE] menu would appear as follows:



2.3 HELP FOR BROWSER MENUS

You can use the integrated Help system to provide Help for your specific BROWSER screen when the user presses the HELP/DIAG key while in your screen. The help system will try to find the specific file indicated in the table below, based on which \$TX_SCREEN[n] variable corresponds to that entry. To provide HELP, simply copy your help file to the file indicated on the FR: device.

NOTE

File names are case insensitive.

\$TX_SCREEN[1]	FR:¥H17D0A.HTM
\$TX_SCREEN[2]	FR:¥H17D0B.HTM
\$TX_SCREEN[3]	FR:¥H17D0C.HTM
\$TX_SCREEN[4]	FR:¥H17D0D.HTM
\$TX_SCREEN[5]	FR:¥H17D0E.HTM
\$TX_SCREEN[6]	FR:¥H17D0F.HTM
\$TX_SCREEN[7]	FR:¥H17D10.HTM
\$TX_SCREEN[8]	FR:¥H17D11.HTM
\$TX_SCREEN[9]	FR:¥H17D12.HTM
\$TX_SCREEN[10]	FR:¥H17D13.HTM

2.4 EXTENDED STATUS WINDOW

The *i*Pendant allows you to add your own links to the Extended Status window, which is on the left side of the Status/Single Display. 255 links are available in the Extended Status window.

A series of .STM files will be used to make up the Extended Status window. The system will create the table of links based on the files it finds. The files will be ordered by ascending number. The numbers do not have to be sequential. If the same number is already used, the file will be inserted after the one already found.

To create a new link, follow these steps:

STEP

1. Create a web page based on the template in Appendix A.
2. Copy your file using the following convention. File names are case insensitive.

FR:EXTn.STM

Where n is 1 to 255

3. Restart the controller to CONTROLLED START and select FCTN> COLD.

Your new Extended status link should now be available on the Extended Status Page.

2.5 BACKUP AND RESTORE OF BROWSER FILES

You can copy your web pages and associated files to FR: device. These files will be backed up and restored as Application Files. When you select "Application," all files listed in the \$FILE_APPBCK system variable will be saved. The following table describes the various kinds of application files. You should not modify this system variable.

Name	Description
*.VR	KAREL variable files
*.PC	KAREL program files
*.TX	Dictionary files
FR:*.HTM	HTML web pages on FR: device
FR:*.STM	HTML web pages using <i>i</i> Pendant Controls or Server Side Includes on FR: device
FR:*.GIF	GIF image files on FR: device
FR:*.JPG	JPEG image files on FR: device
FR:*.JS	JavaScript include files on FR: device
FR:*.CSS	Cascading Style Sheets on FR: device
FR:CUSTLIST.DT	Browser Favorites

3 USING FANUC *i*Pendant CONTROLS

The FANUC *i*Pendant Controls are Microsoft ActiveX controls that allow you to create operator panel context for the *i*Pendant.

Microsoft is registered trademark of Microsoft Corporation.

ActiveX is registered trademark of Microsoft Corporation.

NOTE

Other Microsoft ActiveX controls cannot be used with the *i*Pendant.

3.1 *i*Pendant CONTROLS SUMMARY

The *i*Pendant Controls main functions are as follows:

- Permit the dynamic display and input of Register, System and KAREL Variables, and I/O values.
- Change between web pages (Manually and Automatically).

3.2 RECOMMENDED ENVIRONMENT

The following items are required to develop and run custom screens on the *i*Pendant in the EasyPanel environment:

- FANUC Robotics *i*Pendant Controls installed on the PC.
- Microsoft® Office SharePoint® Designer 2007™ (freeware) installed on the PC. Do not use SharePoint Designer 2010 since it is not a generic HTML editor.

SharePoint Designer 2007 is the recommended application for development because it interacts with ActiveX controls by displaying custom property pages and showing more information at design time. It is available to download from Microsoft. All references to SharePoint Designer and SharePoint Designer screens in this document are for SharePoint Designer 2007. A file named ipctrls.stm is provided with the *i*Pendant controls setup. This file contains samples of each control that can be used to cut and paste controls into your custom pages if desired.

Other web authoring tools, such as Dreamweaver, can also be used; however you may have to enter all the ActiveX control parameters manually. A file named ipctrls_verbose.stm is provided with the *i*Pendant controls setup. This file contains samples of each control with all parameters that can be used to cut and paste controls into your custom pages if desired.

NOTE

The R-30*i*A *i*Pendant does not support style sheets so Microsoft FrontPage is still recommended for development with the older *i*Pendant. Please refer to V7.70 *i*Pendant Screen Customize Function for R-30*i*A *i*Pendant.

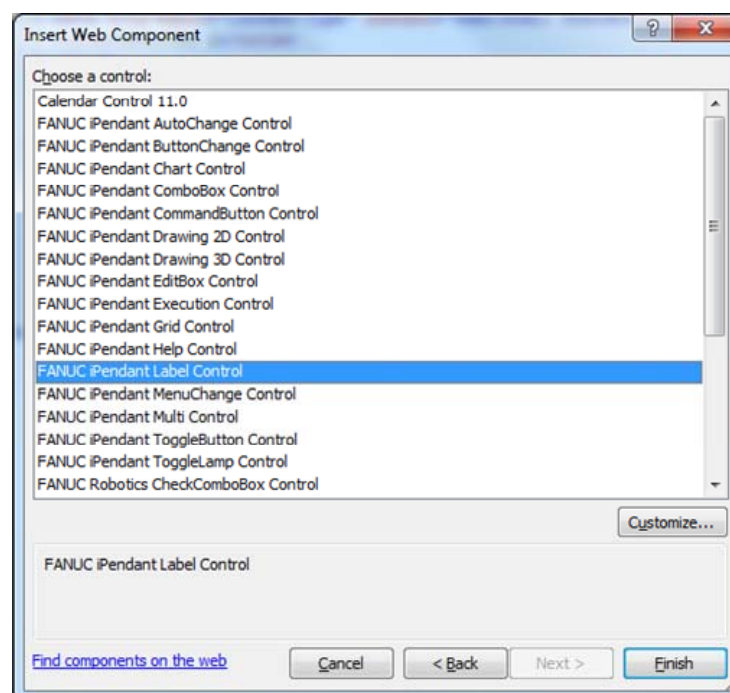
3.3 INSTALLATION

To create custom screen, FANUC iPendant Controls must be installed to your PC. Please install ROBOGUIDE because install of ROBOGUIDE installs automatically iPendant Controls, too. To confirm installation, use following procedure.

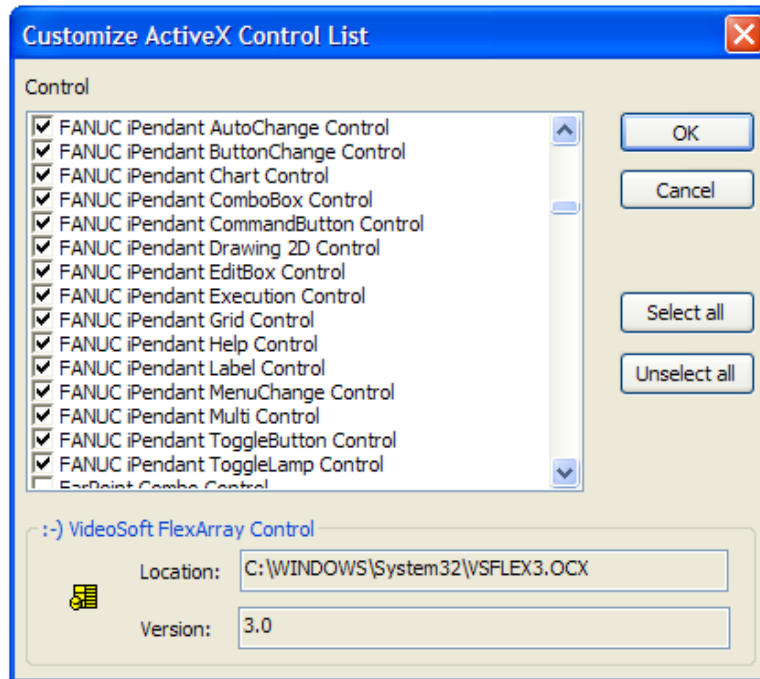
To verify correct installation:

STEP

1. Start Microsoft Office SharePoint Designer 2007 and open a new blank page. Refer to your SharePoint Designer 2007 documentation for more information.
2. Select Insert | Web Component from the menu bar. This brings up a dialog box. Select Advanced Controls and choose ActiveX Control. This brings up the Insert ActiveX Control dialog box.



3. If FANUC iPendant Controls do not appear, select Customize button. If they were installed properly, you will see a screen where you can select all the FANUC iPendant Controls.



4. If you insert an ActiveX component, and it shows up with the following image in SharePoint Designer 2007:
You most likely installed the .ocx on a drive that is currently no longer available. (i.e. a networked drive)



3.3.1 ActiveX Control Shortcut

To add “ActiveX Control” onto the Toolbar, follow these steps. You only need to do this once:

1. Select View | Toolbars | Customize from the menu bar.
2. From the Commands tab, select Insert category, cursor to ActiveX Control
3. With the left mouse, drag the command out of the dialog box to the Toolbar

3.4 CONTROL FEATURES SUMMARY

The following is a brief description of each control that is included with the FANUC Robotics *i*Pendant Controls:

Label

Used to display the value of Register, Variable and I/O. Also used to display fixed strings and the combination of fixed strings and digits.

EditBox

Used to change the value of a Register, a Variable or an I/O point. Popup keyboard and *i*Pendant numeric keypad are supported.

ToggleLamp

Used to change the color or image displayed by the control if the value of a Register, a Variable or an I/O point, fulfills the specified single condition. Three types of lamps are available: panel, circle, or an image. This control can also be used to display a fixed image.

CommandButton

Used to write the specified value to a Register, a Variable or an I/O point. Two types of buttons are available: rectangular pushbutton and image.

ToggleButton

Used to write one of two specified values to a Register, a Variable or an I/O point based on the state of the button. Three types of buttons are available: rectangular pushbutton, checkbox and image.

Multi

Used to display up to 10 different images or strings based on the value of a Register, a Variable or an I/O point. It can be used to create simple animations like a progress bar.

AutoChange

Used to change a page being displayed automatically based on the value of a Register, a Variable or an I/O point. This can be used to change the displayed page from a TP or KAREL program.

MenuChange

Used to select a page to be displayed from a popup menu. The pages are displayed when the button is pressed. Two types of buttons are available: rectangular pushbutton and image.

ButtonChange

Used to display the specified page. Two types of buttons are available: rectangular pushbutton and image.

ComboBox

Used to change the value of a Register, a Variable or an I/O point using a popup menu selection.

Help

Used to display the specified help page when the HELP key is pressed.

Execution

Used to run the specified KAREL program when the page is loaded.

Chart

Used to graphically display data from the controller as a bar or line chart.

Drawing 2D

Used to draw Text, Lines, Paths, Circles, Rectangles, Diamonds or images on the iPendant

Drawing 3D

Used to draw the robot, tooling, parts, and other cell components on the iPendant

Grid

Used to render Text, Lines, Circles, Rectangles, and Images registered to a grid.

Each control has several settable properties, which will be described later, but in general each control allows you to set:

- Which Register, System variable, KAREL variable, or I/O type to use.
- The border of the control, which can be 3D (thin and bold border), straight (black and forecolor) and none.
- The size, color, and font used by the control.

Section 5 describes how to use these controls to make an Operator Panel.

4 SharePoint Designer 2007

This chapter gives tips on using SharePoint Designer 2007 to develop web pages for the *iPendant*. Refer to the SharePoint Designer 2007 documentation for complete details.

4.1 WORKING WITH WEBS

SharePoint Designer 2007 works with what is called a web to create and manage your site development. A web is all the pages contained in your website. Whenever you work within SharePoint Designer 2007, you want to make sure you are working within a web. If you are working with ROBOGUIDE, use the Open Web commands under File, and point to the MC directory of your virtual robot as your web. SharePoint Designer 2007 will insert several files that it needs which will not affect the operation of the virtual robot.

If you are not working in ROBOGUIDE, then create a new web (from the File menu) somewhere on your PC to store your files. Later you can “Publish” these files directly to the controller.

NOTE

You should put all your web files into one directory without any subdirectories. Subdirectories are not fully supported on the R-30*iB* Controller.

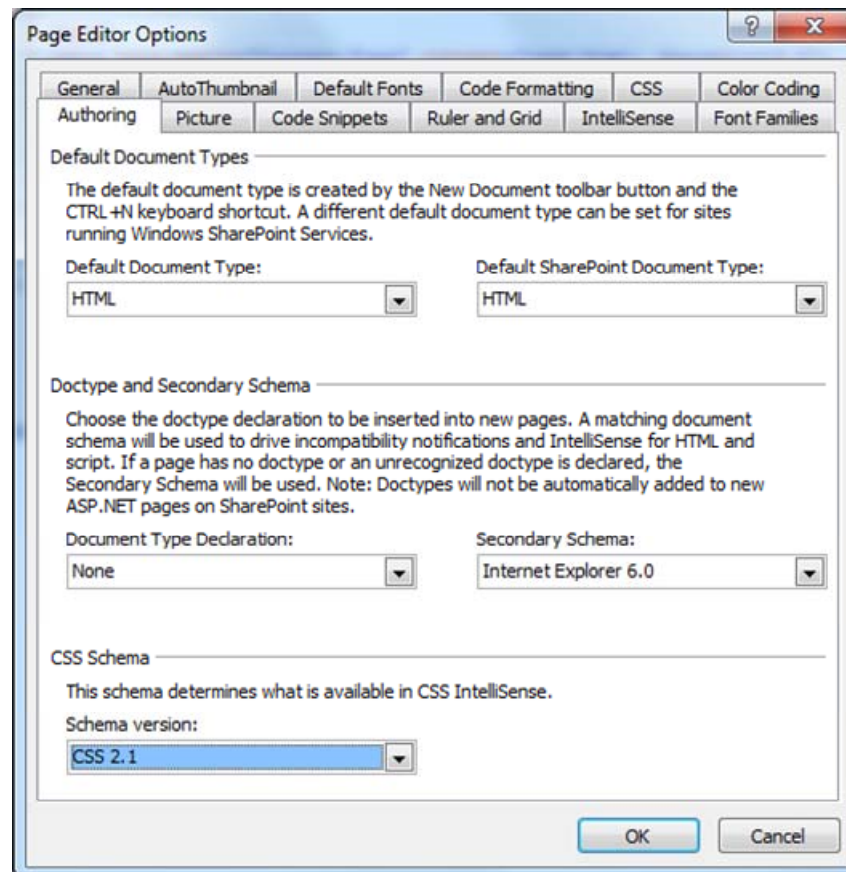
4.2 WORKING WITH PAGES

When SharePoint Designer 2007 is used, you can create content by inserting *iPendant* controls on a web page. Many HTML tags and forms are also available for use. SharePoint Designer 2007 gives you the option of viewing web pages as a normal WYSIWYG document (Design) or as HTML (Code), or both (Split).

NOTE

When viewing your pages the FANUC Robotics *iPendant* Controls will appear static, since they are not connected to the robot controller.

The dialog box available from Tools | Page Editor Options | Authoring Tab allows you to set up your compatibility options specifically for *iPendant*:



HTML should be your default document type. ASPX is not supported.

The dialog box available from Site | Site Settings | Language allows you to set up your default page encoding.

Setting it to US/Western European (ISO) will put the following in any new web pages.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

4.2.1 File Names

Any web page that contains FANUC Robotics *i*Pendant Controls must be saved with an .stm file extension or *i*Pendant will not recognize the FANUC Robotics *i*Pendant controls. The file name must follow the robot's 12.3 naming convention with no spaces.

Files that are specific to a particular language should end in a suffix. However, the links should not include the suffixes. The web server will automatically find the correct file based on the current language setting of the robot. For instance, if you link to arc.stm and the robot language is set to English, the web server will find arc.stm if it exists. If it does not exist the Web Server will look for arceg.stm.

Language	Suffix
English	eg
Japanese	jp
Kanji	kn
French	fr
German	gr
Spanish	sp
Chinese	ch
Taiwanese	tw
Portuguese	pt
Other	ot

4.2.2 Meta Tags

Meta tags are used to instruct the browser about the page. One common use is to instruct the browser to always refresh a page when the page is loaded. Use the following tag on any page that you do not want cached by the browser:

```
<meta http-equiv="Cache-Control" content="no-cache">
```

NOTE

If FANUC Robotics *i*Pendant Controls are used on a page, the page does not need to be refreshed and the above Meta tag should not be included since they will slow the display of the page.

Any web pages that use Server Side Includes (SSI) will not be cached even if the meta tag is omitted. This is because the controller needs to substitute the SSI for the current value when the web page is displayed. The *i*Pendant background color is an SSI.

```
<body bgcolor="<!-- #echo var=_BGCOLOR -->">
```

When the background color changes, the browser cache is automatically cleared. Therefore if the background color is the only SSI in the web page or the SSI value will not change while the controller power is on, then you may tell the browser to cache your page. Use the following tag on any page that you want cached by the browser that uses SSI:

```
<meta http-equiv="Cache-Control" content="public">
```

If you web page has no SSI, then you do not require this tag since the default behavior of the browser is to cache all web pages.

Please refer to the FANUC Robot series R-30*i*A/R-30*i*A Mate/R-30*i*B CONTROLLER Ethernet Function OPERATOR'S MANUAL for information on using SSI.

4.2.3 Page Properties

In the Design View, right-click the page, and then click **Page Properties** on the shortcut menu. This allows you to change the background color and text color of a page. The page TITLE specified will be shown in the Title bar on the *i*Pendant when this page is displayed. To use the *i*Pendant background color, you can type this into the HTML.

```
<body bgcolor="<!-- #echo var=_BGCOLOR -->">
```

4.2.4 Window Size

To avoid scroll bars on your web pages, the size of the page should be the same as the size of the *i*Pendant screen:

Window	Width	Height
Single	632	367
Double Prim	313	379
Double Dual	313	379
Triple Prim	313	379
Triple Dual	313	184
Triple Third	313	176
Status/Single Prim	405	375

Window	Width	Height
Double Horizontal Prim	631	184
Double Horizontal Dual	631	176
Triple Horizontal Prim	313	184
Triple Horizontal Dual	313	184
Triple Horizontal Third	631	176

NOTE

The *i*Pendant is supposed to be 640 x 480 but it is 8 pixels short both in width and height. The height has been removed from the function keys so no difference is noticed. However, the width will be 8 pixels larger on your PC.

A table with 1 row and column can be inserted on a blank web page as a guideline with the width and height set (in pixels) to the appropriate values from the table above.

NOTE

Both the Horizontal and Vertical Scroll bars are 16 pixels wide, so if you design pages that exceed the above sizes, you will need to take this into account.

An alternative is to set the table height and width to 100% instead of specifying a pixel value. If the page does not require scroll bars, you can remove the space on the right reserved for the scroll bars by using a style:

```
<style>
  body {
    overflow: hidden;
  }
</style>
```

Use additional tables inside of this table for positioning of *iPendant* Controls as discussed in the next section. An example for whole mode is contained in Appendix B.

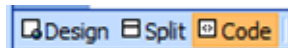
4.2.5 Positioning

SharePoint Designer 2007 has the ability to design web pages with pixel-precise positioning. This feature uses cascading style sheets (CSS).

Another way to position controls, text and images on an *iPendant* screen is through the use of tables embedded inside of other tables. Use the Cell and Table Properties dialog boxes to get pixel-perfect adjustments by right clicking on the table or cell and adjusting the necessary parameters. A Table toolbar is also available. See Using Table examples in Appendix B.

4.2.6 HTML Editing

Inserting code into your web page is done in the “Code” view of SharePoint Designer 2007. To select this view, select either the Split or Code view from the view bar at the bottom of the page.



To quickly select a section of code in the code window, use the Quick Tag Selector at the top of the page. To enable the Quick Tag Selector go to View | Quick Tag Selector from the menu bar. You can select the beginning tag in a tag set and SharePoint Designer 2007 will highlight all the information that falls between the beginning tag and the ending tag for that set. Or you can select View | Reveal Tags to have the tags displayed in the Design View.

4.2.7 Font Size

Font size will be defined as Points (pt size) or Pixels (px size). Most style sheets and HTML tags use Pixels. The *iPendant* Controls use Points. When the TrueFont parameter is 0, then the mapping occurs as shown within the table so web pages designed with the R-30iA *iPendant* are still compatible with the R-30iB *iPendant*.

Points	Pixels	Font Size - CSS values	<i>iPendant</i> Controls (TrueFont = 0)
7pt	9px		
7.5pt	10px	1 - xx-small	
8pt	11px		
9pt	12px		<= 15
10pt	13px	2 - x-small	
10.5pt	14px		
11pt	15px		16-17
12pt	16px	3 – small (default)	
13pt	17px		18-23
13.5pt	18px	4 - medium	
14pt	19px		
14.5pt	20px		
15pt	21px		
16pt	22px		
17pt	23px		24-31
18pt	24px	5 – large	
20pt	26px		
22pt	29px		>=32

Points	Pixels	Font Size - CSS values	iPendant Controls (TrueFont = 0)
24pt	32px	6 - x-large	
26pt	35px		
27pt	36px		
28pt	37px		
29pt	38px		
30pt	40px		
32pt	42px		
34pt	45px		
36pt	48px	7 - xx-large	

When TrueFont is 0, then only bold is supported by the *iPendant*. Italic and underline is not supported. Italic will use bold.

When TrueFont is used (non zero), then Bold, Italic, and Underline are supported by the *iPendant*. All Point font sizes are supported, although any size under 10pt is too small to read. When you insert a new *iPendant* Control on your web page, the default is to use TrueFont.

4.2.8 Font Name

The *iPendant* supports the font names defined in the table. The *iPendant* Controls can support any of these fonts. However, if the TrueFont parameter is 0, then the Font Name is ignored and only Courier New, MS Gothic, and SimSun are used based on the current language. This was done so web pages designed with the R-30*iA* *iPendant* are still compatible with the R-30*iB* *iPendant*.

Font Name	Type
Courier New	Monospaced slab serif typeface
MS Gothic	Monospaced east Asian sans-serif typeface
SimSun	Monospaced Chinese serif typeface
Verdana (default)	Sans-serif typeface designed to be readable at small sizes on a computer screen
Tahoma	Similar to Verdana, Tahoma can be used for tighter spacing of numeric data
Times New Roman	Serif typeface
Arial	Contemporary sans-serif typeface
Wingdings	Series of dingbat fonts which render letters as a variety of symbols.
Symbol	Similar to Times New Roman with a selection of commonly used mathematical symbols.

4.2.9 Images

The *iPendant* supports GIF and JPG formats. Generally, GIF images are best used to display small graphics, such as buttons, icons, and banners, or images that contain large blocks of solid colors and little detail. Use the JPG format for images that contain a high level of detail or colors. The Picture Properties dialog box can be used to set the attributes for the image. PNG format is supported except when used in an *iPendant* Control on the PC.

4.2.10 Links

For navigation purposes, the *iPendant* MenuChange and ButtonChange controls can be used to select new pages. The anchor (<a>) tag can also be used. The href attribute defines a source hyperlink. The value of the attribute is the URL of the destination. Typically, the URL should be a relative address, but it is sometimes difficult to know the base address.

A URL can be any file or web page accessible from the robot, a KAREL program or KCL command. Please refer to the FANUC Robot series R-30iA/R-30iA Mate/R-30iB CONTROLLER Ethernet Function OPERATOR'S MANUAL for information on using KAREL or KCL.

NOTE

KAREL (R632) must be installed on the robot controller in order to load KAREL programs in the case North America Setting (R650) is ordered. In the case Standard Setting (R651) is ordered, KAREL (R632) is not needed.

Here are some examples for using URLs.

```
<a href="/KCL/show%20var%20$version">KCL show var $version</a>
<a href="/KCLDO/reset">KCLDO reset</a>
<a href="/KCLDO/set%20port%20tpout[1]=1">KCLDO set port tpout[1]=1</a>
<a href="/mc/demo.stm">Demo</a>
<a href="/KAREL/webtp?tpkey=50">
<a href="/KARELCMD/progname?name1=value1&name2=value2">KAREL Command</a>
```

Using JavaScript:

```
window.location.href = "/KAREL/webtp?tpkey=50"
window.location.href = "/KCLDO/reset";
```

KCL will show a response page. KCLDO will perform the command without a response. Typically a KAREL program must return a response page. If a KAREL program defines a static variable, return_code, and sets it to 204, then the command will be performed without a response.

The Browser menu has BACK and FORWARD function keys. It keeps a history of URLs and can replay them. If you use /KARELCMD/ or /KCLDO/, then the Browser will not keep a history of the URL so it cannot be accidentally replayed by the BACK and FORWARD keys. In fact, the only difference between using /KAREL/ and /KARELCMD/ is that /KAREL/ will store the URL in the history and /KARELCMD/ will not.

4.2.11 Forms

Forms are very useful when you don't need to monitor data from the controller. JavaScript can be used with form elements making them very powerful. The iPendant allows you to remove the form tags and keep just the buttons and text boxes. You can remove the Submit and Reset buttons from the form if they are not required.

You can create a web page that can pass parameters from a form in the browser to a KAREL program. The KAREL program is invoked based on the "submit" action in the form and parameters included in the form are passed with the URL. The FANUC Robotics web server complies with standards found in the HTTP 1.0 Specification. Note that only the HTTP "GET" method is supported at this time. The KAREL program must declare string variables whose names match any parameter names being passed from the form in order to access it. An additional string variable called "URL" should be declared to see the complete URL request sent from the browser to the KAREL Program(for debugging). For example:

```
<FORM ACTION="/KAREL/pnlsvr" METHOD="GET">
  <INPUT TYPE="hidden" NAME="object" VALUE="dout">
  <INPUT TYPE="hidden" NAME="operate" VALUE="set">
  <INPUT TYPE="hidden" NAME="index" VALUE="1">
  <INPUT TYPE="hidden" NAME="value" VALUE="on">
  <INPUT TYPE="submit" VALUE="SET DOUT[1]=ON"></FORM>
```

See Form example in Appendix B

4.2.12 Frames

Frames should be avoided if possible. The *iPendant* does support frames but they can be difficult to navigate without a Touch Panel. Reserved target names of **_blank** and **_top** should never be used since they will force the *iPendant* to log off the controller. (See below for more information) Floating frames (IFRAME tags) are supported.

If the *iPendant* inadvertently browses to a web page that contains a target name of **_top**, then it will cause the *iPendant* to “logout” of the robot. For instance, browsing to this web page and pressing ENTER on the link would cause a logout.

```
<html>
<body>
<center>
<a href="dologout.htm" target="_top">Press ENTER to Logout</a>
</center>
</body>
</html>
```

Once a logout occurs, you can continue to browse web pages using the keys listed in the table below. The Status LED's, Status Window, Override Display, and Function keys will no longer be available. Most Keys on the *iPendant* will be disabled. HOLD, Teach Pendant Enable, E-STOP and Deadman switches will continue to function. Pressing the MENU key will log you back into the robot, however this may take several seconds.

Available Keys	
PREV	Back one Web Page
NEXT	Forward on Web Page
MENU	Login to the robot and resume <i>iPendant</i> Operation
ENTER	Select Link
CURSOR KEYS	Navigate Web Page Links

4.2.13 Themes and Styles

SharePoint Designer 2007 Themes and Styles are supported in *iPendant*.

4.2.14 Ajax

Ajax (an acronym for Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, the controller asynchronously (in the background) without interfering with the display and behavior of the existing page. Data is retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed (JSON and JavaScript is often used instead), and the requests do not need to be asynchronous.

HTML and CSS can be used in combination to mark up and style information. The DOM (Document Object Model) is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

See Appendix B, Ajax Example for more details.

4.2.15 jQuery

jQuery is a fast and concise JavaScript library that simplifies HTML document traversing, event handling, and Ajax interactions. The controller can serve the jQuery library. Use the following include file to use jQuery:

```
<script src='/frh/jquery/js/jquery.js'></script>
```

jQuery UI is a JavaScript library that provides interactions and widgets for developing user interface. A custom download was created from <http://jqueryui.com/download>

All Components
UI lightness Theme
Version 1.8.11

Use the following include files to use jQuery UI:

```
<link rel='stylesheet' type='text/css' href='/frh/jquery/css/jquery-ui.css'>
<script src='/frh/jquery/js/jquery.js'></script>
<script src='/frh/jquery/js/jquery-ui.js'></script>
```

There is much documentation on the Internet about jQuery and jQuery UI.

NOTE

The *iPendant* uses Internet Explorer Embedded which does not have a fast JavaScript engine. Limiting the use of JavaScript is advised. The *iPendant* Color Setup web page uses jQuery UI so you can get an idea of the speed.

4.2.16 Scripting Elements

SharePoint Designer 2007 uses the Microsoft Script Editor available from Tools | Macros menu to add scripts directly into your pages. Only JavaScript is supported on *iPendant*. Client-side scripts are supported. Server-side scripts are not. JavaScript functions should be placed within the header of your Web page. JavaScript placed within the body will be processed top to bottom. Sometimes it is necessary to perform different scripts based on the browser. To determine if the browser is *iPendant* use the following:

```
var _ip = (navigator.userAgent.indexOf("IEMobile") >= 0);
if (_ip == true) {
    // iPendant browser
}
else {
    // Other browsers
}
```

The FANUC Robotics Server Side Include (SSI) directives are supported on *iPendant*. This provides dynamic information only when the page is displayed. Such information can include the current value of a program variable (part count, for example), current status of an I/O point, or the current error listing. Please refer to the FANUC Robot series R-30*iA*/R-30*iA* Mate/R-30*iB* CONTROLLER Ethernet Function OPERATOR'S MANUAL for information on using SSI.

4.2.16.1 Adding script

The Microsoft Script editor includes features that help you create scripts. You can create event handlers for elements on the page, which are scripts that run in response to actions such as when a user chooses a button, when a document first loads, or other events. You can also use editor features to create standalone script blocks to contain any script, not only event handlers.

To set the **DefaultClientScript** property in the Properties window:

1. Open the page for which you want to set the **DefaultClientScript** property.
2. In the drop-down list at the top of the **Properties** window, select **DOCUMENT**.
3. In the **Document** section of the **Properties** window, click the **DefaultClientScript** property, and then select JavaScript for client event handlers.

If you are allowed to use regedit, you can change the default by searching for VBScript (Match whole string only) and setting the value of DefClientLanguage to JavaScript. It may be in several places.

The Script Outline window displays a tree view containing the object hierarchy for the client. Each script on the page appears as a node on the tree. Beneath each object, the hierarchy also contains a list of events supported by that object. If a handler exists for that event, the name of the event is displayed in bold. To create a new handler, double-click the name of the event. To jump to an existing script, click its node in the tree.

When you double click the event name, the editor performs the following actions:

- Creates or moves to one of the following script blocks at the top of the document, depending on where the script will run and what language it will be in:
- Inserts a new, blank event-handling procedure for the element and event you specified.
- If the script will be in JavaScript, adds an event attribute (for example, `onclick=`) to the element.
- Positions the insertion point at the second line of the new script, ready for you to enter commands.

For JavaScript functions, the format is:

```
function elementID_event() {
}
```

When creating JavaScript event handlers, the editor also adds the following attributes to the HTML element itself:

```
event="return elementID_event()"
```

If you are writing script, the HTML editor uses IntelliSense[®] — it displays options that help you complete statements. When you type in the name of an object available on your page followed by a period (.), the editor displays all members of that object's class.



4.2.16.2 Debugging in source view



Source view enables you to execute debugger commands, such as setting breakpoints, by choosing commands from the Debug menu or the Debug toolbar. The left margin of the edit window displays glyphs indicating breakpoints.

When the debugger is running, the current page is displayed in Source view so you can see individual lines of script. The current line is indicated in the margin with an arrow indicator.

4.2.16.3 Debugging in internet explorer

While viewing your page in Internet Explorer on a PC, you can press F12. This brings up the Developer Tools. You can select Script, choose the web page, and press Start debugging. You can set breakpoints and watch variables. You have all the capabilities of a source code debugger.

4.2.16.4 Scripting iPendant controls

You may create objects using JavaScript but then you cannot use the WYSIWYG editing capabilities in SharePoint Designer 2007.

Here is an example of creating an object using JavaScript so the DataIndex can be calculated at run-time:

```
<SCRIPT LANGUAGE="JavaScript"><!--
```

```

var _reqvar = <!--#echo var = $BBSTART_DI.$PORT_NUM -->
_reqvar = _reqvar + 2
req = '<object classid="clsid:7106066C-0E45-11D3-81B6-0000E206D65E" id="FRIPToggleButton1"
width="87" height="50">¥n'
req += ' <param name="_Version" value="65536">¥n'
req += ' <param name="_ExtentX" value="2302">¥n'
req += ' <param name="_ExtentY" value="1323">¥n'
req += ' <param name="_StockProps" value="15">¥n'
req += ' <param name="Caption" value="+X">¥n'
req += ' <param name="ForeColor" value="0">¥n'
req += ' <param name="BackColor" value="16776960">¥n'
req += ' <param name="DataType" value="1">¥n'
req += ' <param name="DataIndex" value="" + _reqvar + "">¥n'
req += ' <param name="Border" value="4">¥n'
req += ' <param name="Type" value="0">¥n'
req += ' <param name="ViewType" value="0">¥n'
req += ' <param name="TrueColor" value="12632256">¥n'
req += ' <param name="FalseColor" value="16776960">¥n'
req += ' <param name="TrueStrColor" value="0">¥n'
req += ' <param name="FalseStrColor" value="0">¥n'
req += ' <param name="TrueValue" value="0">¥n'
req += ' <param name="FalseValue" value="0">¥n'
req += ' <param name="TrueImage" value>¥n'
req += ' <param name="FalseImage" value>¥n'
req += ' <param name="OtherPhase" value="-1">¥n'
req += '</object>'
document.write(req);
//--></SCRIPT>

```

Internet Explorer views the *iPendant* ActiveX controls as objects and allows you to get and set the properties using JavaScript. Some of the *iPendant* Controls allow you to set the properties using JavaScript. The width and height is not settable at this time. The get value property is not available at this time. We plan to add additional JavaScript support in the future. Some properties that are working:

```

document.Label1.ForeColor = "0";
document.Label1.BackColor = "255";
document.Label1.Caption = document.form1.button1.value;
document.Label1.Border = "2";
document.Label1.HAlign = "2";
document.Label1.VAlign = "2";
document.Label1.Font.Name = "Courier New";
document.Label1.Font.Size = 12;

```

```

document.Label2.ForeColor = "16777215";
document.Label2.BackColor = "16711680";
document.Label2.Caption = "";
document.Label2.DataIndex = "$TP_DEFPROG";
document.Label2.DataType = "102";

```

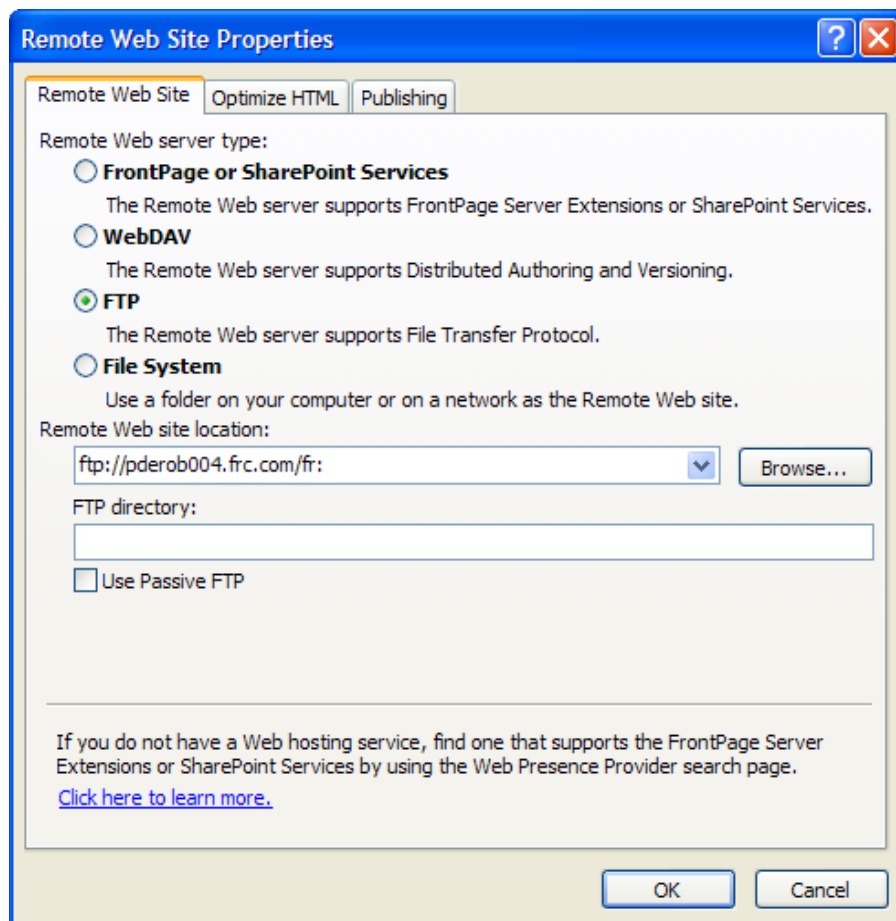
4.2.17 DOM Elements

The *iPendant* supports the Document Object Model for standard HTML elements. Here are some examples:

Element	JavaScript
URL	window.location.href
Button in Form	Document.form1.button1.value
Text in Form	Document.form1.text1.value
Selected Option in Form	document.form1.list1.options[document.form1.list1.selectedIndex].text
Button in Frame "User1" as seen from Frame "User2"	window.parent.user1.document.form1.button1.value

4.3 PUBLISHING YOUR WEB

SharePoint Designer 2007 allows you to publish your web to the robot. File | Publish Site brings up a dialog box. The Publish Site dialog box allows you to FTP your files to the robot. Your robot must have an FTP server running. Please refer to the FANUC Robot series R-30iA/R-30iA Mate/R-30iB CONTROLLER Ethernet Function OPERATOR'S MANUAL for information on using FTP.

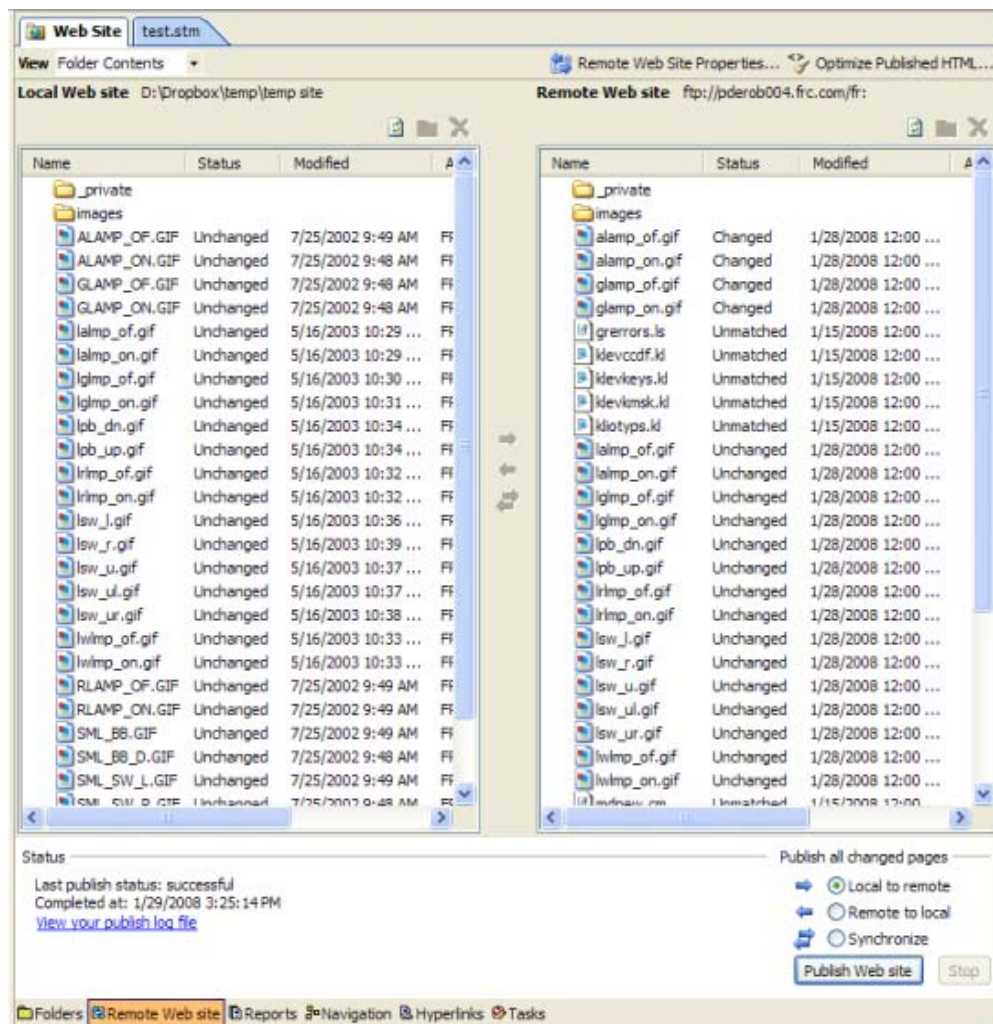


Select FTP as the server type and use ftp://robot_ip_addr/fr: to publish files on the FR: device as shown below:

NOTE

The robot does not support subdirectory creation from FTP so you should only publish files in the web's current folder. If SharePoint Designer 2007 is trying to publish files created in another subdirectory such as _derived, you will need to remove these files.

Once you click OK the Web Site tab will be displayed and SharePoint Designer 2007 will attempt to connect to your robot. After a successful connection is established you can choose which files to download to the robot.



See the SharePoint Designer 2007 Help for additional information on transferring and synchronizing files.

5

MAKING A CUSTOM *iPendant* SCREEN USING THE *iPendant* CONTROLS

The easiest way to create an easy operator panel or custom screen for the *iPendant* is by putting the *iPendant* controls on the web page and by setting their properties.

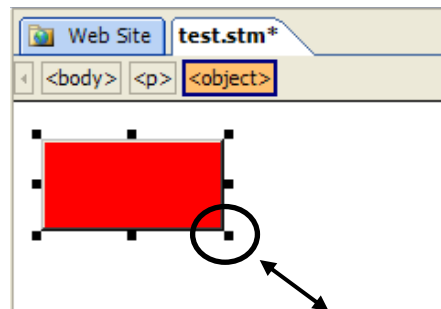
5.1 CONTROL ARRANGEMENT

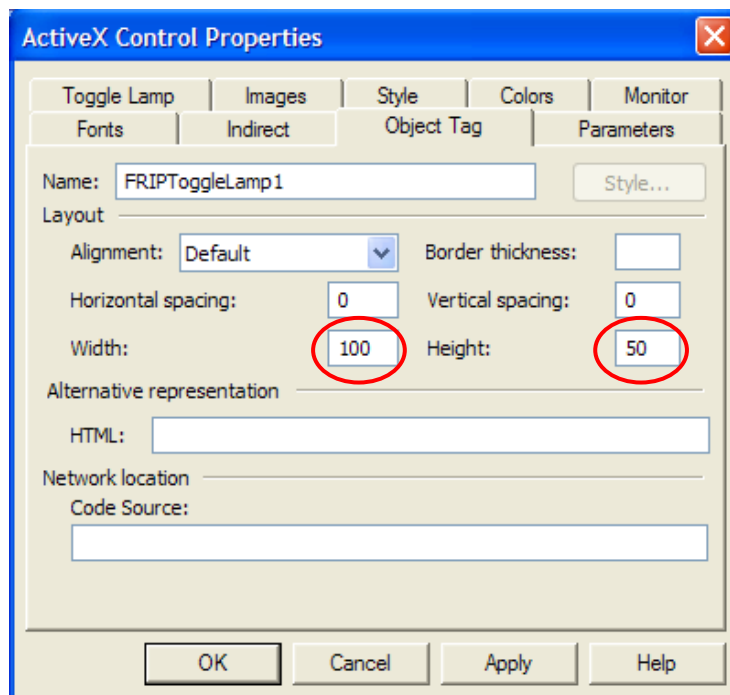
If you haven't already added "ActiveX Control" onto the Toolbar, follow these steps. Refer to the subsection 3.3.1. You only need to do this once:

1. Select View | Toolbars | Customize from the menu bar.
2. From the Commands tab, select Insert category, cursor to ActiveX Control
3. With the left mouse, drag the command out of the dialog box to the Toolbar

To add an *iPendant* Control to your web in SharePoint Designer 2007, follow these steps:

1. Position your cursor where you want the control to appear.
2. Select ActiveX Control from the Toolbar.
3. Choose the control you want to insert from the list of available FANUC *iPendant* controls, and click OK. The selected control is now inserted into your page.
4. Configure its properties by double-clicking on the control. The property dialog box that appears will depend on the control you have inserted.
5. You can resize the control by grabbing one of the handles and moving it.
6. On the Object Tag property page for the control you can reset the size to exactly what you want.





5.2 COMMON CONTROL PROPERTIES

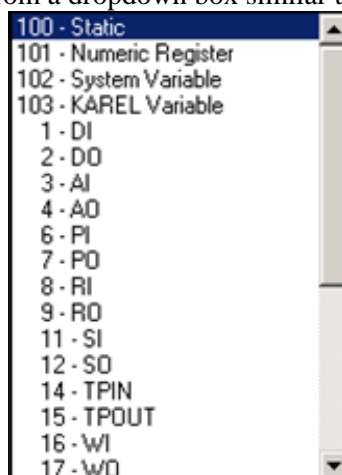
Most of the controls have the following common properties.

5.2.1 Object Tag

The Object Tag dialog allows you to specify some standard attributes associated with your control. The Name is used when an error occurs. The Width and Height can be specified in pixels. Of course, you can resize the control by dragging the control's handles with the mouse.

5.2.2 DataType and DataIndex

DataType Specifies the type of the data to be monitored or modified depending on the type of control. The allowable types can be selected from a dropdown box similar to the one below:



Where:

100 – Static Displays the strings specified in Caption.

101 – Numeric Register Displays the value of the numeric register specified in DataIndex.

- 102 – System Variable Displays the value of the System Variable specified in DataIndex.
- 103 – KAREL Variable Displays the value of the KAREL Variable specified in DataIndex.
- 104 – Dictionary Element Displays the dictionary element specified in DataIndex.
- 112 – String Register Displays the value of the string register specified in DataIndex.
 - 1 – DI Displays the value of DI specified in DataIndex.
 - 2 – DO Displays the value of DO specified in DataIndex.
 - ... Displays the value of the I/O type specified in DataType and DataIndex.

DataIndex Specifies the number or the variable name associated with the DataType.

For numeric and string registers, this is the register number. For example, 2.

For System Variables, the type must be Integer, Real, Boolean, Short, Byte, or String. For example, \$MNUTOOLNUM[1]

For KAREL Variables, enclose the program name inside [...]. The type must be Integer, Real, Boolean, Short, Byte, or String. For example,

```
[USEREXT]STR_VAR
[USEREXT]STRUC_VAR.FIELD1
```

For Dictionary Elements, specify the dictionary name and enclose the dictionary element inside [...]. For example,

```
TPAR[5]
```

For I/O, specify the port number. For example, 5

5.2.3 Images

The Images dialog allows you to select the images to display when the ViewType is Image.

- The display size is automatically adjusted to the size of the image which is specified by FalseImage or DataDefault for the Multi Control. Therefore the size of any other images specified should be the same size as FalseImage. It is not possible to change the size using the mouse.
- It is necessary to copy all the image files which are specified to the directory on the iPendant where the web page is located.
- If iPendant cannot find an image file, “No Image File” is displayed on the control.
- SharePoint Designer 2007 may not be able to find the Image files after the page is closed and reopened. You can specify the directory containing the image files by setting a Registry String. An example is shown below.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\FANUC\FANUC Robotics iPendant Controls]
"CurrentDirectory"="v:\qa\ipendant\cgop"
```

Or you can reselect one of the images from the Property Page dialog box. The ActiveX Controls will set this registry key for you. You may need to select View | Refresh to refresh the web page.

- Transparent gif images are supported. The color that shows behind the image must be specified as follows:

For CommandButton, ButtonChange, MenuChange, ToggleButton, and ToggleLamp:
 FalseColor is used when the image is not pressed.
 TrueColor is used when the image is pressed.

BackColor should be set to the background color of the web page.

For Multi Control:

BackColor is used.

5.2.4 Border

Border Select the border design of the control out of the types shown below.



Where:

0 - Thin3D Create a thin 3D line.

1 - None No border line.

2 - Black Create a thin black line.

3 - ForeColor Create a thin line whose color is the equal to the foreground color (color used for displaying characters).

4 - Bold3D Create a bold 3D line.

5.2.5 Colors

The Colors dialog allows you to specify the color of certain elements. The iPendant supports 256 colors. All controls have the following Colors associated with them:

ForeColor Specify the color of characters.

BackColor Specify the background color.

Some controls will have additional choices such as TrueColor and BackColor. (See the individual controls for additional details)

5.2.6 Fonts

The Fonts dialog allows you to specify the font used with the control. The iPendant supports a subset of the Fonts available in SharePoint Designer 2007. Please see section 4.2.7 Font Size and 4.2.8 Font Name for more details.

By default, the **TrueFont** parameter is set (non zero). With this setting, the EasyPanel controls are designed to be compatible with R-30iB iPendant which can support Bold, Italic, Underline, and any font size. The font shown in SharePoint Designer 2007 is the same as what you will see on the iPendant provided you use a Font Name that is supported by the iPendant.

If you want the fonts to be compatible with R-30iA iPendant, then set the **TrueFont** parameter to 0 and use the following font settings:

Tag	Value
Font:	Courier New
Font Style:	Regular or Bold
Size:	14, 16, 18, or 24
Strikeout	Do not check
Underline	Do not check

The EasyPanel controls may convert 16 down to 15.75. If this happens, use 16.5.

5.2.7 Alignment

HAlign Selects the horizontal alignment of characters out of the types shown below.

- 0 - Left** Left align the text.
- 1 - Center** Center the text.
- 2 - Right** Right align the text.

VAlign Selects the vertical alignment of characters out of the types shown below.

- 0 - Top** Top align the text.
- 1 - Center** Center the text.
- 2 - Bottom** Bottom align the text.

5.2.8 Monitor

The Monitor dialog allows you to specify whether the item specified in the DataType field is Monitored or Updated Periodically and the time interval in milliseconds used to monitor or update the data. If the Periodic checkbox is not checked (FALSE) the data will be monitored at the specified rate and the current value will only be sent to the iPendant if the value has changed since the last period. If the Periodic checkbox is checked (TRUE) the value of the item specified in the DataType field will be sent to the iPendant at the interval rate regardless of whether or not it has changed since the last update. Monitoring the data is more efficient because the data is only sent from the robot to the iPendant when the data has changed.

The interval time will default to 250 ms. The minimum interval time is 100 ms. The periodic switch will default to unchecked (FALSE).

Identical monitors for the iPendant are shared, even across multiple pages. The lowest interval time is used. For instance, if the page in the left window is monitoring DI[1] at 250 ms and the page in the right window starts monitoring DI[1] at 100 ms, then the left page will also monitor at 100 ms. If the right page is changed to another page, the left page will continue to monitor at 100 ms until the page is changed.

The Monitor dialog allows you to specify FastLoad:

Checked Specifies that properties that are set to their corresponding “default” values are not included in the web page. This effectively saves load time.

Unchecked: Specifies that all parameters are included in the web page, whether they have “default” values or not.

5.2.9 Function Key ViewType

The following EasyPanel Controls will support the Function Key ViewType:

- EditBox Control
- CommandButton Control
- ToggleButton Control
- ButtonChange Control
- MenuChange Control
- ComboBox Control

The ViewType will display a dropdown box with F2 – F5 and F7 – F10 as selections. F1 and F6 are not available because the controller reserves them for the [TYPE] function key. If a function key is selected, then the EasyPanel control will be invisible, but it will register itself with the FunctionKey control. The EasyPanel control can reside anywhere on the visible portion of the web page. If it is not on the visible portion (scroll bars are necessary to see it) then it may not be created. It is only operational if it has been created. The control can be any size as long as it is big enough to be created. The size doesn't matter since it is invisible.

To avoid cursoring to the invisible control, use `tabindex="-1"` in the object tag.

Example:

```
<object classid="clsid:71060660-0E45-11D3-81B6-0000E206D650" id="FRIPEditBox1" width="116"
height="50" tabindex="-1">
  <param name="Caption" value="F2">
  <param name="ViewType" value="1">
  ...
</object>
```

The button caption and/or button images will be used by the FunctionKey control. Both TrueImage and FalseImage are used. The Image, if specified, is displayed first; and the Caption, if specified, is displayed next. Properties that are ignored include most of the display properties:

- Font
- Halign
- VAlign
- Border

All FunctionKey controls will use their color properties, such as ForeColor and BackColor.

5.2.10 Caption

If the DataType value is other than 100 – Static, the string in the caption is managed as a format string. The “%v” in the string is converted as the value of the specified variable. When the format string is NULL, only the value of the specified variable is displayed. Any C Style Format specifier may also be used instead of “%v”. In the example below, the left side shows the result and the “fmt:” shows the format string that was specified in the Caption.

FORMAT SPECIFIERS	
R[1] : 0 ms	fmt: R[1] : %v ms
DO[1] : OFF	fmt: DO[1] : %v
GO[1] : 0	fmt: GO[1] : %u
1. 8000E+000	fmt: %20. 4E, value: [userext]real_var
1. 8000e+000	fmt: % -20. 4e, value: [userext]real_var
Percent % CGOP1	fmt: Percent %% %s, value: \$TP_DEFPROG
CGOP1	fmt: % -20s, value: \$TP_DEFPROG
CGOP1	fmt: %20s, value: \$TP_DEFPROG

If the Caption contains multiple words and it doesn't fit into the available columns, it is automatically split into multiple rows. If the automatic split is not appropriate, the Caption can be manually split into multiple rows by adding `¥n` where appropriate. You may use `%n` anywhere in the Caption to suppress the value and show only the Caption.

5.2.11 Pulse DO

The feature allows the user to pulse a digital output port for a specified number of milliseconds. In SharePoint Designer 2007, if the user selects `DataType = DO` for `CommandButton` or `ToggleButton`, then the Pulse Checkbox will be enabled. Once the Pulse checkbox is clicked, the ms textbox is enabled.

☐ Pulse Time: ms

If Time is zero, no pulse will occur. A pulse always writes the value in `SetValue` and then after the time has expired, writes the reverse. So if the port is "normally off", `SetValue` can be set to `True` to pulse it on. If the port is "normally on", `SetValue` can be set to `False` to pulse it off.

The maximum pulse value is 32767 ms. The default is 100 ms.

5.2.12 Port Simulation

This feature allows the user to display and modify the port simulation status for an I/O signal. In SharePoint Designer 2007, if the user selects an I/O type for `DataType` that supports simulation, then the Simulation Checkbox will be enabled.

☒ Sim Status

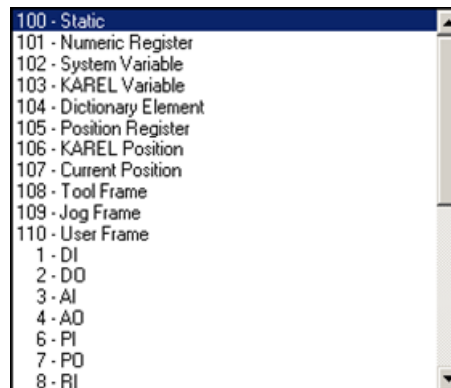
The Port Simulation Type will be similar to the Boolean type. It automatically displays `SIMULATE`, `UNSIMULATE` but you can specify the string precision in the Caption field. So if `Caption = "%.1S"` (note precision is after the period), then only "S" or "U" will be displayed.

If "SIM" or "UNSIM" is desired, then the ComboBox Control or Multi Control can be used instead. These controls allow you to specify whatever strings you want displayed. ComboBox is used to both monitor and modify the simulation status while Multi is used to only monitor the simulation status.

5.2.13 Positions

The Label Control allows positions to be displayed. The CommandButton Control allows Position Register and KAREL Positions to be recorded.

DataType Specifies the type of the position data to be monitored or recorded:



The allowable types can be selected from a dropdown box similar to the one above:

Where:

105 – Position Register Displays or records the value of the Position Register specified in DataIndex.

106 – KAREL Position Displays or records the value of the KAREL Position specified in DataIndex.

107 – Current Position Displays the value of the Current Position in the representation specified in DataIndex.

108 – Tool Frame Displays the value of the Tool Frame,
MNUTOOL[GroupNum, PosId], whose PosId is specified in DataIndex.

109 – Jog Frame Displays the value of the Jog Frame,
[TPFDEF]JOGFRAMES[GroupNum, PosId], whose PosId is specified in DataIndex.

110 – User Frame Displays the value of the User Frame,
\$MNUFRAME[GroupNum, PosId], whose PosId is specified in DataIndex.

DataIndex Specifies the number or the variable name associated with the DataType.

For Position Registers, Tool, Jog, and User Frames, the position id is specified.

For KAREL Positions, enclose the program name inside [...].

[USEREXT]POSN

[USEREXT]XYZEXT

[*SYSTEM*]\$GROUP[1].\$UFRAME

For Current Position, the representation is specified. JOINT is the default.

JOINT

WORLD

USER

GroupNum Specifies the group number associated with the DataType/DataIndex if more than one group is available on the robot controller. Not used for KAREL Positions which are inherently defined for a single group.

PosAlign Selects the position alignment from the types shown below.

0 - Default Align the position similar to Position Menu.

1 - Subwindow Align the position similar to TPP and Position Register subwindows.

2 - Vertical Align the position vertically.

The other properties of the label control are also used, such as Fonts, Border, Halign, Valign, and Colors.

Here are some examples of the current position with different alignments:

Default - Left/Top								
JOINT			GP1			UT:1		
J1:	31.138	J2:	16.253	J3:	15.989			
J4:	23.306	J5:	20.498	J6:	28.325			
E1:	8.280	E2:	44.064	E3:	6.653			

Subwindow - Centered								
WORLD			GP1			UT:1		
								CONF:UT 0
X	1110.000	mm	W	180.000	deg			
Y	0.000	mm	P	0.000	mm			
Z	620.000	mm	R	0.000	deg			
E1	3.758	mm	E2	59.616	mm			
E3	2.110	mm						

Vertical- Right/Bottom				
USER	GP1	UF:1	UT:1	
G:F	U T,	0,	0,	0
X:	1055.540		mm	
Y:	-994.081		mm	
Z:	4176.247		mm	
W:	-7.671		deg	
P:	-38.192		deg	
R:	42.969		deg	
E1:	0.769		deg	
E2:	474.408		mm	
E3:	6.653		mm	

All position types have a default title. However, you can override the default title by specifying the **Caption** parameter. Each **PosAlign** is different:

Default Truncates the caption to 19 characters

Subwindow Truncates the caption to 5 characters

Vertical Uses the entire caption. You can split the caption using %n or the caption will automatically split to fit in the available space.

The comment for Position Registers and Frame Positions is shown on a separate line.

```

PR[1]                GP1
[Perch Position]
CONF:F U T, 0, 0, 0
X: 2277.512 Y: 270.391 Z: 3913.003
W: -6.627 P: -5.219 R: 36.404
E1: 12.960 E2: 3.211 E3: 4.694

```

The default title for a KAREL position is its name. It will be shown on the comment line instead of the top line.

```

                                GP1
[tscvar]xyz
CONF:F U T, 0, 0, 0
X: 1874.370 Y: 480.508 Z: 1401.627
W: -59.703 P: -56.915 R: -98.003

```

The **TrueFont** parameter is disabled for position display.

5.2.14 Indirect DataType and DataIndex

Five different Indirect DataType/DataIndex keywords can be used in any EasyPanel parameter that defines a String. The special keywords !INDIRECT1, !INDIRECT2, ... !INDIRECT5 will be used to specify indirection. The indirect value is read as a string and substituted into the parameter.

- Multiple keywords can be used in a parameter.
DataIndex: \$MCR_GRP[!INDIRECT1].!INDIRECT2
- Multiple parameters can contain keywords.
DataIndex: \$MCR_GRP[!INDIRECT1].\$DRY_JOG_OVR
Caption: !INDIRECT2
- A keyword can be used multiple times.
DataType: 101 – Numeric Register
DataIndex: !INDIRECT1
Caption: !INDIRECT1
- Indirect keyword is allowed when specifying Indirect DataType/DataIndex.
Indirect2DataType: 101 – Numeric Register
Indirect2DataIndex: !INDIRECT1

In SharePoint Designer 2007, the Indirect Property tab can be used to set up the Indirect DataType/DataIndex properties. The indirection is evaluated each time the page is loaded. The Indirect DataType/DataIndex properties are monitored so they will change dynamically.

An example of indirection in the EditBox Control:

The KAREL program, indirect.kl, contains a variable, regno: INTEGER, which is used to determine the register number to display.

```

<param name="Caption" value="R[!Indirect1] : %v">
<param name="DataType" value="101">
<param name="DataIndex" value="!Indirect1">
<param name="Indirect1DataType" value="103">
<param name="Indirect1DataIndex" value="[indirect]regno">

```

5.2.15 PANEID

The special keyword !PANEID can be used in any EasyPanel parameter that defines a String. The proper value is substituted into the parameter according to the table below:

!PANEID	Description
1	Primary iPendant pane
2	Dual iPendant pane
3	Third iPendant pane
4 – 9	Panes for Internet Explorer connections

An example of indirection in the EditBox Control:

The KAREL program, indirect.kl, contains an array variable, regno: ARRAY[9] OF INTEGER, which is used to determine the register number to display for each pane.

```
<param name="Caption" value="Pane Id: !paneid, R[!Indirect1] : %v">
<param name="DataType" value="101">
<param name="DataIndex" value="!Indirect1">
<param name="Indirect1DataType" value="103">
<param name="Indirect1DataIndex" value="[indirect]regno[!paneid]">
```

5.2.16 Pipe

Properly configured iPendant controls will actively monitor and modify controller information and processes without you having to write code on the controller. There are times, however, where you might require additional flexibility and control. The Pipe mechanism provides this ability.

If you configure an iPendant control's Pipe parameter to have a value of a filename,

```
<param name="Pipe" value="mypipe!paneid.dat">
```

you will be able to send the control commands while it is active. Open the file using the string "PIP:" as its device prefix. Then write string commands to the file in the format:

```
id="control_id" param_name="new_value"
```

where:

control_id is the string you provided in the Object tag Name field
param_name is the name of the parameter you wish to modify.
new_value is the new value you want to assign to the parameter

A KAREL example to change the caption of the label control named Label_1 follows.

```
OPEN FILE f1 ('AP', 'PIP:mypipe1.dat')
WRITE f1 (' Id="Label_1" caption="This is my new caption"')
```

Remember, Label_1 must have been configured to listen to pipe "mypipe!paneid.dat" at design time.

5.2.17 SetFocus

Use the SetFocus parameter to tell the browser to give focus to or remove it from the control. This only applies to the following controls that can accept focus.

- EditText Control
- CommandButton Control
- ToggleButton Control
- ButtonChange Control
- MenuChange Control
- ComboBox Control

SetFocus must be issued through the pipe mechanism.

```
<param name="SetFocus" value="1">
```

Sending a value of “1” directs the browser to give focus to the identified control. Sending a value of “0” sets the focus to the parent HTML page, essentially removing focus from the identified control.

5.2.18 ClickMe

Use the ClickMe parameter to tell the browser to send the action event to the control. This looks to the control exactly as if the user had clicked on it. This only applies to the following controls that can be clicked.

- EditText Control
- CommandButton Control
- ToggleButton Control
- ButtonChange Control
- MenuChange Control
- ComboBox Control

ClickMe must be issued through the pipe mechanism.

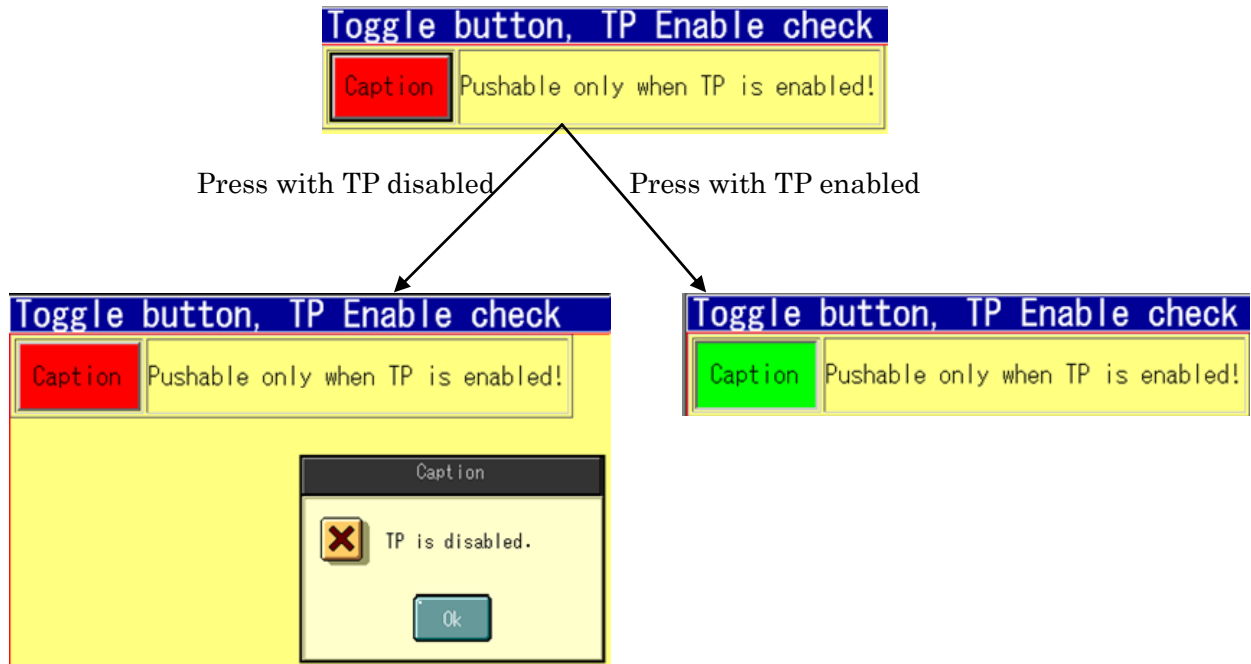
```
<param name="ClickMe" value="1">
```

This feature can be especially useful with the ComboBox in that it will cause the control to display the choices. It will save you from having to write duplicate code to generate the choices under program control that the ComboBox is ready to display.

The value sent has no meaning at this time, but we suggest sending a 1 as other values may have significance in the future.

5.3 CHECK CONDITION

Many iPendant Controls have “Check Condition” tab in their property pages. When check condition properties are specified in the controls, you cannot press the button unless the specified key is pressed or specified I/O signal is ON.



The user can select any combination of the following key:

- Deadman switch
- TP enable switch
- Shift key
- Input/Output signal specified by user

Key and I/O check by this function cannot be used for safety purpose.
This function is just to prevent wrong operation.

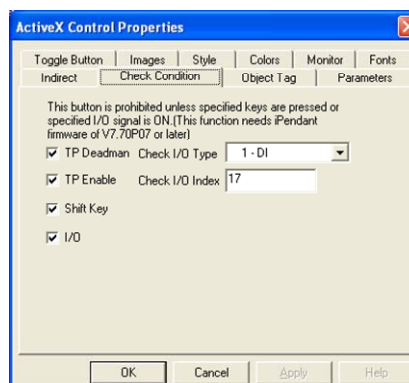
5.3.1 Supported Controls

Following controls will support the functionality:

- EditText
- CommandButton
- ToggleButton
- MenuChange
- ButtonChange
- ComboBox

5.3.2 User Interface

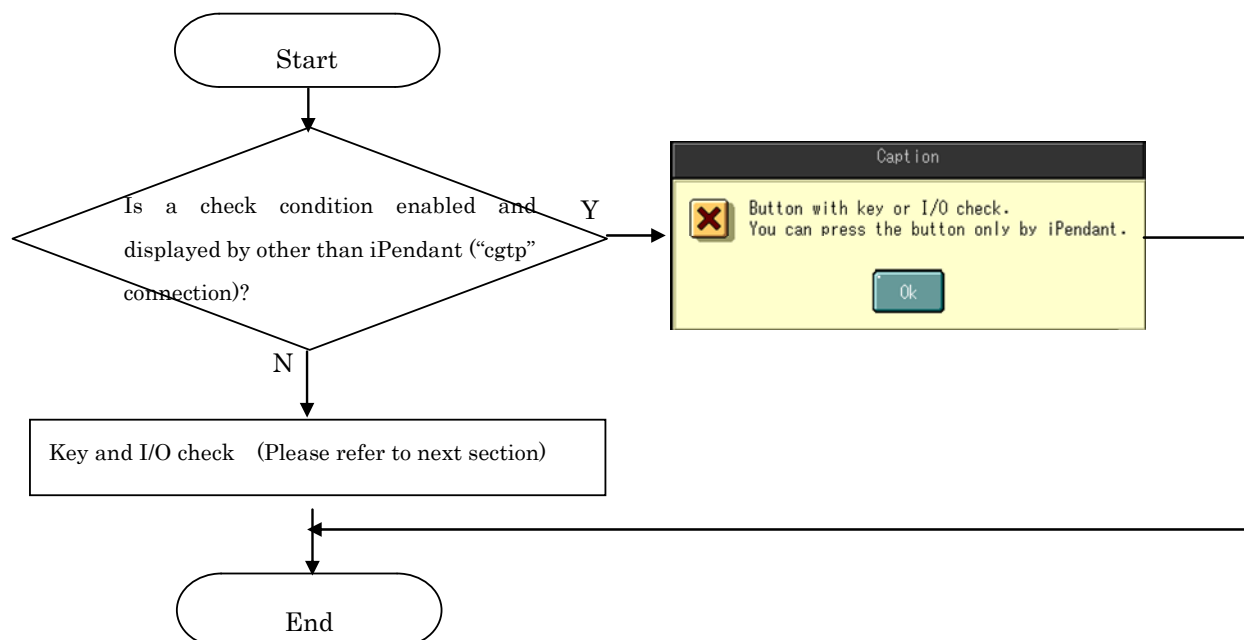
Controls listed in section 5.3.1 have “Check Condition” tab in their property pages.



“Check I/O Type” and “Check I/O Index” are not displayed if “I/O” check box is not checked.
All check boxes are not checked by default.

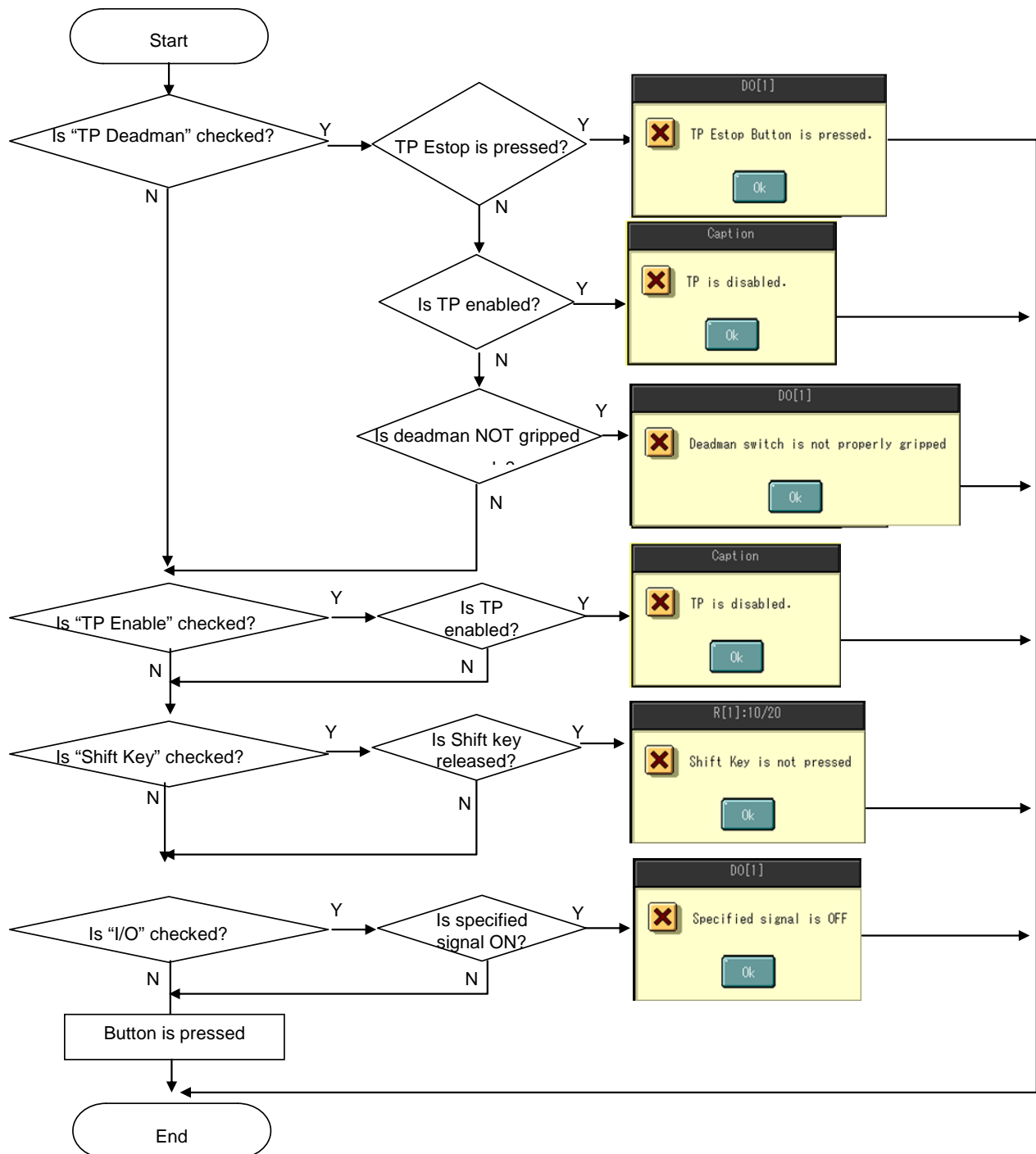
Item	Description
TP Deadman	If checked, the button can be pressed when following conditions are all satisfied. Please note that TP Estop and TP enable switch are checked. 1 Deadman switch is properly gripped. 2 TP Estop is OFF. 3 TP is enabled.
TP Enable	If checked, the button can be pressed only when TP is enabled.
Shift Key	If checked, the button can be pressed only when shift key is pressed.
I/O	If checked, the button can be pressed only when a signal specified by “Check I/O Type” and “Check I/O Index” is ON.
Check I/O Type	This item specifies type of I/O to be checked. This item is shown only when “I/O” is checked.
Check I/O Index	This item specifies index of I/O to be checked. This item is shown only when “I/O” is checked.

5.3.3 Error Message 1



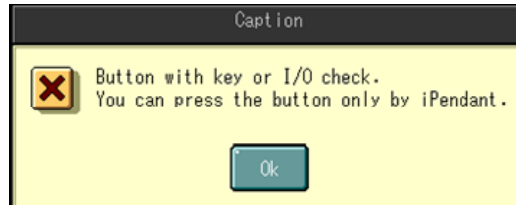
Please refer to section 5.3.5 “Display other than *iPendant*”, too.

5.3.4 Error Message 2



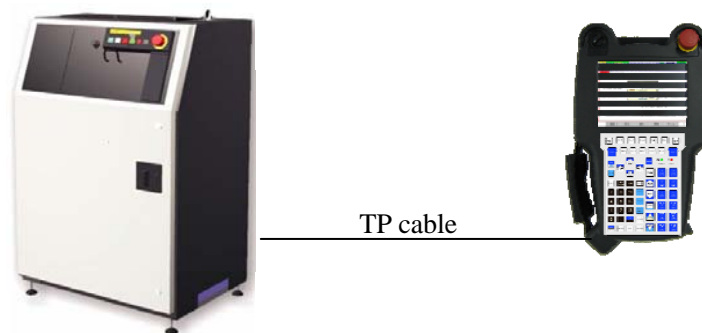
5.3.5 Display other than iPendant

iPendant Controls can be displayed in various ways. If key or I/O check is enabled, you cannot press the button if it is not displayed by iPendant. Following dialog box is displayed.



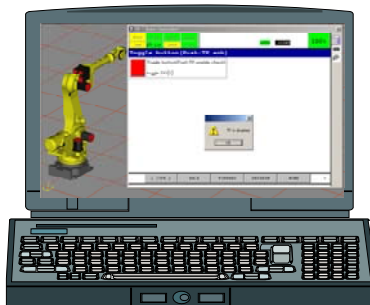
We show examples below.

- A) iPendant displays web page from connected controller



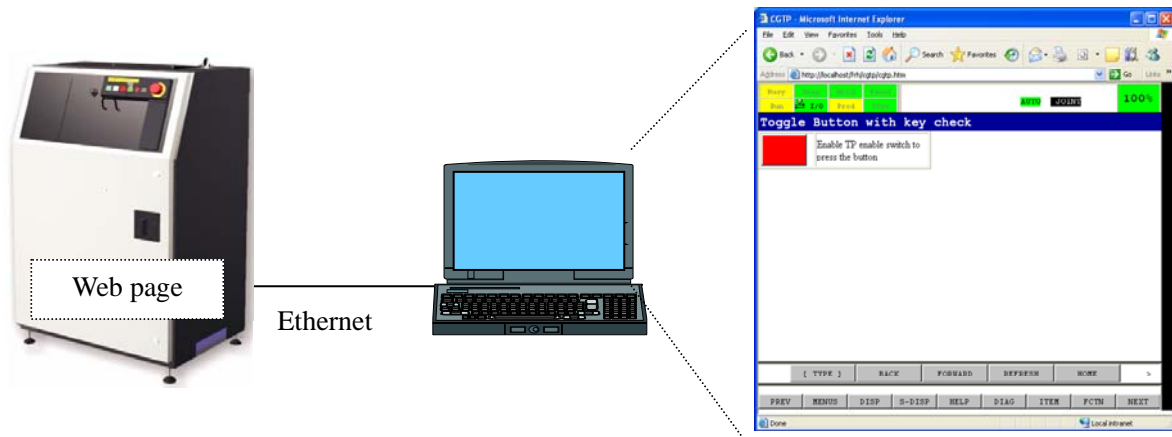
This is the most common way to display a FANUC iPendant Control on a real controller. You can push the button with key or I/O check. Key of real iPendant and I/O of the controller are checked.

- B) ROBOGUIDE connects using Internet Explorer and it is iPendant



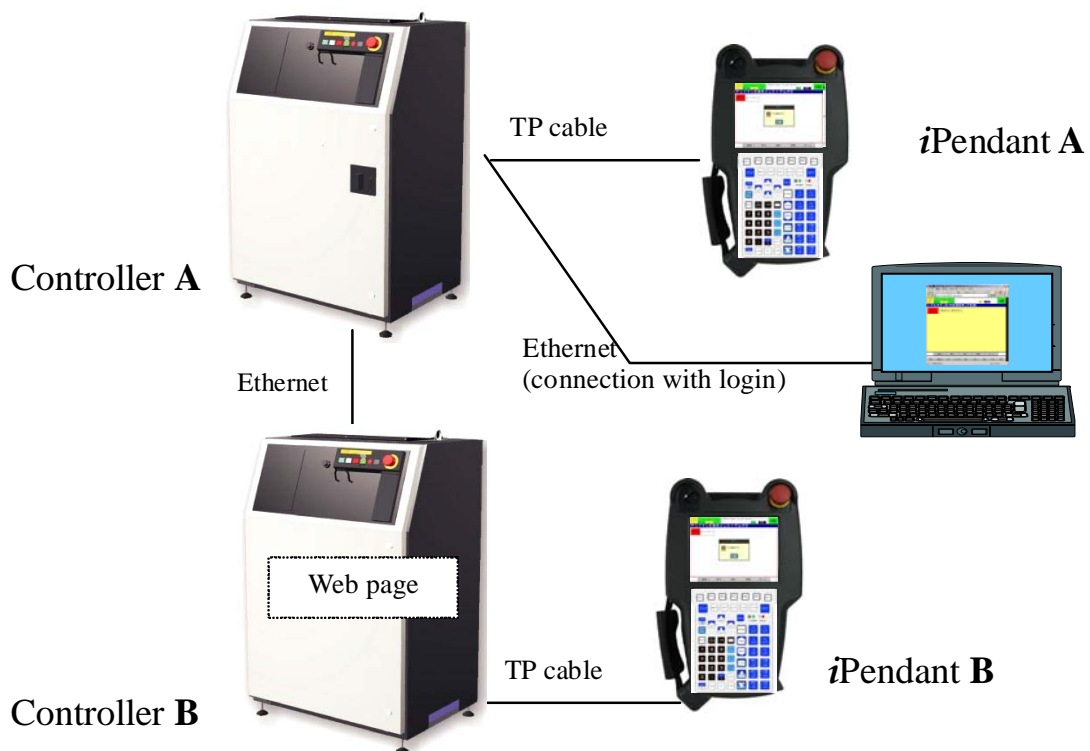
This is the most common way to display a FANUC iPendant Control by ROBOGUIDE. You can push the button with key or I/O check. Key of virtual iPendant and I/O of virtual robot controller are checked.

- C) Internet Explorer connects using Navigate iPendant (CGTP)



Internet Explorer logs into the robot as a CGTP connection. Then Internet Explorer displays web page. You cannot push buttons with key or I/O check.

- D) *iPendant* or Internet Explorer displays web page from remote robot.
For example, *iPendant* A can display web page in controller B.



By *iPendant* A, you can press button with key or I/O check.

TP Deadman, TP Enable, and Shift key: key of *iPendant* A is checked.

I/O : If "Check I/O Type" is TPIN, TPIN of controller A is checked.

Otherwise, I/O of controller B is checked.

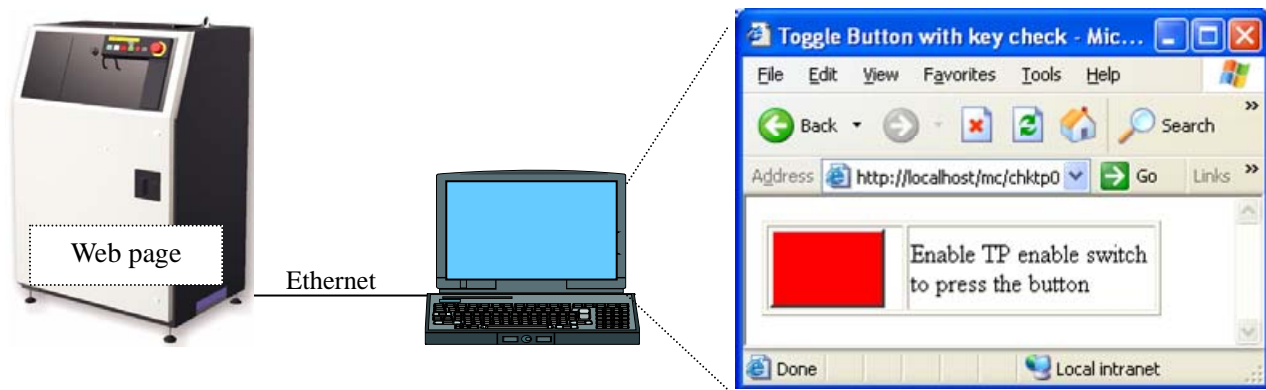
Suppose Internet Explorer log onto controller A. Then display webpage in controller B via controller A.

For example, using link to webpage in favorite screen of controller A.

This is an example of remote display using Internet Explorer.

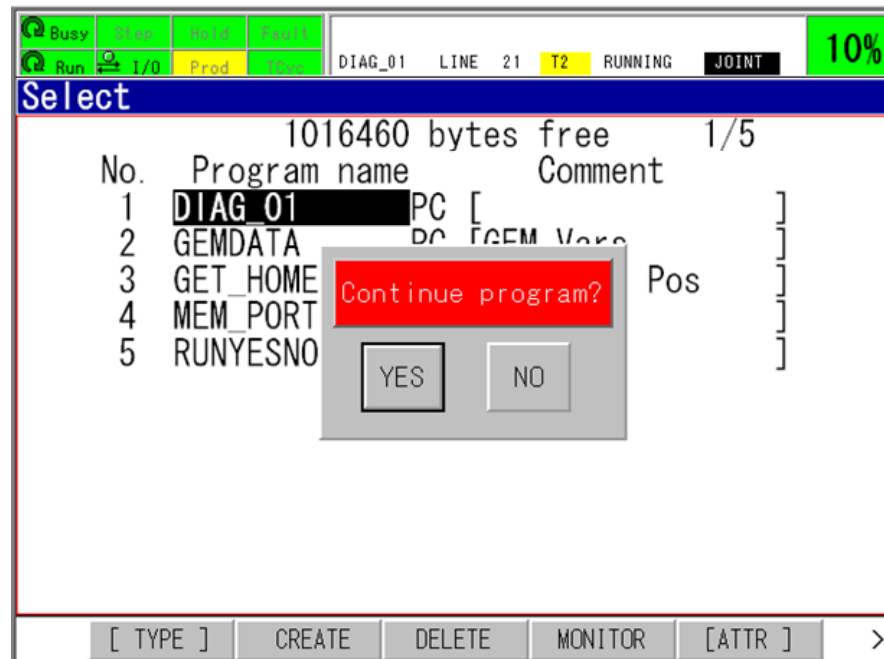
In this case, you cannot push button with key or I/O check by Internet Explorer.

- E) Internet Explorer displays web page from robot directly, without login



Internet Explorer specifies URL of Web page directly, without displaying cgtpl.htm (without login). You cannot push button with key or I/O check.

- F) Display by dialog box



You cannot use this function when you place iPendant Controls on a dialog box, not on web page.

5.3.6 Caution and Limitations

- A) Message Box display of momentary Toggle button
 Message Box is displayed when button is released just like alternate button.
 Message Box is not displayed when button is pressed.
- B) Condition check of momentary toggle button
 Condition is checked both at press and release.
 Suppose there is momentary toggle button with one of condition is checked. The button is now OFF state. Suppose the condition is satisfied when button is pressed and the condition is not satisfied when button is released. The button turns ON when button is pressed. But it does not back to OFF state when button is released. Message Box is displayed.

- C) Condition to monitor must be stable during operation.
Especially you should not specify signal that frequently changes.
- D) If you press button just after status change of condition is checked, key or I/O status may not be recognized correctly.
- E) If deadman switch is not working properly, check of the switch doesn't work. For example, single chain iPendant and Dual chain panel controller should not be used together.
- F) You can display iPendant Controls by Internet Explorer, but you cannot press buttons with key or I/O check. Do not use the function on web page to be displayed on Internet Explorer.

5.3.7 ComboBox Control Limitation

When the ComboBox Control is displayed remotely as in D or without login as in E, it only works in read-only mode. It will display the currently selected value on the button correctly, but it will not respond to any key presses.

5.3.8 Execution Control Limitation

When the Execution Control is used remotely as in D or without login as in E, it will run and abort the KAREL program properly but will not set any parameters nor send any events.

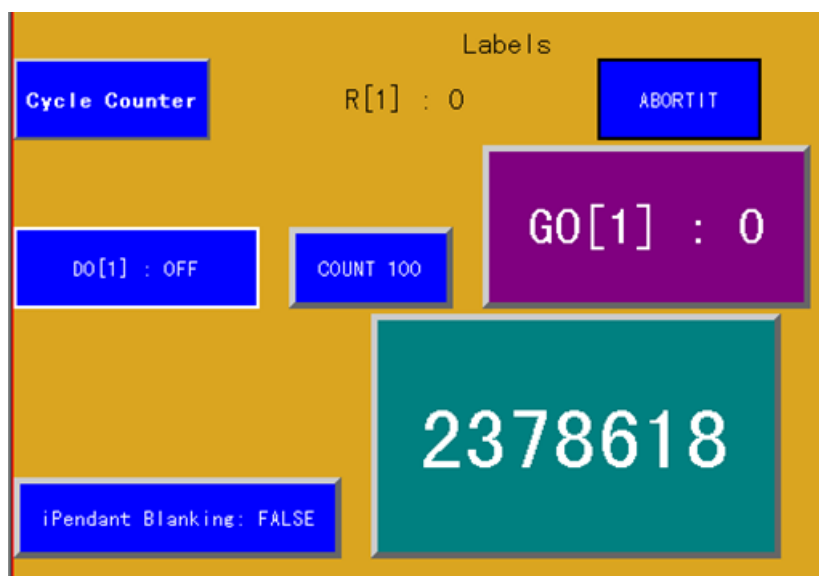
5.3.9 Help Control Limitation

When the Help Control is used remotely as in D or without login as in E, it is ignored as if it did not exist.

5.4 CONTROL DESCRIPTION

This section describes the controls that can be used on iPendant, in order.

5.4.1 Label Control



Explanation

The value of a Register, String Register, System or KAREL Variable or I/O is displayed as shown in the figure above. Fixed strings can also be displayed.

NOTE

If the read value is Boolean I/O type, ON/OFF string is displayed.

If the read value is Boolean var type, TRUE/FALSE string is displayed.

If the DataType value is 100 – Static, the string in Caption is displayed as a fixed string.

If the DataType value is other than 100 – Static, the string in caption is managed as a format string.

Property

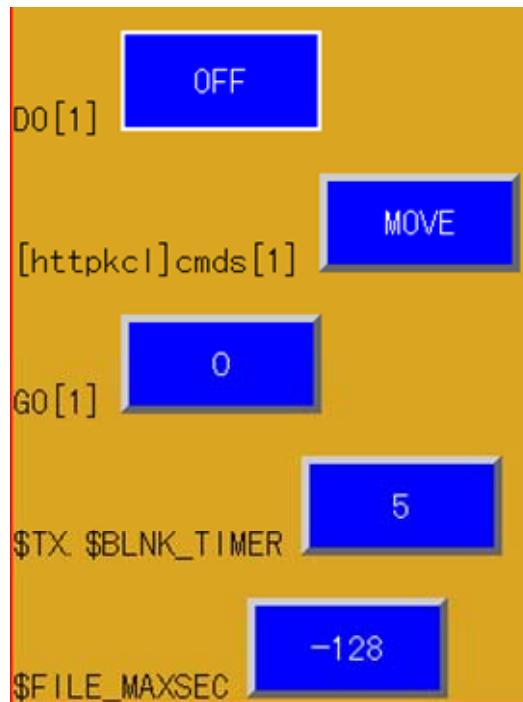
Property can be divided into the groups shown below.

Related data for read : DataType, DataIndex

Related display : ForeColor, BackColor, Caption, Font, Border.

Property	Description
Caption	Specify the fixed String or format string. (Refer to Note)(The maximum is 128 characters.)
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
DataType	Specify the type of the data for display.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
Border	Select the border design of the control.
PosAlign	Used only for position types. Selects the position alignment from the types shown below. 0 - Default Align the position similar to Position Menu. 1 - Subwindow Align the position similar to TPP and Position Register subwindows. 2 - Vertical Align the position vertically.
GroupNum	Used only for position types. Specifies the group number associated with the DataType/DataIndex if more than one group is available on the robot controller. Not used for KAREL Positions which are inherently defined for a single group.

5.4.2 EditText Control



Explanation

This is used to change the value of a Register, String Register, System or KAREL Variable (except XYZWPR type) or I/O.

The specified data value can also be monitored and displayed.

When you select this control on the page, the virtual keyboard is displayed and it accepts input.

Two kinds of virtual keyboards are supported, NumericKey and FullKey.

NOTE

If the written variable is Boolean type, you can use TRUE/FALSE string, ON/OFF string, or 0/1 value.

Property

Properties can be divided into the groups shown below:

Related data for read : DataType, DataIndex

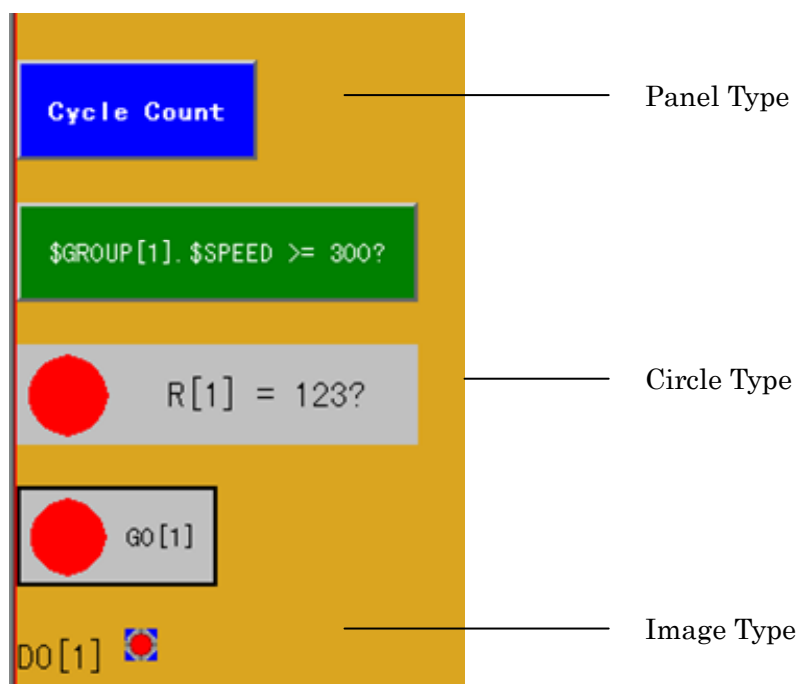
Related display : ForeColor, BackColor, Caption, Font, Border

Related virtual keyboard : Type

Property	Description
Caption	Specify the fixed String. (The maximum is 128 characters.)
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
DataType	Specify the type of the data for display and change.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed and changed instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
Border	Select the border design of the control.

Property	Description
Type	Select the type of virtual keyboard. 0 - NumericKey Display the virtual keyboard for numeric input. 1 - FullKey Display the virtual keyboard for alpha and numeric input. 2 - NumericInc Display the virtual keyboard for numeric input. Numeric key and several other keys are displayed on the key board.
ValueMin	Specify the minimum value. Available only when NumericKey is selected.
ValueMax	Specify the maximum value. Available only when NumericKey is selected.
IncrValue	This is used only when NumericInc type is selected. This decides incremental/decremental value when increment/decrement key is pressed.

5.4.3 ToggleLamp Control



Explanation

This is used to change the color of the control if the value of a Register, String Register, System or KAREL Variable (except XYZWPR type) or I/O fulfills the specified condition with the specified value or not.

The six kinds of condition operators, EQ, NE, LT, LE, GT and GE are supported. Three kinds of lamps are available. They are the panel, the circle, and the image.

In case of the image type, by exchanging the two kinds of images, a toggle lamp can be created. It is also possible to display a fixed image.

NOTE

CmpOperator:

[read value] [condition expression] [standard value for compare](e.g. X LT Y means $X < Y$)

The expression is valued like this.(read value : DataType, DataIndex)

(condition expression : CmpOperator)

(standard value for compare : CmpValue or ValueStr)

Property

Property can be divided into the groups shown below:

Related data for read : DataType, DataIndex

Related display : ForeColor, BackColor, Caption, Font, Border, ViewType

Related specifying display color : TrueColor, FalseColor

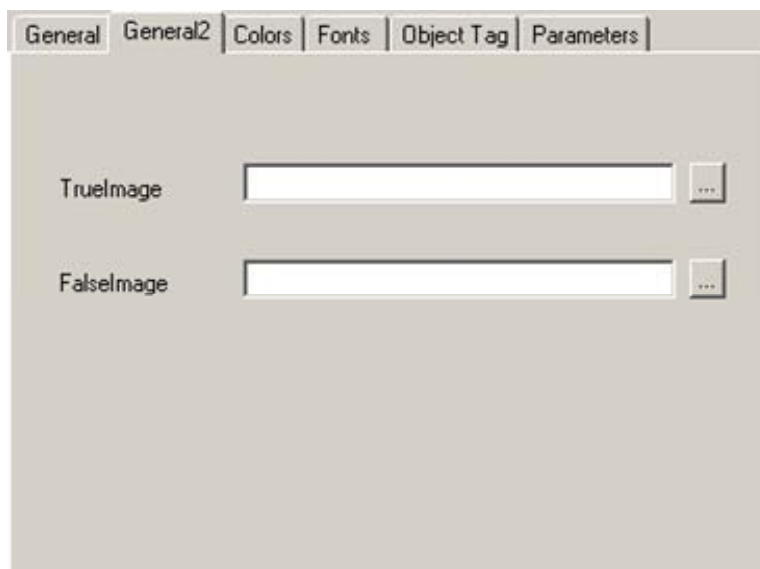
Related displayed image : TrueImage, FalseImage

Related specifying operation : CmpOperator, CmpValue or ValueStr

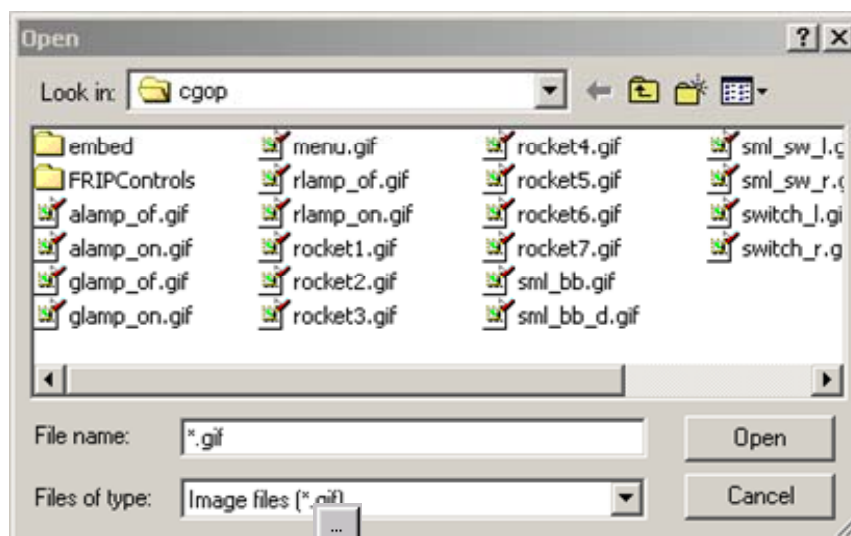
Property	Description
Caption	Specify the fixed String. (The maximum is 128 characters.)
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
DataType	Specify the type of the data to display. The following data types are supported 100 - Static Not used usually. If ViewType is image type, the image specified in FalseImage is displayed as fixed. 101 - Numeric Register Compare the value of register specified in DataIndex with CompareValue. 102 - System Variable Compare the value of the System Variable specified in DataIndex with CompareValue. 103 - KAREL Variable Compare the value of the KAREL Variable specified in DataIndex with CompareValue. 112 - String Register Compare the value of the string register specified in DataIndex with ValueStr. I/O Compare the value of I/O specified in DataType and DataIndex with CmpValue.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 - Panel Specify the panel type. 1 - Circle Specify the circle type. 3 - Image Specify the image type.
Type	Select the data type to transact. 0 - Logical Data is transacted as Boolean type. The setting of CmpOperator and the value of CmpValue are ignored. 1 - Numerical Data is transacted as numerical type. The setting of CmpOperator and the value of CmpValue is applied. 2 - String Data is transacted as string type. The setting of CmpOperator and the value of ValueStr is applied.
CmpOperator	Select the condition expression evaluated as TRUE (Refer to Note 1). This value is effective when Type is numerical. 0 - EQ Specify the equal case (=). 1 - NE Specify the not equal case (<>). 2 - LT Specify the less than case (<). 3 - LE Specify the less than or equal case (<=). 4 - GT Specify the greater than case (>). 5 - GE Specify the greater than or equal case (>=).

Property	Description
CmpValue	Specify the standard value for comparison. If the result of comparison between read value and this value is TRUE, TrueColor or TrueImage is displayed. Otherwise FalseColor or FalseImage is displayed. This value is effective when Type is numerical.
ValueStr	Specify the string value for comparison. If the result of comparison between read value and this value is TRUE, TrueColor or TrueImage is displayed, otherwise FalseColor or FalseImage is displayed. This value is effective when Type is string.
TrueColor	Specify the color to be displayed when the read value fulfill the condition expression.
FalseColor	Specify the color to be displayed when the read value does not fulfill the condition expression.
TrueImage	Specify the image to be displayed when the read value fulfill the condition expression. Used only in case that ViewType is Image type.
FalseImage	Specify the image to be displayed when the read value does not fulfill the condition expression. Used only in case that ViewType is Image type.

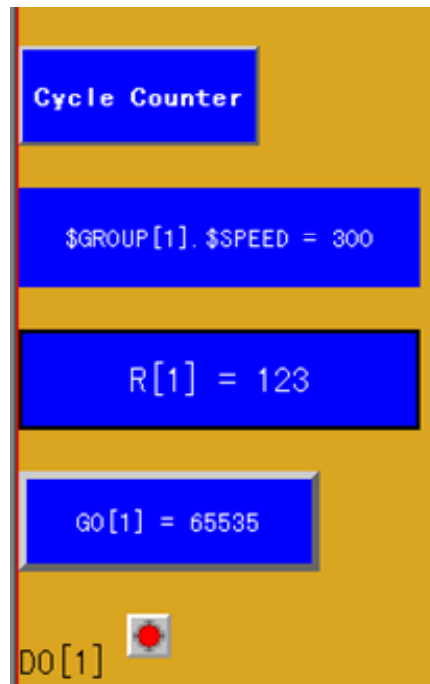
When ViewType is image type, General2 property page is available.



When ... is clicked, it is possible to specify the property using the file dialog. See the following screen for an example.



5.4.4 CommandButton Control



Explanation

Used to write the specified value to a Register, String Register, System, or KAREL Variable (except XYZWPR type) or I/O whenever the button is pushed. Can also be used to record Position Register or KAREL Position whenever the button is pushed. This can be used with Check Condition so SHIFT or TP Enable can be required. See Check Condition section.

The image button is also available.

The monitor function for the written data is not supported.

Property

Property can be divided into the below groups:

Related data for read : DataType, DataIndex

Related display : ForeColor, BackColor, Caption, Font, Border, ViewType

Related specifying image : TrueImage, FalseImage

Related specifying written value : SetValue or ValueStr

Property	Description								
Caption	Specify the fixed String.								
ForeColor	Specify the background color.								
BackColor	Specify the font name, font style and size.								
Font	Specify the font name, font style and size.								
HAlign	Specify the horizontal alignment of characters.								
VAlign	Specify the vertical alignment of characters.								
DataType	Specify the type of the data to change. The following data types are supported: <table border="0"> <tr> <td>100 - Static</td><td>Not used usually.</td></tr> <tr> <td>101 - Numeric Register</td><td>Change the value of register specified in DataIndex.</td></tr> <tr> <td>102 - System Variable</td><td>Change the value of the System Variable specified in DataIndex.</td></tr> <tr> <td>103 - KAREL Variable</td><td>Change the value of the KAREL Variable specified in DataIndex.</td></tr> </table>	100 - Static	Not used usually.	101 - Numeric Register	Change the value of register specified in DataIndex.	102 - System Variable	Change the value of the System Variable specified in DataIndex.	103 - KAREL Variable	Change the value of the KAREL Variable specified in DataIndex.
100 - Static	Not used usually.								
101 - Numeric Register	Change the value of register specified in DataIndex.								
102 - System Variable	Change the value of the System Variable specified in DataIndex.								
103 - KAREL Variable	Change the value of the KAREL Variable specified in DataIndex.								

Property	Description
DataType	105 – Position Register Record the position register specified in DataIndex. 106 - KAREL Position Record the position of the KAREL Variable specified in DataIndex. 112 – String Register Change the value of string register specified in DataIndex. I/O Change the value of I/O specified in DataType and DataIndex.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed and changed instead of the actual point.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 - Normal Specify the normal button. 1 - Image Specify the image button.
Type	Select the data type to transact. 0 - Logical Data is transacted as Boolean type. (SetValue is TRUE/FALSE) 1 - Numerical Data is transacted as numerical type. The value of SetValue is applied. 2 - String Data is transacted as String type. The value of ValueStr is applied.
SetValue	Specify the value written when the button is pushed for Logical or Numerical types.
ValueStr	Specify the string value written when the button is pushed for String type.
TrueImage	Specify the image to be displayed when the button is pushed. Used only in case that ViewType is Image type.
FalseImage	Specify the image to be displayed when the button is not pushed. Used only in case that ViewType is Image type.
TrueColor	Specify the background color to be used when a transparent image is pressed. Used only in case that ViewType is Image type.
FalseColor	Specify the background color to be used when the transparent image is not pressed. Used only in case that ViewType is Image type.
GroupNum	Specify the group number associated with the Position Register if more than one group is available on the robot controller. The default is group 1.

5.4.5 ToggleButton Control



Explanation

Used to change the value of a Register, String Register, System or KAREL Variable (except XYZWPR type) or I/O to the specified value following the ON(Pushed)/OFF(Popped) status of the button.

The monitor function for the written data is also supported.

The image button is also available.

NOTE

- The specified variable is rewritten with the specified value just after this button is operated. However the function to hold the specified value is not supported.
- In case of numeric type, the value set by On/Off of button is TrueValue/FalseValue.
- In case of string type, the value set by On/Off of button is TrueValueStr/ValueStr.
- In case of logical type, the value set by On/OFF of button is fixed value (TRUE/FALSE) and it is not possible to change this value except to Invert it.
- In case of numeric type, if the value of the specified Register, System/KAREL Variable and I/O is changed neither TrueValue nor FalseValue, the status of the button will follow the setting of the OtherPhase.
- In case of string type, if the value of the specified String Register is changed neither TrueValueStr nor ValueStr, the status of the button will follow the setting of the OtherPhase.

Property

Property can be divided into the below groups:

Related data for read: DataType, DataIndex

Related display: ForeColor, BackColor, Caption, Font, Border, ViewType

Related specifying display color : TrueColor, FalseColor, TrueStrColor, FalseStrColor

Related specifying image: TrueImage, FalseImage

Related specifying standard value: Type, TrueValue, FalseValue, TrueValueStr, ValueStr

Property	Description
Caption	Specify the fixed String.
TrueCaption	Specify the fixed String when the value is TRUE. If not specified, Caption is used.
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
DataType	Specify the type of the data to change. The following data types are supported: 100 - Static Not used usually. 101 - Numeric Register Change the value of register specified in DataIndex. 102 - System Variable Change the value of the System Variable specified in DataIndex. 103 - KAREL Variable Change the value of the KAREL Variable specified in DataIndex. 112 - String Register Change the value of string register specified in DataIndex. I/O Change the value of I/O specified in DataType and DataIndex.
DataIndex	Specify the number or the variable name.

Property	Description
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed and changed instead of the actual point.
Momentary	When TRUE, the button status is changed to ON when pressed and OFF when released. This only works when the button is pressed with the Touch Panel or if the mouse is pressed in Internet Explorer.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. Specify the image button. 0 - Normal Specify the normal button. 1 - CheckBox Specify the checkbox button. 2 - Image Specify the image button.
Type	Select the data type to transact. 0 - Logical Data is transacted as Boolean type. The value of TrueValue and FalseValue is ignored. 1 - Numerical Data is transacted as numerical type. The value of TrueValue and FalseValue is applied.
Invert	Specifies the value should be inverted. This value is effective when Type is Logical. When button status is ON, value written is OFF (FALSE). When button status is OFF, value written is ON (TRUE).
TrueValue	Specify the value written when the button status is changed to ON (TRUE). This value is effective when Type is numerical.
FalseValue	Specify the value written when the button status is changed to OFF (FALSE). This value is effective when Type is numerical.
TrueValueStr	Specify the string value written when the button status is changed to ON (TRUE). This value is effective when Type is string.
ValueStr	Specify the string value written when the button status is changed to OFF (FALSE). This value is effective when Type is string.
OtherPhase	Specify the status (TRUE/FALSE) in case that the value is equal to neither TrueValue nor FalseValue. This value is effective when Type is numerical.
TrueColor	Specify the color displayed when the read value is equal to TrueValue or not equal to FalseValue.
FalseColor	Specify the color displayed when the read value is equal to FalseValue or not equal to TrueValue.
TrueStrColor	Specify the color of characters displayed when the read value is equal to TrueValue or not equal to FalseValue. Used when the ViewType is not the image type.
FalseStrColor	Specify the color of characters displayed when the read value is equal to FalseValue or not equal to TrueValue. Used when the ViewType is not the image type.
TrueImage	Specify the image to be displayed when the button status is ON. Used only in case that ViewType is Image type.
FalseImage	Specify the image to be displayed when the button status is OFF. Used only in case that ViewType is Image type.
Confirm	Specify a confirmation popup should be displayed before setting the button status to ON. This parameter must be typed into the HTML. <param name="Confirm" value="-1">

5.4.6 Multi Control



Explanation

Used to change a maximum of 10 kinds of images or strings if the value of Register, System or KAREL Variable (except XYZWPR type) and I/O is within specified range or not. It is the multi version of ToggleLamp and can be used to create animations, such as progress bars.

For example:

If the read value is within ValueMin01 through ValueMax01, therefore $\text{ValueMin01} \leq (\text{read value}) \leq \text{ValueMax01}$ is fulfilled, the strings or image specified in Data01 is displayed.

NOTE

- If the read value is within the multi specified range, the smallest number condition is applied.
- If the read value is out of all specified ranges, the default image or string (specified in DataDefault) is displayed.

Property

Property can be divided into the groups below:

Related data for read: DataType, DataIndex

Related display: ForeColor, BackColor, Font, Border, Type

Related specifying standard value: DataDefault

Data01 - Data10

ValueMin01 - ValueMin10

ValueMax01 - ValueMax10

Property	Description
ForeColor	Specify the color of characters. This is effective when ViewType is label.
BackColor	Specify the background color.
Font	Specify the font name, font style and size. This is effective when ViewType is label.
HAlign	Specify the horizontal alignment of characters. This is effective when ViewType is label.
VAlign	Specify the vertical alignment of characters. This is effective when ViewType is label.

Property	Description
DataType	Specify the type of the data to display. The following data types are supported: 100 - Static Not used usually. If ViewType is label, the strings specified in DataDefault are displayed. If ViewType is image, the image specified in DataDefault is displayed. 101 - Numeric Register Compare the value of register specified in DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order. 102 - System Variable Compare the value of the System Variable specified in DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order. 103 - KAREL Variable Compare the value of the KAREL Variable specified in DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order. I/O Compare the value of I/O specified in DataType and DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 - Label Display strings. DataDefault, Data01 - Data10 are treated as the strings for display. 1 - Image Display image. DataDefault, Data01 - Data10 are treated as the image file name.
DataDefault	Specify the strings or the image file name if the read value does not fulfill any condition.
Data01 - Data10	Specify the strings or the image file name if the read value fulfills the condition. These data correspond to the same ordinal condition.
ValueMin01 - ValueMin10	Specify the start value of the condition range. It is possible to set 10 conditions from 01 to 10.
ValueMax01 - ValueMax10	Specify the end value of the condition range. It is possible to set 10 conditions from 01 to 10.

5.4.7 AutoChange Control



Explanation

An invisible control that is used to change the web page automatically if the value of Register, System or KAREL Variable (except XYZWPR type) or I/O is within specified range or not. Used to change the web page from the teach pendant program.

For example:

If the read value is within ValueMin01 through ValueMax01, therefore ValueMin01 <= (read value) <= ValueMax01 is fulfilled, the web page specified in PageName01 is displayed.

NOTE

- If the read value is within the multi specified range, the smallest number condition is applied.
- If the read value is out of all specified ranges, the change of web page is not done.
- The condition is ignored during screen initialization.
- This control is invisible at run-time.
- For logical variable set 0 for FALSE, and 1 for TRUE

Property

Property can be divided into the below groups.

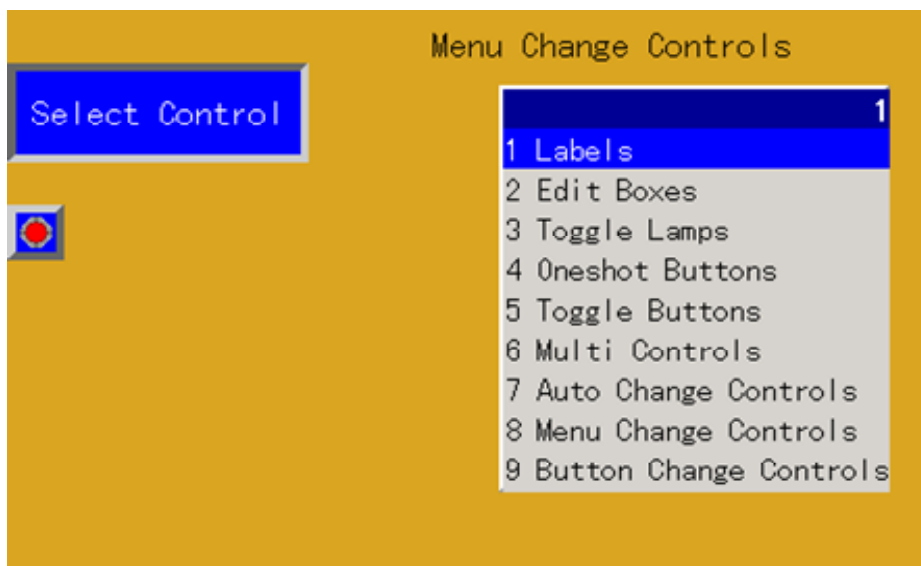
Related data for read: DataType, DataIndex

Related specifying standard value:

PageName01 - PageName10, ValueMin01 - ValueMin10, ValueMax01 - ValueMax10

Property	Description
DataType	Specify the type of the data to monitor. The following data types are supported: 100 - Static Not used usually. 101 - Numeric Register Compare the value of register specified in DataIndex is within ValueMinXX through 1 ValueMaxXX and XX is 01 through 10 in order. 102 - System Variable Compare the value of the System Variable specified in DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order. 103 - KAREL Variable Compare the value of the KAREL Variable specified in DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order. I/O Compare the value of I/O specified in DataType and DataIndex is within ValueMinXX through ValueMaxXX, and XX is 01 through 10 in order.
DataIndex	Specify the number or the variable name.
IOSim	Used only for I/O types. When TRUE, the port simulation status is monitored instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
PageName01 - PageName10	Specify the web page to display when the read value fulfills the condition. These data correspond to the same ordinal condition.
ValueMin01 - ValueMin10	Specify the start value of the condition range. It is possible to set 10 conditions from 01 to 10.
ValueMax01 - ValueMax10	Specify the end value of the condition range. It is possible to set 10 conditions from 01 to 10.

5.4.8 MenuChange Control



Explanation

Used to select the web page from the popup menu. The menu is displayed when the MenuChange button is clicked and can have a maximum of 10 items. The image button is also available.

NOTE

It is necessary to set menu item without a break. When there is a null data in PageCaptionXX, the rest of data after the null data is not displayed, even if there is effective data after the null data.

Property

Property can be divided into the following groups.

Related display: Caption, ForeColor, BackColor, Font, Border

Related changing content: PageCaption01 - PageCaption10,
PageName01 - PageName10

Related specifying image: TrueImage, FalseImage

Property	Description
Caption	Specify the fixed String.
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 – Normal Specify the normal button. 1 – Image Specify the image button.
PageCaption01 - PageCaption10	Specify the strings displayed on popup menu for selecting menus. These data correspond to the same ordinal PageNameXX.
PageName01 - PageName10	Specify the page name displayed after popup menu is selected. These data correspond to the same ordinal PageCaptionXX.
TrueImage	Specify the image to be displayed when the button is pushed. Used only in case that ViewType is Image type.

Property	Description
FalselImage	Specify the image to be displayed when the button is not pushed. Used only in case that ViewType is Image type.
TrueColor	Specify the background color to be used when a transparent image is pressed. Used only in case that ViewType is Image type.
FalseColor	Specify the background color to be used when the transparent image is not pressed. Used only in case that ViewType is Image type.

5.4.9 ButtonChange Control



Explanation

Used to display the specified web page. Can also be used to perform a command using a URL. (KCL and KCLDO Commands)

The image button is also available.

Property

Property can be divided into the below groups.

Related display: Caption, ForeColor, BackColor, Font, Border

Related specifying image: TruelImage, FalselImage

Related changing content: PageName

Property	Description
Caption	Specify the fixed String.
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 - Normal Specify the normal button. 1 - Image Specify the image button.
PageName	Specify the page name displayed after button is pushed or specify a command to perform using a URL.
TruelImage	Specify the image to be displayed when the button is pushed. Used only in case that ViewType is Image type.
FalselImage	Specify the image to be displayed when the button is not pushed. Used only in case that ViewType is Image type.

Property	Description
TrueColor	Specify the background color to be used when a transparent image is pressed. Used only in case that ViewType is Image type.
FalseColor	Specify the background color to be used when the transparent image is not pressed. Used only in case that ViewType is Image type.

5.4.10 Help Control



Explanation

The Help Control will be used to specify a single help page. The help page is displayed when the HELP key is pressed. The Help Control will be invisible on the page. It can reside anywhere on the visible portion of the web page. If it is not on the visible portion then it may not be created. It is only operational if it has been created. Only one Help Control should be used per web page. It is undefined which help page will be used if multiple Help controls exist.

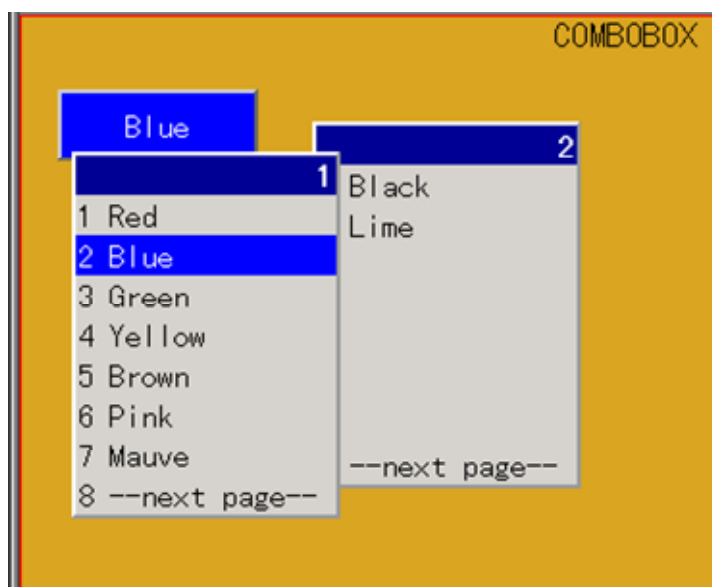
The rules for displaying the help page are exactly the same as the current iPendant help system. You cannot specify how it operates. You can only specify the web page to display when help is activated

Property

PageName

Specify the page name displayed when HELP is selected. This can be any valid URL.

5.4.11 ComboBox Control



Explanation

The ComboBox Control allows the selection of an item from a popup menu. The control displays the currently selected item. When the button is clicked (enter key), a popup list is displayed. When an item is selected, the selected value and selected text is written to the specified data on the controller. The selected value is monitored so can be dynamically changed. In the case of string registers, the string register value is also monitored so can be dynamically changed.

The selection list can be populated by various data fields:

- List of programs with program type filter.
- Contents of KAREL string array.
- Dictionary elements.
- List of file names.
- List of string registers.

Property

Related display of button:

Property	Description
Caption	Specify the format string.
ForeColor	Specify the color of characters.
BackColor	Specify the background color.
Font	Specify the font name, font style and size.
HAlign	Specify the horizontal alignment of characters.
VAlign	Specify the vertical alignment of characters.
Border	Select the border design of the control.
ViewType	Select the type of the button to display. 0 Normal Specify the normal button. 1-8 Specify the function key.

Related selection of list:

Property	Description
DataType	Specify the type of the data for the selected value of the list.
DataIndex	Specify the number or the variable name of the data for the selected value of the list.
IOSim	Used only for I/O types. When TRUE, the port simulation status is displayed and changed instead of the actual point.
Interval	Specify the interval time in ms.
Periodic	Specify whether to monitor the data or send periodically.
TextType	Specify the type of the data to set for the selected text of the list. Not valid if the Type property is set to ProgramType or FileType.
TextIndex	Specify the number or the variable name to set for the selected text of the list. Not valid if the Type property is set to ProgramType or FileType.

Related list:

Property	Description
Type	Specify the type of the ComboBox. 0 - ListType Not available in this release. 1 - ProgramType Similar to %pk in the Form Manger. The list is populated by the loaded programs on the robot. The ProgType property is used to further classify which types of programs are included in the list. The list is updated every time the list is displayed. Upon selection of the list, the program name is copied to the data specified by DataType/DataIndex. 2- VariableType Similar to %v in the Form Manger. The list is populated by a KAREL string array. The DataSource property is used to specify the KAREL string array. The list is updated every time the list is displayed. Upon selection of the list, the value is copied to the data specified by DataType/DataIndex. The string is copied to the data specified by TextType/TextIndex. 3 - DictionaryType Similar to %w in the Form Manager. The list is populated by a dictionary element. The DataSource property is used to specify the starting dictionary element. The list is updated every time the list is displayed. Upon selection of the list, the value is copied to the data specified by DataType/DataIndex. The string is copied to the data specified by TextType/TextIndex.

Property	Description
Type	<p>4 - FileType The list is populated by the file names. The DataSource property is used to specify the directory. The list is updated every time the list is displayed. Upon selection of the list, the file name is copied to the string specified by DataType/DataIndex.</p> <p>5- StringRegisterType The list is populated by the string register comments. The DataSource property is not used. The list is updated every time the list is displayed. Upon selection of the list, the string register index is copied to the data specified by DataType/DataIndex. The string register value is copied to the data specified by TextType/TextIndex. The string register value is displayed in the caption and is dynamically updated.</p>
ProgType	<p>Specify the program type if the Type property is set to ProgramType.</p> <p>1 - TP programs The list is populated by only TP programs (default).</p> <p>2 - KAREL programs The list is populated by only KAREL programs.</p> <p>6 - All The list is populated by TP, KAREL, and VR programs.</p> <p>16 - TP & KAREL The list is populated by TP and KAREL programs.</p>
DataSource	Specify the source used to populate the list when the Type property is set to VariableType, DictionaryType, or FileType.

Usage of DataSource property depends on value of Type.

Type	Description
VariableType	<p>DataSource should specify a KAREL string array. Each element of the array should define a choice in the list. The first element in the array is related to the value 0 and is never used. The second element is related to the value 1 and is the first item in the list. This is defined for compatibility with the Form Manager. The last item is either the end of the array or the first uninitialized value. For example, DataSource = "[RUNFORM]CHOICES"</p> <p>[RUNFORM] CHOICES:ARRAY[6] OF STRING[12] =</p> <p>[1] *uninit*</p> <p>[2] 'Red' <= value 1</p> <p>[3] 'Blue' <= value 2</p> <p>[4] 'Green' <= value 3</p> <p>[5] 'Yellow' <= value 4</p> <p>[6] *uninit*</p> <p>Example for Boolean type:</p> <p>[RUNFORM] IOSIM:ARRAY[3] OF STRING[12] =</p> <p>[1] *uninit*</p> <p>[2] 'UNSIM' <= value 0 for Boolean types</p> <p>[3] 'SIM' <= value 1 for Boolean types</p>

Type	Description
DictionaryType	<p>DataSource should specify the starting dictionary element. This first element is related to the value 1. One dictionary element is needed to define each entry in the list. The last entry should be followed by a dictionary element that contains "¥a". For example, DataSource = "FORM[9]"</p> <p>\$9</p> <p>"Red" * value will be set to 1</p> <p>\$-</p> <p>"Blue" * value will be set to 2</p> <p>\$-</p> <p>"Green" * value will be set to 3</p> <p>\$-</p> <p>"Yellow" * value will be set to 4</p> <p>\$-</p> <p>"¥a" * specifies end of list</p> <p>Example for Boolean type, DataSource = "FORM[9]"</p> <p>\$9</p> <p>"UNSIM" * value 0 for Boolean types</p> <p>\$-</p> <p>"SIM" * value 1 for Boolean types</p> <p>\$-</p> <p>"¥a" * specifies end of list</p>
FileType	<p>DataSource should specify a File Specification. If no DataSource is specified, all files on the default device of the robot are shown.</p> <p>Some examples of DataSource:</p> <p>MF: ¥*.DT</p> <p>*.DT (uses default device)</p> <p>FR: ¥*.DT</p>

5.4.12 Execution Control



Explanation

The Execution Control can be used to specify a KAREL program to run to call. The Execution Control will be invisible on the page. It can reside anywhere on the visible portion of the web page. If it is not on the visible portion then it may not be created. It is only operational if it has been created. Only one Execution Control should be used per web page.

After the page is completely loaded, the KAREL program is run. If the KAREL program is not loaded, it will not be run. The KAREL program will already be running if another pane has the same web page loaded. A static variable, ref_cnt, is created and maintained. The KAREL program is aborted when no panes are left to process. The KAREL program can increment the reference count if you wish it to remain running: ref_cnt = ref_cnt + 1

Property

Property	Description
Name	Specify the name of the KAREL program.
Type	Select the type of execution. 0 - KAREL Run a KAREL program.
Parameter1 – Parameter8	Specify the string parameter to set in \$UI_PANEDATA[PANEID].\$PARAMETER1 through \$UI_PANEDATA[PANEID].\$PARAMETER8.
ClickEvents	Send button click events to the KAREL program.

Property	Description
ChangeEvents	Send value change events to the KAREL program.
FocusEvents	Send focus events to the KAREL program.

NOTE

Currently, ClickEvents, ChangeEvents, and FocusEvents are only available while using DISCTRL_FORM; Otherwise, they should remain unchecked for efficiency reasons. See Appendix C for details on DISCTRL_FORM.

5.5 CONTROL DESIGN ADVICE

The process speed shows a tendency to be slow as the number of controls on the page increases. Using Image files, the process speed shows a tendency to be slow as the number of colors increase.

5.5.1 Error Code Dialog

When an error occurs, a dialog appears, such as



The error title contains the error. The object name and error content is displayed in the dialog box. See “Section 5.2.1 Object Tag” for details on changing the object name.

5.5.2 Error Code Messages

Error content

Error content	Content, Cause, Remedy
Access initialize error.	<p>Content: Error occurred trying to access the specified data type.</p> <p>Cause: Specified DataType and/or DataIndex were not valid.</p> <p>Remedy: Specify a valid type in DataType and valid index in DataIndex. To access a dictionary element use dict_name[element_number], such as TPAR[5]</p>
Register access initialize error.	<p>Content: Error occurred trying to access the specified Register.</p> <p>Cause: Specified register number doesn't exist on the robot.</p> <p>Remedy: Specify the register number in DataIndex which exists on the robot.</p>

Error content	Content, Cause, Remedy
System Variable access initialize error.	<p>Content : Error occurred trying to access the specified System Variable.</p> <p>Cause : Specified System Variable doesn't exist on the robot.</p> <p>Remedy : Specify the System Variable name in DataIndex which exists on the robot, such as \$MNUTOLNUM[1]. The System Variable type must be Integer, Real, Boolean, Short, Byte, or String.</p>
KAREL Variable access initialize error.	<p>Content : Error occurred trying to access the specified KAREL Variable.</p> <p>Cause : Specified KAREL Variable doesn't exist on the robot.</p> <p>Remedy : Specify the KAREL Variable name in DataIndex which exists on the robot, such as [KRLPRG]KRLARY[1]. The KAREL variable type must be Integer, Real, Boolean, Short, Byte, or String.</p>
I/O access initialize error.	<p>Content : Error occurred trying to access the specified I/O port.</p> <p>Cause : Specified I/O type or port number doesn't exist on the robot.</p> <p>Remedy : Specify the I/O type in DataType and the port number in DataIndex which exists on the robot.</p>
Write error.	<p>Content : Error occurred trying to write the data.</p> <p>Cause : Data may be out of valid range.</p> <p>Remedy : Check the new value is within a valid range. Check the DataType and DataIndex are specified correctly.</p>
No write access to data.	<p>Content : Error occurred trying to write the data.</p> <p>Cause : Tried to write data which is write protected.</p> <p>Remedy : Try to set manually at control start.</p>
Value out of range.	<p>Content : Error occurred trying to write the data. The valid ranges should be shown in the dialog box.</p> <p>Cause : Value is out of range.</p> <p>Remedy : Enter new value within specified ranges.</p>
Unable to connect to controller.	<p>Content : Error occurred trying to connect to the robot controller to access the data. The RPC_MAIN port number that the iPendant is trying to connect to should be shown in the dialog box.</p> <p>Cause : Controller may not be loaded correctly. The RPC server on the controller must be loaded and running.</p> <p>Remedy : Verify controller is loaded properly. Version V6.21 or later is required. Verify RPC_MAIN port number in \$SERVENT system variable is valid on the controller.</p>

Error content	Content, Cause, Remedy
No Image File.	<p>Content : No Image File is shown instead of the specified image.</p> <p>Cause : Specified image file could not be loaded from the robot.</p> <p>Remedy : Verify the image file is specified correctly as a control parameter. Verify the image file is copied to the correct device and directory on the robot.</p>
HTTP/1.0 404 File Not Found	<p>Content : "HTTP/1.0 404 File Not Found" is shown instead of the specified web page.</p> <p>Cause : Specified web page could not be found on the robot.</p> <p>Remedy: Verify the web page is specified correctly as a control parameter. Verify the web page is copied to the correct device and directory on the robot.</p>

6 USING THE CHART CONTROL

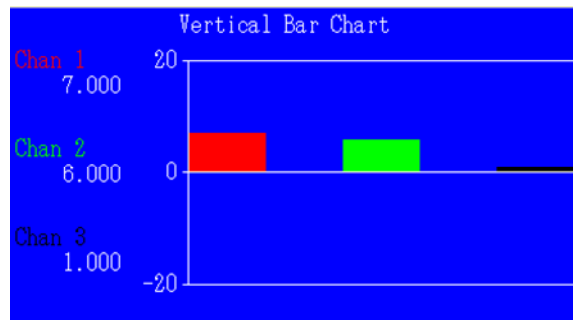
The Chart control is a means to graphically render data on the *iPendant* or an externally connected browser such as Internet Explorer.

The Chart control is configured through properties within the web page to control the display and configure the data sources. A Chart has one or more channels or data sources. These sources are supplied from the robot controller, and specified by name and monitor rate.

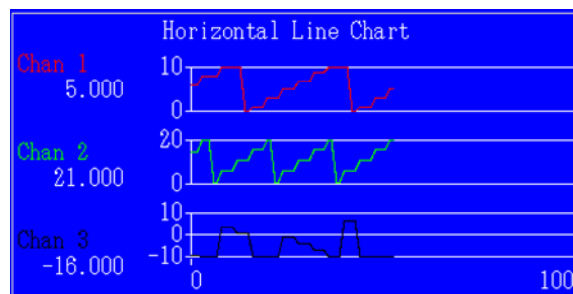
Properties of the control also determine the scaling, format, colorization and layout of many of the Chart's characteristics.

A Chart may be specified as either a Bar or Line chart.

1) Bar chart:



2) Line chart:



These charts can be oriented horizontally or vertically.

A maximum of 8 channels per Chart can be configured and active at one time.

Conventions:

Usually axes are referred to as X-axis and Y-axis for 2-dimensional charting. This gets confusing when horizontal or vertical orientation requires translation to the browser or plug-in native coordinate system. Instead we will use the following nomenclature regardless of the orientation:

SampleAxis or SampleScale

SampleAxis or SampleScale refers to the sample index.

For a Bar Chart this is the base axis of the bars.

For a Line Chart this is the common axis among multiple channels. It may be time, or just a running sample increment but it is common and relates the individual channels.

DataAxis or DataScale

DataAxis or DataScale refers to the data value of the channel.

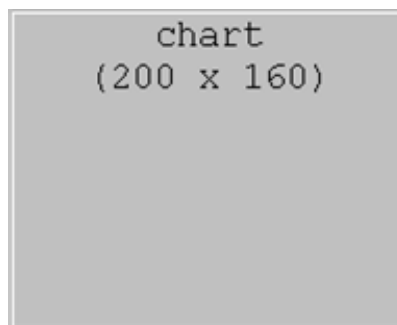
For a Bar Chart this is common to all channels.

For a Line Chart this is independent among all channels since its scaling will dictate the visibility of small or large changes in the channels data value.

The easiest way to create chart on the screen for the *i*Pendant is by putting the Chart controls on the web page and by setting its properties.

The characteristics controlling items such as chart layout, orientation, data source, labels, colors, etc. can be controlled by the user setting the appropriate properties of the Chart control.

Initially when a drawing control is first inserted on the web page, by Microsoft Office SharePoint Designer 2007, it is displayed with the default light gray background in 200 by 160 pixel size. The two text items in the control reflect the Caption and size (width and height). These are dynamically updated when the size or Caption is changed.



As the chart is resized and channels are enabled, the display changes dynamically to give a general feel for the overall layout.

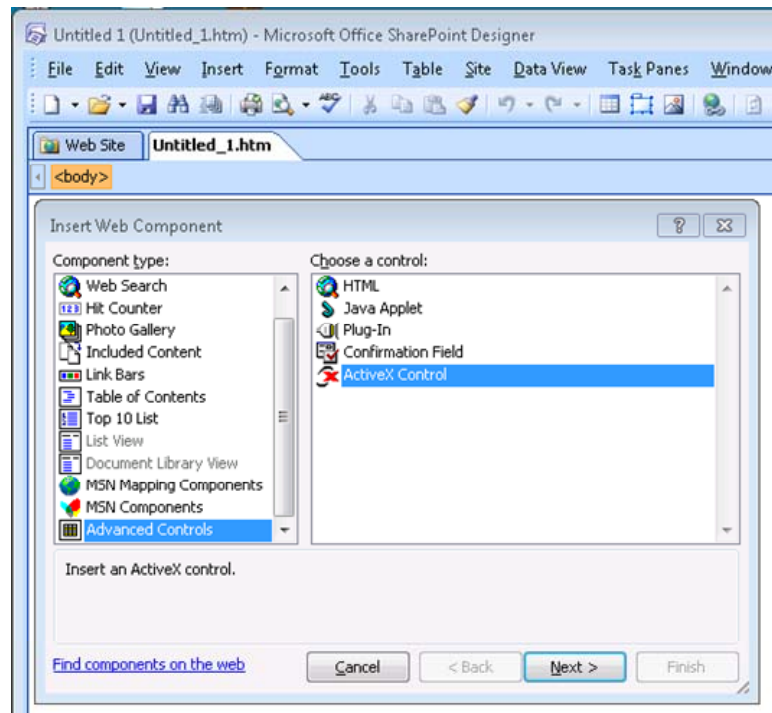
```
'My example chart'
Channel #1 (440 x 220)
  1.000
Channel #2 <Line,H>
  1.000
Channel #3
  1.000
Channel #4
  1.000
'some note'
```

6.1 CONTROL ARRANGEMENT

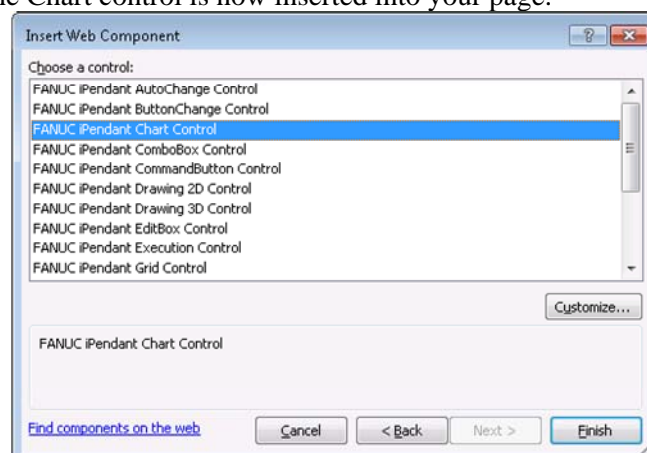
To add a Chart Control to your web in SharePoint Designer 2007, follow these steps:

STEP

- 1) Position your cursor where you want the control to appear.
- 2) Select **Insert | Web Component | Advanced Controls| ActiveX Control** from the menu bar.



- 3) Press **NEXT >**
- 4) Choose the **FANUC iPendant Chart Control** from the list of available FANUC iPendant controls, and click Finish. The Chart control is now inserted into your page.



NOTE

If the **FANUC iPendant Chart Control** is not an option in the list, use the **Customize...** button to enable it. If the option is not available under the **Customize...** setup, then the proper software for the FANUC controls has not been installed.

- 5) The particular properties can be configured from the ActiveX property pages. Either double click on the control or right click and select **ActiveX Control Properties**.

6.2 COMMON CHART CONTROL PROPERTIES

A Charting control has the following properties.

6.2.1 Object Tag

The Object Tag dialog allows you to specify some standard attributes associated with your control. The Name is used when an error occurs. The **width** and **height** can be specified in pixels or percentage (%). Of course, you can resize the control in SharePoint Designer 2007 by dragging the control's handles with the mouse.

The remaining properties are grouped by their property page tab.

6.2.2 Fonts

The font properties associated with the chart as the “defaults” are modified from the **Fonts** property page. Text strings use these “default” settings when no other font attributes are specified.

Enhanced font control has been introduced with the R-30iB controller. This includes being able to specify:

- Fonts by name
- Strikeout
- Underline

New pages created will automatically take advantage of this enhanced control. To edit or create web pages that exhibit font characteristics previous to these R-30iB enhancements you must edit the HTML source. Replace the line:

```
<param name="TrueFont" value="1">
```

with

```
<param name="TrueFont" value="0">
```

Font control beyond that configured by the ActiveX Control Properties pages described below can be made by directly modifying the HTML source or through XML attributes sent via the pipe using the following properties names:

CaptionFont
AnnotateFont
LabelFont
DataFont

The value for these properties is a comma separated list decoded as follows:

The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used.

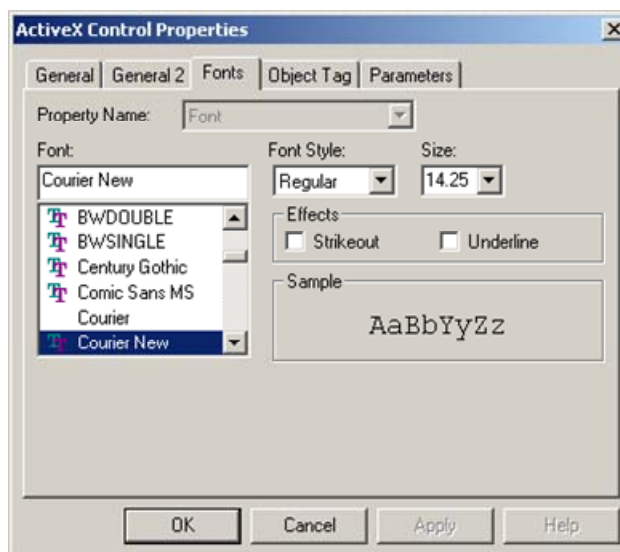
The remaining items are optional

- b or -b to turn bold on/off
- i or -i to turn italic on/off
- s or -s to turn strikethrough on/off
- u or -u to turn underline on/off

- wxx to set the weight to xx
- r to set bold, italic, strikethrough and underline off and weight to 0.

If you wish to override the default font for specific items on the control, set one or more of the following parameters:

CaptionFontName
AnnotateFontName
LabelFontName
DataFontName

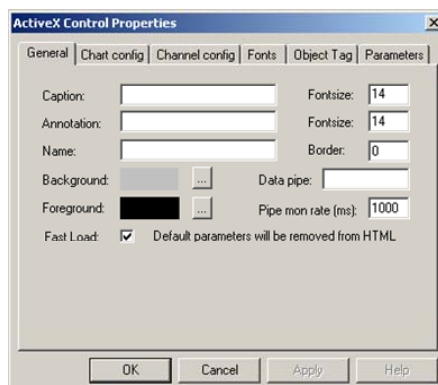


Item	Description
Font	Font specifies the name of the font passed in the Font property.
Font Style	Font Style can specify regular, bold, italic and italic bold. This affects whether the properties FontBold and FontItalic are optionally passed.
Size	FontSize specifies the size of the font. It is passed in the FontSize property.
Strikeout	Strikeout specifies whether the FontStrikethrough property is passed.
Underline	Underline specifies whether the FontUnderline property is passed.

NOTE

On the *iPendant*, 8-point font is about the smallest that is readable.

6.2.3 General Chart Properties



6.2.3.1 Caption & CaptionFontSize

Caption Specifies a text string to be displayed as a general label for the chart.

This string is centered at the top of the Chart control, in the control's foreground color and font properties.

CaptionFontSize is used to override the "default" font size specified from the Fonts property tab.

6.2.3.2 Annotation & AnnotateFontSize

Annotation Specifies a text string to be displayed as a note for the chart.

This string is centered at the bottom of the Chart control, in the control's foreground color and font properties.

AnnotateFontSize is used to override the "default" font size specified from the Fonts property tab.

6.2.3.3 Name

Name Is a text string that specifies an association to a KAREL program on the controller.

Currently, when the Drawing control is instantiated, the control checks for the existence of two program variables on the controller. This variable can be used to pass text commands in the format and with the content of other properties to dynamically affect the Drawing control's operation.

[name] cmdstr If the variable exists and is of type STRING the control creates a monitor for this variable. Strings of commands can be written (set) to this variable and are delivered to the Drawing control to affect 'dynamic' actions such as data updates or changes in some chart properties.

[name]cmdack If the variable exists and is of type BOOLEAN or INTEGER, the Drawing control uses this variable to acknowledge that commands have been accepted via the **command** variable and acted on. The **cmdack** variable is set TRUE or 1 when that command is accepted by the plug-in.

Note: The **Name** parameter can utilize **!PanelID**.

6.2.3.4 Border

Border Select the border surrounding the Chart control.

Allowable values are:

- >0 A 3 dimensional raised border, that number of pixels thick.
- =0 No border line.
- <0 A 3 dimensional depressed border, that number of pixels thick.

6.2.3.5 Foreground & Background Colors

The Colors dialog allows you to specify the color of certain elements. The *iPendant* supports 256 colors. Colors associated with the Chart control are:

- ForeColor** Specify the "default" color of any entity that does not have an explicit color defined.
- BackColor** Specify the "default" background color.

Note: Colors are specified as a decimal value represent bbgrr (blue green red) format, where as hexadecimal value preceded by the # sign are in rrggbb (red green blue) format.

For example; ForeColor = 10531008 (decimal) is the same as ForeColor = #C0B0A0 (hex).

6.2.3.6 Pipe

Pipe Is a text string that specifies a named data file created on the controller and only associated with the PIP: device. The name can be any 8.3-formatted name, for example: drawing1.dat.

This file is used to dynamically deliver data to the drawing control on the *iPendant*. Data that you write to the pipe file is read and used by the control.

The name of this file is sent to the controller when the chart control is instantiated on the *iPendant*. The file name is concatenated to PIP: and the file opened. If the file does not exist then the file is created and opened for read. If the file does exist prior to this, the file is opened for read access and a seek to the end of file done to eliminate the possibility of stale data.

When the last chart control using this file is gone, the file is deleted.

NOTE

The **Pipe** parameter can utilize **!PanelID**.

6.2.3.7 PipeMonRate

PipeMonRate Specifies the time in milliseconds at which the Pipe file data is sampled.

NOTE

This value should be specified at a rate that give reasonable performance and display update. The controller will enforce a minimum of 100ms.

6.2.3.8 FastLoad

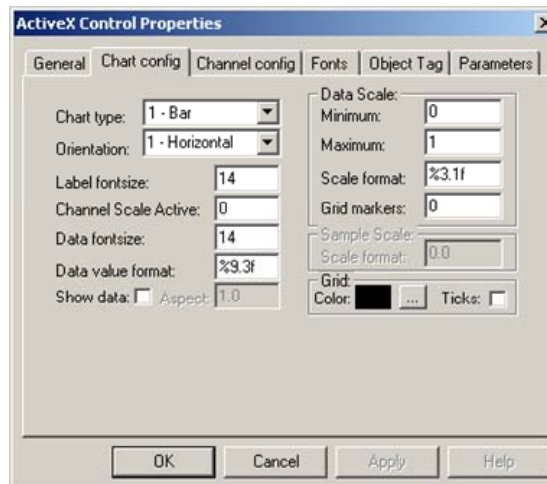
FastLoad is selected by the checkbox.

Checked Specifies that properties that are set to their corresponding “default” values are not included in the web page. This effectively saves load time.

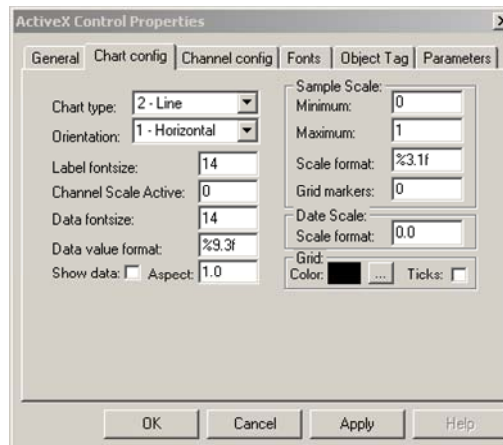
Unchecked Specifies that all parameters are included in the web page, whether they have “default” values or not.

6.2.4 Chart config tab

Chart config properties differ slightly between a Bar chart and Line chart. For the Bar chart the **Chart config** tab is as follows:



And for a Line chart the Chart config tab is as follows:



NOTE

The Data Scale and Sample Scale properties have different meanings between the two types of charts (Bar or Line).

6.2.4.1 ChartType

ChartType Specifies the type of chart as a numeric value.

Allowable values are:

- 1 Displays a bar chart
- 2 Displays a line chart

6.2.4.2 Orientation

Orientation Specifies the chart's orientation on the *iPendant* as a numeric value.

Allowable values are:

- 1 Displays a horizontal chart
- 2 Displays a vertical chart

6.2.4.3 LabelFontSize

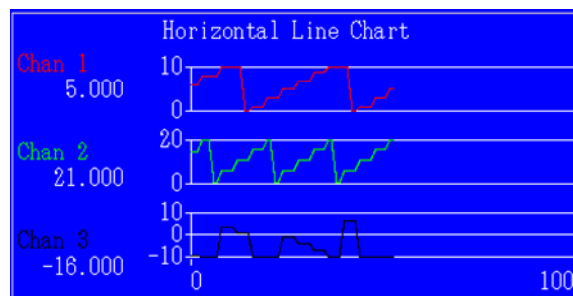
LabelFontSize is used to override the “default” font size specified from the Fonts property tab.

6.2.4.4 LineScaleActive

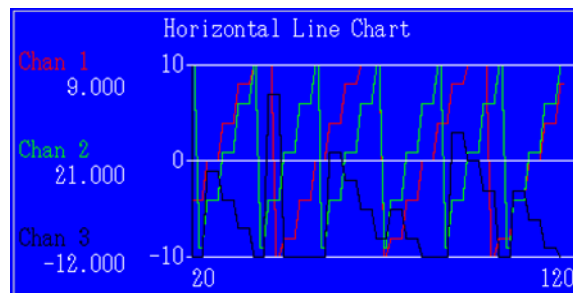
LineScaleActive Is set from the “Channel Scale Active: box. It controls which data scale(s) are selected for display on a line chart.

When multiple channels are configured and active a chart may display in two ways.

1) A multiple ‘stacked’ charts each with their individual data scales.



2) A single chart, with one channel selected for the data scale. In this case the data scale edge will be in the color of the corresponding channel (in the following example it is channel 3, or black). When channels are overlaid in this manner the non-selected channels are still scaled to their own data scale.



Allowable values are:

- 0 Channels are independently displayed in a ‘stacked’ fashion
- 1 to 8 Channels are overlaid. The value selects the channel to display the data scale for. All other channels are still called along the data scale by their own data scale.

6.2.4.5 DataFontSize

DataFontSize is used to override the “default” font size specified from the Fonts property tab.

6.2.4.6 DataFormat

DataFormat Specifies the format in which the channel’s data value will be displayed. The display appears in the legend, near the channel’s label, if DataShowValues = 1.

The format is in ‘C’ printf format for a single precision floating point number. For example; %9.3f
Default if not specified is %9.3f.

NOTE

Currently %f is the only allowed format.

6.2.4.7 DataShowValues

DataShowValues Specifies whether or not the values of each channel is displayed in a table near the channels label and in the DataFormat.

Allowable values are:

- 0** Do not display values
- 1** Display the values.

Default if not specified is 0.

6.2.4.8 SampleScaleAspect

The Aspect box sets SampleScaleAspect. It specifies a SAMPLE SCALE multiplier value.

This is a simple multiplier value applied to the sample scale after the SampleScale Minimum and Maximum values.

A Typical use might be to equate samples to time.

For example;

- 1 sample every 32ms.
- Sample scale range is initially set to 0 to 200 samples.
- SampleScaleAspect=0.010
- The result is that the sample scale minimum and maximums are multiplied by this factor, resulting in a displayed sample scale of 0 to 2 (seconds).

NOTE

There are no explicit checks on the bounds of this value. It is simply used as a multiplier, so it could also be a value > 1.

6.2.4.9 SampleScale

SampleScale Specifies the sample scale's minimum and maximum extent, since all LINE CHART channels have one common sample scale.

This property only pertains to a Line chart.

The extents can be specified as a pair of comma separated numbers (min,max), or a single value.

Allowable values are:

- min** If value < 0, max is assumed 0.
- min,max** Min and max as specified.
- max** If value > 0, min is assumed 0.

For a horizontal Line chart; **min** is the left value and **max** is the right value.

For a vertical line chart; **min** is the bottom value and **max** is the top value.

NOTE

For a Bar Chart this property is ignored.

6.2.4.10 SampleScaleFormat

SampleScaleFormat Specifies the format of the sample scale labels.

This property only pertains to a Line chart.

The format is in 'C' printf format for a single precision floating point number. For example; %3.1f

Default if not specified is %3.1f

NOTE

Currently %f is the only allowed format.

6.2.4.11 SampleGrid

SampleGrid Is controlled by the **Grid Markers** box. It specifies the interval of a grid to be displayed along the **SampleScale**.

This property only pertains to a Line chart.

Allowable values are:

- 0** No grid is displayed.
- >0** Display grid in increment of this along the axis.

6.2.4.12 DataScaleFormat

DataScaleFormat Specifies the format of the DataScale labels.

This property only pertains to a Line chart.

The format is in 'C' printf format for a single precision floating point number. For example; %3.1f

Default if not specified is %3.1f.

NOTE

Currently %f is the only allowed format.

6.2.4.13 DataScale

DataScale Specifies the data scale's minimum and maximum extent, since all BAR CHART channels have one common data scale.

This property only pertains to a Bar chart.

The extents can be specified as a pair of comma separated numbers (min,max), or a single value.

Allowable values are:

- min** If value < 0, max is assumed 0.
- min,max** Min and max as specified.
- max** If value > 0, min is assumed 0.

For a horizontal Bar chart; **min** is the left value and **max** is the right value.

For a vertical Bar chart; **min** is the bottom value and **max** is the top value.

NOTE

For a Line Chart this property is ignored.

6.2.4.14 DataScaleFormat

DataScaleFormat Specifies the format of the data scale labels.

This property only pertains to a Bar chart.

The format is in 'C' printf format for a single precision floating point number. For example; %3.1f

Default if not specified is %3.1f

NOTE

Currently %f is the only allowed format.

6.2.4.15 DataGrid

DataGrid Is controlled by the **Grid Markers** box. It specifies the interval of a grid to be displayed along the **DataScale**.

This property only pertains to a Bar chart.

Allowable values are:

- 0 No grid is displayed.
- >0 Display grid in increment of this along the axis.

6.2.4.16 GridColor

GridColor Specifies the color for the Grid lines.

Note: There is only one color for both Sample and Data scale grids.

Default if not specified is the same color as the data and sample scales.

6.2.4.17 GridType

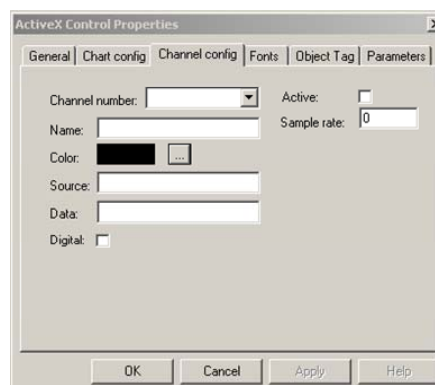
GridType Is controlled by the Ticks checkbox. It specifies how the Grid lines are displayed.

Allowable values are:

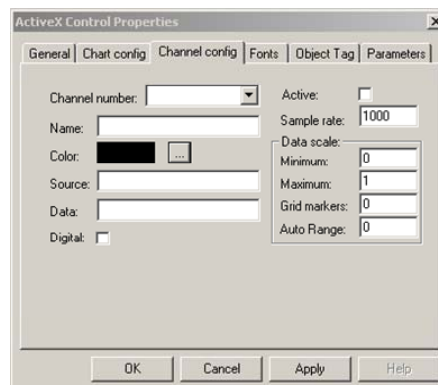
- 0 Grid lines are displayed all the way across the chart.
- 1 Grid lines are displayed as "tick" marks along the axis scale(s).

6.2.5 Channel config tab

Channel config properties of a Bar Chart control are:



Or for a Line Chart control are:



The only significant difference is that for a Line Chart the Data Scale properties can be set on a per channel basis.

The Channel number selects the channel number (N) that the remainder of the properties on this tab affects.

Allowable numbers are 1 through 8.

6.2.5.1 ChN_Name

ChN_Name Specifies a text string to be used as name for the channel.

This is displayed on the chart as the Channel label in the channel's color (specified by **chN_Color**).

Default if not specified is "chN_Name", where the numeric channel number replaces N.

NOTE

If the name is specified with a leading \$, it is interpreted as a system variable which should be of type KSTRING. In this case, the contents of the string variable will be used as the name.

6.2.5.2 ChN_Color

ChN_Color Specifies the color for the channel.

The color value can be specified in the hexadecimal or decimal format that the ForeColor and BackColor use.

Default if not specified is the chart's foreground color (specified by **ForeColor**).

6.2.5.3 ChN_Source

ChN_Source Specifies the source of data monitored from the controller.

The source can specify any of the following. Note that where indexes are appropriate, they are specified enclosed by [].

Allowable values are:

DataScale	Description
numreg[n]	Numeric register n.
\$sysvar	System variable.
[prog]var	KAREL variable.
DIN[n]	Digital in n.

DataScale	Description
DOUT[n]	Digital out n.
ANIN[n]	Analog in n.
ANOUT[n]	Analog out n.
PLCIN[n]	PLC in n.
PLCOUT[n]	PLC out n.
RDI[n]	Robot digital in n.
RDO[n]	Robot digital out n.
SOPIN[n]	Standard Operator Panel in n.
SOPOUT[n]	Standard Operator Panel out n.
TPIN[n]	Teach Pendant in n.
TPOUT[n]	Teach Pendant out n.
WELDIN[n]	Weld in n.
WELDOUT[n]	Weld out n.
GPIN[n]	Group in n.
GPOUT[n]	Group out n.
LDIN[n]	Laser Digital in n.
LDOUT[n]	Laser Digital out n.
LAIN[n]	Laser Analog in n.
LAOUT[n]	Laser Analog out n.
WSIN[n]	Weld stick in n.
WSOUT[n]	Weld stick out n.

6.2.5.4 ChN_Data

ChN_Data Specifies data for a single channel as a single or sequence of data points.

Values are specified as a comma separated series of data values:

data Data is one channel data point.

data, data,... Data is a sequence channel data points.

This can be used to create a 'static' chart where all data is supplied as web page content.

6.2.5.5 ChN_Digital

ChN_Digital Specifies whether this channel should be represented as an analog or digital signal with square transitions.

This property only pertains to a Line chart.

Allowable values are:

0 Channel is analog representation.

1 Channel is digital representation.

As a digital representation, the transition is made at the next sample that changes.

6.2.5.6 ChN_State

ChN_State Specifies whether the specified channel is on or off from the Active: checkbox.

Allowable values are:

- 0** Channel is off (inactive).
- 1** Channel is on (active).

6.2.5.7 ChN_Rate

ChN_Rate Specifies the rate in milliseconds that the channel source is monitored at from the Sample rate: input.

NOTE

When multiple channels are active the fastest rate among all channels is used as the monitor rate for all channels.

6.2.5.8 ChN_DataScale

ChN_DataScale Specifies the data scale minimum and maximum extents for a Line Chart.

This property only pertains to a Line chart.

The extents can be specified as a pair of comma separated numbers (min,max), or a single value.

Allowable values are:

- min** Assumed if value < 0, max is then 0.
- min,max** Min and max specified.
- max** Assumed if value > 0, min is then 0.

6.2.5.9 ChN_DataGrid

ChNDataGrid Specifies the interval of a grid to be displayed along the DataScale.

This property only pertains to a Line chart.

Allowable values are:

- 0** No grid is displayed.
- >0** Display grid in increment of this along the axis.

6.2.5.10 ChN_AutoRange

ChN_AutoRange Species that a channel can auto range or peak detect.

To avoid a false peaks, the number of points that must exceed the min or max data scale can be specified.

The value is a comma-separated pair of numbers representing state and number of consecutive points (**state,npoints**).

state is a numeric flag that turns auto range on or off. Allowable values are:

- 0** Do not autorange
- !=0** Auto range after n consecutive points fall outside the data scale.

npoints specifies the number of consecutive points that must be outside the data scale, on either extreme, before autoscale takes effect.

6.2.6 Miscellaneous

6.2.6.1 SampleMarkerN

SampleMarkerN Specifies one of four markers that appear as lines parallel to the sample axis on a line chart in the color specified by the **SampleMarkerColor**. (N = marker #)

This property only pertains to a Line chart.
Values are specified as a comma separated pair:

0 sample marker is off
state, value state is 1 or 0 for on or off, value is along the sample axis.

6.2.6.2 SampleMarkerColor

SampleMarkerColor Specifies the color of the sample markers.
This property only pertains to a Line chart.
This is in the decimal or hexadecimal format that the **ForeColor** and **BackColor** use.

6.2.6.3 ChN_DataMarkerN

ChN_DataMarkerN Specifies one of four markers PER CHANNEL that appear as lines parallel to the data axis on the chart in the channels color. (N = marker #)

Values are specified as a comma separated pair:

0 data marker is off
state, value state is 1 or 0 for on or off, value is along the data axis.

6.2.6.4 Ch_Data_N

Ch_Data_N Specifies data for ALL channels.

NOTE

This is not to be confused with **ChN_Data**.

Values are specified as a comma separated series of data:

Data_ch1,...,data_chN Data is the actual sequence of data points.

This sequence may be a repeating sequence on data points that are distributed among N channels. For example: Ch_Data_3=1,2,3,4,5,6,7,8,9,10 causes:

1,4,7,10 Sent to channel 1.
2,5,8 Sent to channel 2.
3,6,9 Sent to channel 3.

This can be used to create a 'static' chart where all data is supplied as web page content.

NOTE

No accounting is made to make sure that the number of data points is integral to the number of channels (N) specified.

6.2.6.5 ChN_Clear

ChN_Clear Specifies that channel N is to be cleared, that is all data is deleted.

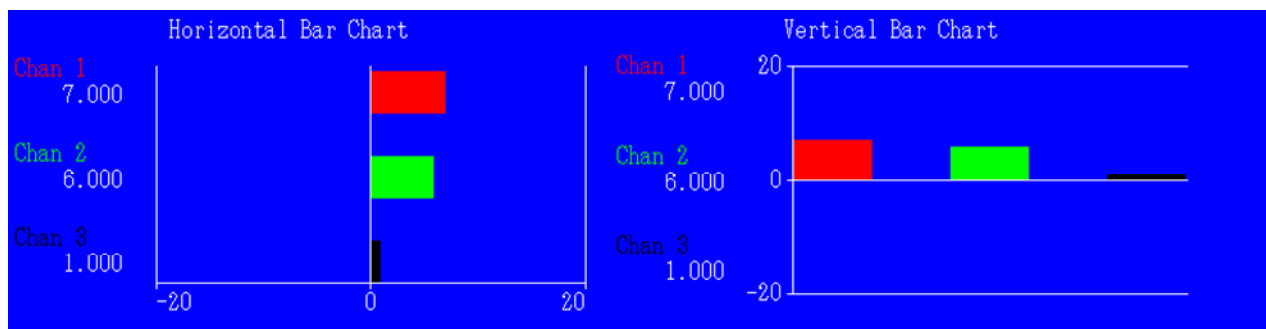
6.2.6.6 ChartClear

ChartClear Specifies that all channels on a chart are to be cleared, that is all data is deleted.

6.3 CHART CONTROL DESCRIPTION

This section describes the charting controls that can be used on *iPendant*.

6.3.1 Bar Chart Control



Explanation

A Bar Chart control presents data in the familiar format of 2-dimensional bars.

Each bar represents a separate channel with specific characteristics to uniquely identify it.

A legend is supplied to identify the channel by name and optionally show the current value.

The current limitation supports monitoring up to 6 channels simultaneously.

Each bar (channel) is scaled evenly across the sample axis.

DataScaleMarkers can be placed on the chart sample scale. These are independent, 4 for each channel.

Data types that can be monitored are specified by the channel source as native controller data type. These are monitored at set intervals and converted internally from their native data type to floating point.

Property

Property can be divided into the groups shown below.

This first group should be considered static and not changed once the control is created:

Property	Description
id	Specifies an ID string.
height	Specifies the height in pixels or %.
width	Specifies the width in pixels or %.
FastLoad	used internally by property editor.
ChartType	Specifies the chart type, Bar or Line
FontSize	Specifies the default font size.
Name	Specifies the name to relate to a KAREL program.
Pipe	Specifies the name of the pipe file.
Orientation	Specifies orientation as horizontal or vertical.
ForeColor	Specifies the default foreground color.
BackColor	Specifies the background color.
TrueFont	Indicates that enhanced font control is u

The following may be change dynamically:

Property	Description
Annotate	Specifies the annotation text string to display.
AnnotateFontSize	Specifies the font size for the annotate text.
AnnotateFont	Specifies multiple font characteristics .
Caption	Specifies the annotation text string to display.
CaptionFontSize	Specifies the font size of the caption text.
CaptionFont	Specifies multiple font characteristics for the caption
LabelFontSize	Specifies the font size of the channel label text.
LabelFont	Specifies multiple font characteristics for the label
DataFontSize	Specifies the font size of the channel data value.
DataFont	Specifies multiple font characteristics for the data
Border	Specifies the border thickness in pixels of the control.
DataScale	Specifies the data scale extents.
GridType	Specifies the grid type, whole lines or ticks.
GridColor	Specifies the default gridline color.
DataGrid	Specifies data scale grid increments.
DataShowValues	Specifies that data value be displayed in the legend.
DataFormat	Specifies the 'C' format for the display of the data values.
DataScaleFormat	Specifies the 'C' format for the DataScale extents and markers.
ChanMonRate	Specifies the default rate for channel monitors in milliseconds.
PipeMonRate	Specifies the pipe file monitor rate in milliseconds.
CmdMonRate	Specifies the CmdStr variable monitor rate in milliseconds. (Currently set the same as PipMonRate).
Verbose	Specifies some verbose text if = 1.
ch_Data_N	Specifies a comma separated data stream, N repeating channels per sample.

The following can occur 1 per channel N:

Property	Description
chN_Name	Specifies the text label for the channel legend.
chN_Color	Specifies color of the channel.
chN_Source	Specifies the source to start a monitor on the controller.
chN_Rate	Specifies the monitor rate in milliseconds.
chN_State	Specifies if the channel is active, on or not.
chN_Spread	Not implemented.
chN_Data	Specifies a comma separated data stream for the channel.
chN_AutoRange	Specifies that the channel should auto range.
chN_DataMarkerN	Specifies one of 4 markers along the data scale.

6.3.2 Line Chart Control



Explanation

A Line Chart control presents data in the familiar format that is similar to an oscilloscope trace.

A chart can have independent graphs of each channel in a 'stacked' fashion, or can have all channels overlaid on one graph. When the channels are overlaid one channel can be selected to have its DataScale displayed, in this case the data scale is displayed in the color of the active channel. Otherwise, the DataScale is displayed in the Chart control's default ForeColor.

When displayed in stacked mode, the channels are evenly sized and distributed over the display area of the control.

A legend is supplied to identify the channel by name and optionally show the current value.

The current limitation supports monitoring up to 6 channels simultaneously.

All channels share a common sample axis.

DataScaleMarkers can be placed on the chart sample scale. These are independent, 4 for each channel.

Up to 4 Sample Scale markers can be placed on the Chart. These are common to all channels.

Data types that can be monitored are specified by the channel source as native controller data type. These are monitored at set intervals and converted internally from their native data type to floating point.

Property

Property can be divided into the groups shown below.

This first group should be considered static and not changed once the control is created:

Property	Description
id	Specifies an ID string.
height	Specifies the height in pixels or %.
width	Specifies the width in pixels or %.
CtlType	Specifies chart control.
FastLoad	used internally by property editor.
ChartType	Specifies the chart type, Bar or Line
FontSize	Specifies the default font size.
Name	Specifies the name to relate to a KAREL program.
Pipe	Specifies the name of the pipe file.
Orientation	Specifies orientation as horizontal or vertical.
ForeColor	Specifies the default foreground color.
BackColor	Specifies the background color.
TrueFont	Indicates that enhanced font control is used

The following may be change dynamically:

Property	Description
Annotate	Specifies the annotation text string to display.
AnnotateFontSize	Specifies the font size for the annotate text.
AnnotateFont	Specifies multiple font characteristics .
Caption	Specifies the annotation text string to display.
CaptionFontSize	Specifies the font size of the caption text.
CaptionFont	Specifies multiple font characteristics for the caption
LabelFontSize	Specifies the font size of the channel label text.
LabelFont	Specifies multiple font characteristics for the label
DataFontSize	Specifies the font size of the channel data value.
DataFont	Specifies multiple font characteristics for the data
Border	Specifies the border thickness in pixels of the control.
LineWeight	Specifies line thickness.
GridType	Specifies the grid type, whole lines or ticks.
GridColor	Specifies the default gridline color.
SampleScale	Specifies the sample scale extents.
SampleGrid	Specifies sample scale grid line increments.
SampleScaleAspect	Specifies a multiplier for the SampleScale.
DataShowValues	Specifies that data value be displayed in the legend.
DataFormat	Specifies the 'C' format for the display of the data values.
SampleScaleFormat	Specifies the 'C' format for the SampleScale extents and markers.
DataScaleFormat	Specifies the 'C' format for the DataScale extents and markers.
LineScaleActive	Specifies if Line chart(s) are overlaid or stacked, and if overlaid which channel is selected as the data scale.
SampleMarkerN	Specifies one of 4 markers along the SampleScale.
SampleMarkerColor	Specifies the default SampleMarker colors.
ChanMonRate	Specifies the default rate for channel monitors in milliseconds.
PipeMonRate	Specifies the pipe file monitor rate in milliseconds.
CmdMonRate	Specifies the CmdStr variable monitor rate in milliseconds. (Currently set the same as PipMonRate).
Verbose	Specifies some verbose text if = 1.
ch_Data_N	Specifies a comma separated data stream, N repeating channels per sample.

The following can occur 1 per channel N:

Property	Description
chN_Name	Specifies the text label for the channel legend.
chN_Color	Specifies color of the channel.
chN_Digital	Specifies the channel is digital if 1.
chN_Source	Specifies the source to start a monitor on the controller.
chN_Rate	Specifies the monitor rate in milliseconds.
chN_DataScale	Specifies the channels data scale extents.
chN_DataGrid	Specifies data scale grid line.
chN_State	Specifies if the channel is active, on or not.
chN_Spread	not implemented.
chN_Data	Specifies a comma separated data stream for the channel.

Property	Description
chN_AutoRange	Specifies that the channel should auto range.
chN_DataMarkerN	Specifies one of 4 markers along the data scale.

6.4 PROPERTIES WITH ADDITIONAL OPTIONAL VALUES

A few of the properties were extended to add optional values that can be used. This was done for the sake of compactness and backwards portability.

chN_DataMarkerN="flag,value[,color]"
 color is an optional parameter.
 It modifies only this one single marker.

SampleMarkerN="flag,value[,color]"
 color is an optional parameter.
 It modifies only this one single marker.

ChN_AutoRange="state,npoints[,extendby]"
 extendby is an optional parameter.
 Without it, the autorange will extent the maximum or minimum to equal the new value.
 With it specified the maximum or minimum is extended by this * the current maximum or minimum.

6.5 CHARTING CONTROL DESIGN ADVICE

A completely 'static' chart can be placed on a web page and data completely supplied through it. To do this,

- define the **ChN_Name** and **ChN_Color** for the necessary channels,
- make sure they are active (**ChN_State** = 1)
- if you do not define a source, by **ChN_Source**, for a channel then dynamic updates do not automatically happen,
- use either separate **ChN_Data** properties for each channel, or **Ch_Data_N** for all channels to supply the data.

If a **Name** is designated for a chart, then KAREL variables as described for the **Name** property can be used to deliver dynamic commands to the chart control. For example, a chart with Name = chart1

```
KCL> set def chart1
KCL> create var command string[127]
KCL> create var cmdack integer
Go to the web page that creates chart1
KCL> set var command = 'ch1_data=1,3,5,7,9,...' Will deliver this data to the chart for channel
1.
```

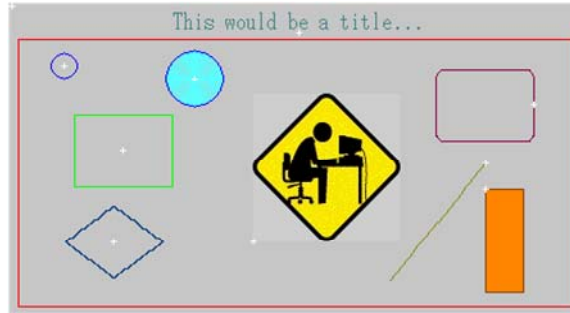
Where formats can be specified please be aware of the possible ranges of these values. For example, the **DataFormat** should account for possible spikes that may appear on a channel.

6.5.1 Error Handling

When an error occurs, either the chart will either not display or will display erroneously.

7 USING THE DRAWING CONTROL

The Drawing control is a means to render line graphics on the *iPendant* or an externally connected browser such as Internet Explorer. The following example is one with each type of drawing entity rendered.



Properties of the control also determine the scaling, format, colorization and layout of many of the Drawing's characteristics.

Conventions:

The Drawing control is a 'white board' that line art entities are placed on. These entities have characteristics (or properties that are unique to each).

Entities are:

- Text – as in an ASCII string
- Line – a single element drawn from point 1 to point 2
- Path – a sequence of lines
- Circle – a circle empty or filled
- Rectangle – a rectangle empty or filled, square cornered or rounded
- Diamond – a diamond (not filled)
- Image – a picture, source: .jpg, .gif or .png file

Entities have attributes. Some general ones are:

- Name – an ASCII name to identify the entity
- Color – a color
- Fill color – color of the fill
- Height & width, or radius – to determine its size
- Location – to locate its origin (x,y)
- Points – Begin and end locations
- Layer – drawing layer that the entity belongs to.

Each entity belongs to a layer. By default if not specified the entity belongs to layer 0.

- Layer 0 is always (ON) displayed.
- Other layers 1 through 9 can be turned on or off.

The normal coordinate system for the Drawing control on the *iPendant* (and Internet Explorer) is:

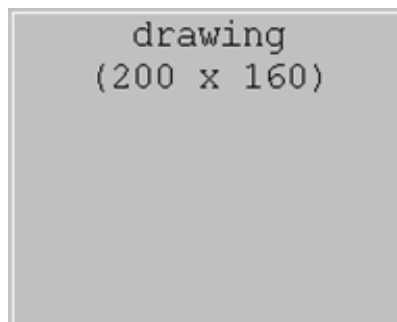
- Upper left is 0, 0
- Upper right is + max X
- Lower Left is + max Y

To simplify things an 'InvertY' flag is defined to translate the coordinate system to:

- Upper left is + max Y
- Upper right is + max X
- Lower Left is 0, 0

The easiest way to create drawing on the screen for the *iPendant* is by putting the Drawing control on the web page and by setting its properties.

Initially when Microsoft Office SharePoint Designer 2007 inserts a drawing control on the web page,, it is displayed with the default light gray background in 200 by 160 pixel size. The two text items in the control reflect the Caption and size (width and height). These are dynamically updated when the size or Caption is changed.

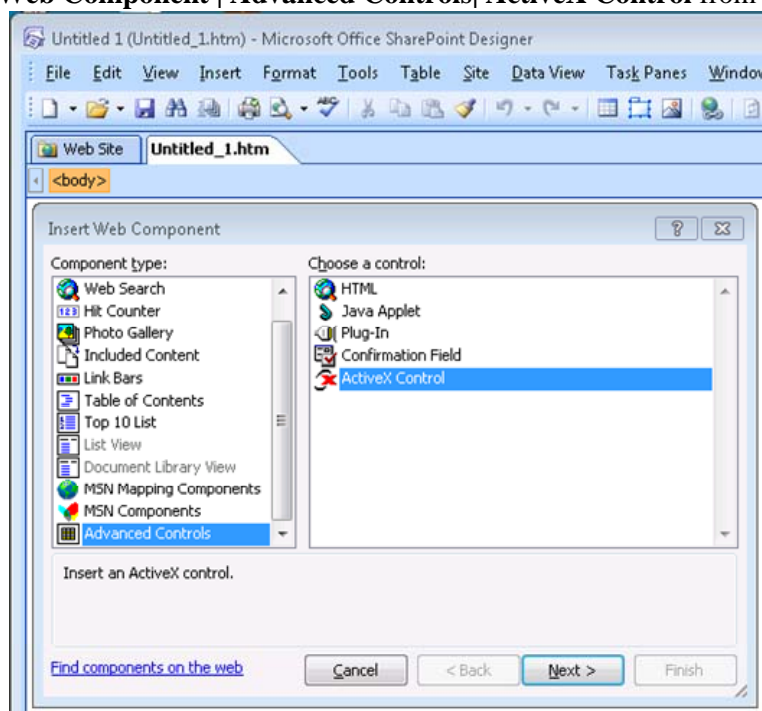


The characteristics controlling items such as orientation, fonts, colors, etc. can be controlled by the user setting the appropriate properties of the Drawing control.

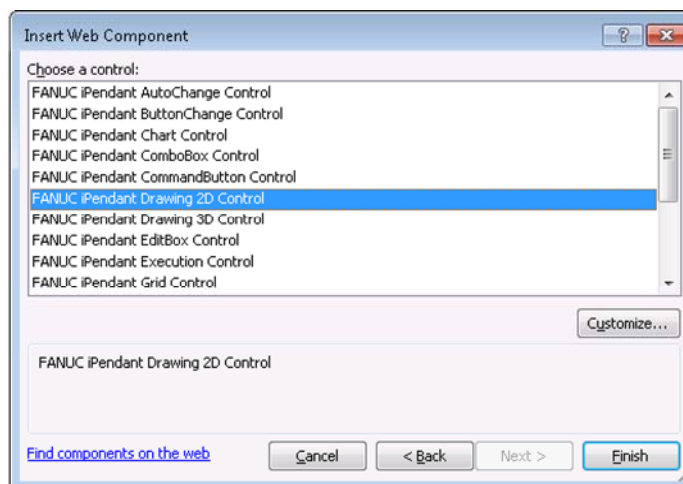
7.1 CONTROL ARRANGEMENT

To add a Drawing 2D Control to your web in SharePoint Designer 2007, follow these steps:

- 1) Position your cursor where you want the control to appear.
- 2) Select **Insert | Web Component | Advanced Controls| ActiveX Control** from the menu bar.



- 3) Press **NEXT >**
- 4) Choose the **FANUC Drawing 2D Control** from the list of available FANUC *i*Pendant controls, and click Finish. The Chart control is now inserted into your page.

**NOTE**

If the **FANUC Drawing 2D Control** is not an option in the list, use the **Customize...** button to enable it. If the option is not available under the **Customize...** setup, then the proper software for the FANUC controls has not been installed.

- 5) The particular properties can be configured from the ActiveX property pages. Either double click on the control or right click and select **ActiveX Control Properties**.

7.2 COMMON DRAWING CONTROL PROPERTIES

A Drawing control has the following properties.

7.2.1 Object Tag

The Object Tag dialog allows you to specify some standard attributes associated with your control. The Name is used when an error occurs. The **width** and **height** can be specified in pixels or percentage (%). Of course, you can resize the control in SharePoint Designer 2007 by dragging the control's handles with the mouse.

The remaining properties are grouped by their property page tabs.

7.2.2 Fonts

The font properties associated with the drawing as the “defaults” are modified from the **Fonts** property page. These “default” settings are used by the **Text** string entities when no other font attributes are specified.

Enhanced font control has been introduced with the R-30*i*B controller. This includes being able to specify:

- Fonts by name
- Strikeout
- Underline

New pages created will automatically take advantage of this enhanced control. To edit or create web pages that exhibit font characteristics previous to these R-30iB enhancements you must edit the HTML source. Replace the line:

```
<param name="TrueFont" value="1">
```

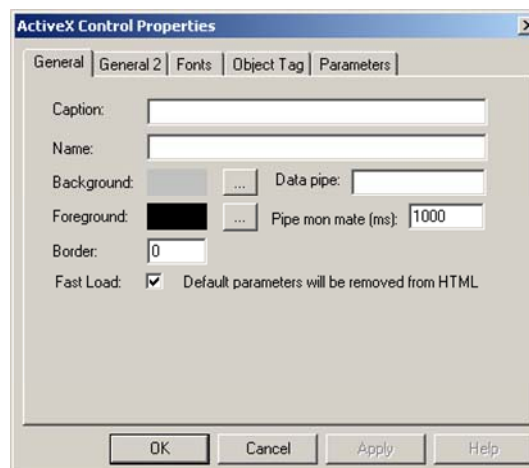
with

```
<param name="TrueFont" value="0">
```



Item	Description
Font	Font specifies the name of the font passed in the Font property.
Font Style	Font Style can specify regular, bold, italic and italic bold. This affects whether the properties FontBold and FontItalic are optionally passed.
Size	Size specifies the size is the font. It is passed in the FontSize property.
Strikeout	Strikeout specifies whether the FontStrikethrough property is passed.
Underline	Underline specifies whether the FontUnderline property is passed.

7.2.3 General tab



7.2.3.1 Caption

Caption Is a text string that is passed as a property but not used. It is there as an aid to the developer since it is displayed on the SharePoint Designer 2007 design and preview views.

7.2.3.2 Name

Name Is a text string that specifies an association to a KAREL program on the controller.

Currently, when the Drawing control is instantiated, the control checks for the existence of two program variables on the controller. These variable can be used to pass text commands in the format and with the content of other properties to dynamically affect the Drawing control's operation.

[name] cmdstr If the variable exists and is of type STRING the control creates a monitor for this variable. Strings of commands can be written (set) to this variable and are delivered to the Drawing control to affect 'dynamic' actions such as data updates or changes in some chart properties.

[name]cmdack If the variable exists and is of type BOOLEAN or INTEGER, the Drawing control uses this variable to acknowledge that commands have been accepted via the **command** variable and acted on. The **cmdack** variable is set TRUE or 1 when that command is accepted by the plug-in.

NOTE

The **Name** parameter can utilize **!PanelID**.

7.2.3.3 Foreground & Background Colors

The Colors dialog allows you to specify the color of certain elements. The *iPendant* supports 256 colors. Colors associated with the Drawing control are:

ForeColor Specify the "default" color of any entity that does not have an explicit color defined.
BackColor Specify the background color.

Note: Colors are specified as a decimal value represent bbgrr (blue green red) format, where as hexadecimal value preceded by the # sign are in rrggbb (red green blue) format.

For example; ForeColor = 10531008 (decimal) is the same as ForeColor = #C0B0A0 (hex).

7.2.3.4 Pipe

Pipe Is a text string that specifies a named data file created on the controller and only associated with the PIP: device.

This file is used to dynamically deliver data to the drawing control on the *iPendant*. The name can be any 8.3-formatted name, for example: drawing1.dat.

The name of this file is sent to the controller when the drawing control is instantiated on the *iPendant*. The file name is concatenated to PIP: and the file opened. If the file does not exist then the file is created and opened for read. If the file does exist prior to this, the file is opened for read access and a seek to the end of file done to eliminate the possibility of stale data.

When the last drawing control using this file is gone, the file is deleted.

NOTE

The **Pipe** parameter can utilize **!PanelID**.

7.2.3.5 PipeMonRate

PipeMonRate Specifies the time in milliseconds at which the Pipe file data is sampled.

Note: This value should be specified at a rate that give reasonable performance and display update. The controller will enforce a minimum of 100ms.

7.2.3.6 Border

Border Selects the size of the border surrounding the Drawing control.

Allowable values are:

- >0 A 3 dimensional raised border, that number of pixels thick.
- =0 No border line.
- <0 A 3 dimensional depressed border, that number of pixels thick.

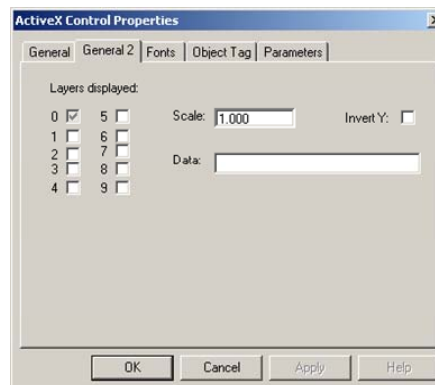
7.2.3.7 FastLoad

FastLoad is selected by the checkbox.

Checked Specifies that properties that are set to their corresponding “default” values are not included in the web page. This effectively saves load time.

Unchecked Specifies that all parameters are included in the web page, whether they have “default” values or not.

7.2.4 General 2 tab



7.2.4.1 Layer control

A Drawing has 10 layers.

Layer 0 is always displayed. By default if an entity does not have a layer attribute, it belongs to layer 0 and is always displayed.

Other layers 1 through 9 can be turned on or off by the check boxes. These check boxes control the value of two parameters **LayerOn** and **LayerOff**.

Both parameters are can be a comma separated lists of layers to turn on or off.

LayerOn Specifies a comma separated list of layers to turn on.

LayerOff Specifies a comma separated list of layers to turn off.

Examples are:

LayerOn="n" Turns on layer n

LayerOff="n,m,o" Turns off layers n,m and o.

7.2.4.2 Scale

Scale specifies a multiplier applied to both the X and Y scaling.

NOTE

This is used as a simple multiplier and can result in display peculiarities for each display entity. The default for this setting should be left at 1.0.

7.2.4.3 InvertY

InvertY Specifies than the Y scale is inverted.

Normally the Y scale begins at the top and increases + to the bottom.

When inverted the Y scale begins at the bottom and increases + to the top.

Allowable values are:

0 Y Not inverted. Y scale begins at the top and increases + to the bottom.

1 Y inverted. Y scale begins at the bottom and increases + to the top.

7.2.4.4 Data

Data Specifies a raw data string that the user can type in.

This is supplied to the Drawing control via the Data parameter.

Since the value is a string and should not contain double quotes, a single quote should be used in place of double quotes. During processing all single quotes in the data value will be converted to double quotes.

NOTE

This implies that single quotes cannot be handled or escaped. They should not be used.

7.2.5 The Entities

7.2.5.1 Text

Text Specifies a text entity on the drawing.

Attributes as follows:

Attribute	Description
text[=id]	Text entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0

Attribute	Description
font=f	Specifies font size, +/- can be used to increment or decrement from the default font size. The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used. The remaining items are optional <ul style="list-style-type: none"> • b or -b to turn bold on/off • i or -i to turn italic on/off • s or -s to turn strikethrough on/off • u or -u to turn underline on/off • wx to set the weight to xx • r to set bold, italic, strikethrough and underline off and weight to 0.
locate=x,y	Specifies x, y origin relative to the white board
color=cvalue	Specifies color as hex or decimal value
bold=b	bold (1=bold, 0=nobold)
italic=i	Specifies italics (1=italic, 0=no italic)

7.2.5.2 Line

Line Specifies a line entity on the drawing.

Attributes as follows:

Attribute	Description
line[=id]	Line entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
points=x1,y1,x2,y2	Specifies the beginning point x1,y1 and ending point x2,y2
color=cvalue	Specifies color as hex or decimal value

7.2.5.3 Path

Path Specifies a path entity on the drawing. The path entity is a sequence of contiguous lines along a sequence of points.

Attributes as follows:

Attribute	Description
path[=id]	Path entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
points=x1,y1,x2,y2,...,xn,yn	Specifies the beginning point that a sequence of lines are drawn along
color=cvalue	Specifies color as hex or decimal value

7.2.5.4 Circ

Circ Specifies a circle entity on the drawing.

Attributes as follows:

Attribute	Description
circ[=id]	Circle entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
locate=x,y	Specifies x, y origin relative to the white board
color=cvalue	Specifies color as hex or decimal value
radius=r	Specifies the circles radius r in pixels.
weight=w	Specifies a thickness w in pixels
fill=fcolor	Optional fill color in hex or decimal
halign=align	Horizontal alignment (left, center, right), Default = left
valign=align	Vertical alignment (top, middle, bottom), Default = bottom

7.2.5.5 Rect

Rect Specifies a rectangle entity on the drawing.

Attributes as follows:

Attribute	Description
rect[=id]	Rectangle entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
locate=x,y	Specifies x, y origin relative to the white board
color=cvalue	Specifies color as hex or decimal value
height=h	Specifies height as h pixels
width=w	Specifies width as w pixels
fill=color	Optional fill color in hex or decimal
halign=align	Horizontal alignment (left, center, right), Default = left
valign=align	Vertical alignment (top, middle, bottom), Default = bottom
radius=r	Specifies the radius r in pixels of the corners.

7.2.5.6 Diam

Diam Specifies a diamond entity on the drawing.

Attributes as follows:

Attribute	Description
diam[=id]	Diamond entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
locate=x,y	Specifies x, y origin relative to the white board
color=cvalue	Specifies color as hex or decimal value
height=h	Specifies height as h pixels
width=w	Specifies width as w pixels
halign=align	Horizontal alignment (left, center, right), Default = left
valign=align	Vertical alignment (top, middle, bottom), Default = bottom

7.2.5.7 Imag

Imag Specifies an image entity on the drawing.

Attributes as follows:

Attribute	Description
imag[=id]	Image entity with optional id
layer=n	Optionally specifies the layer it belongs to. Default is layer 0
locate=x,y	Specifies x, y origin relative to the white board
height=h	Specifies height as h pixels, default is file image height
width=w	Specifies width as w pixels, default is file image width
halign=align	Horizontal alignment (left, center, right), Default = left
valign=align	Vertical alignment (top, middle, bottom), Default = bottom
imagename=path	Specifies the source file name and path of the image

7.2.6 Miscellaneous Properties

The following properties are available but have limited use unless used for dynamic drawing.

If a **Name** is designated for a chart, then KAREL variables as described for the **Name** property can be used to deliver dynamic commands to the drawing control. For example, a drawing with Name = drawing1

```
KCL> set def drawing1
KCL> create var command string[127]
KCL> create var cmdack integer
Go to the web page that creates drawing1
KCL> set var command = '....' Will deliver this data to the drawing.
```

7.2.6.1 Clear

Clear Specifies that the entire drawing be cleared, leaving only the blank white board displayed.

7.2.6.2 Delete

Delete=id Specifies that an entity previously created with id be deleted. It is permanently removed from the drawing.

7.2.6.3 LayerOn

LayerOn Specifies a single layer or comma-separated list of layers that are turned ON. All entities that belong to these layers are then displayed.

7.2.6.4 LayerOff

LayerOff Specifies a single layer or comma-separated list of layers that are turned OFF. All entities that belong to these layers are not displayed. These entities are not deleted they are simply not displayed.

7.2.7 Dynamic Entity modifications

Using the ID to specify an already existing entity, you can modify any of the entities properties without deleting and redrawing it.

For example, assume the following entities are drawn:

```
text=t1 layer=1 string='This is the title' font=12 color=8421440 locate=220,20 valign=bottom
      halign=center
path=pth1 color=255 points=5,25;430,25;430,230;5,230;5,25 layer=3
circ=c1 layer=4 color=16711680 locate=40,45 radius=10
```

Later, by using the ID, properties of those entities can be modified without deleting and redrawing the whole entity:

```
text=txt1 color=#0000ff      Changes the text txt1 to red
circ=c1 locate=10,10         Makes circle c1's origin 10, 10
path=pth1 points=10,10,20,20 Adds two more points to whatever points exist for path pth1.
```

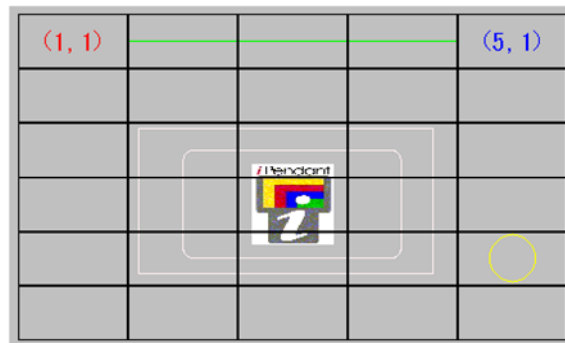
8 USING THE GRID CONTROL

The GRID control is a means to display SHAPE graphics (text, lines, rectangles, circles, images, etc) on a web page aligned in a manner controlled by XML.

The GRID control divides the display real estate into a 2 dimensional array of evenly spaced tiles. SHAPES can then be aligned to the individual tiles for display. The SHAPES are generally located and sized relative to the TILES making it easy to resize the entire display.

The GRID control also makes use of inheritance. The control itself has basic characteristics such as color and font information. The GRID is a child of the control, so characteristics not explicitly declared is inherited from the control. This filters down to the TILE belonging to the GRID and SHAPES belonging to a TILE.

A simple display using the GRID control is the following:



NOTE

Here tile outlines have been turned on to show where the tiles are located. Normally the outlines would not be turned on.

In this example a grid 5 tiles wide by 6 tiles high is displayed.

- Tile 1,1 (upper left) contains the red text “(1,1)” centered in it.
- Tile 1,1 also has a line from tile 1,1, beginning at 100% of the width and 50% of the height. The length of the line is 300% of the origin tile’s width at 1,1.
- Tile 5,1 (upper right) has the blue text “(5,1)”.
- Tile 3,3 contains a .gif image center in it, height is 150% and width is 75%.
- Tile 3,3 also contains 2 white rectangles. The inner one has rounded corners.
- Tile 5,4 contains a yellow circle. Its diameter is specified as 50% of the width of the tile.

The GRID control itself is specified by the user in html syntax within a .stm file, see Section [8.1 INSERTING A GRID CONTROL ON A WEB PAGE](#).

The content of the GRID, TILES and SHAPES are specified in XML (Extensible Markup Language) and is furnished to the Grid control in one of 3 ways:

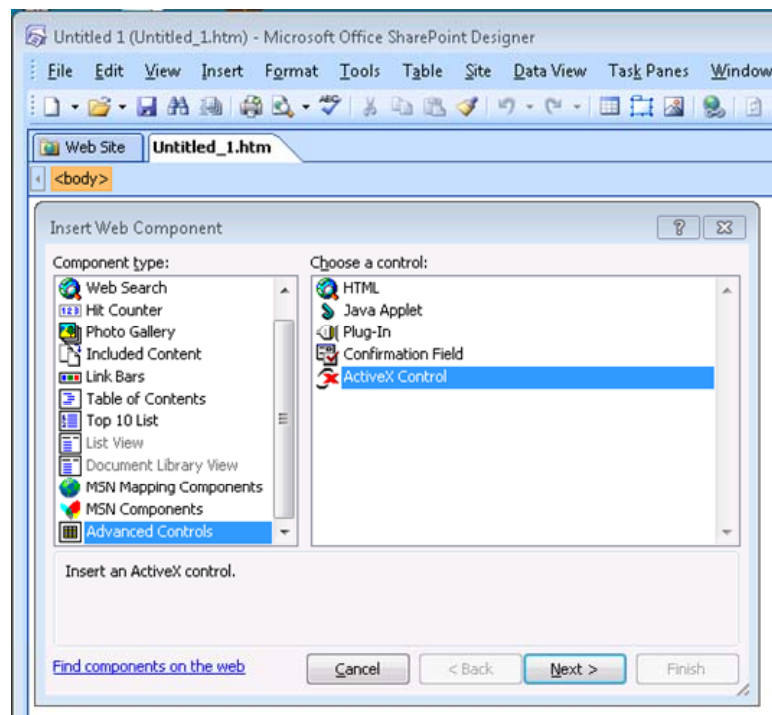
- most often through a pipe file (This is specified in the control’s “pipe” property),
- through the KAREL variables specified by the “name” property or,
- through the “data” property of the web page itself.

8.1 INSERTING A GRID CONTROL ON A WEB PAGE

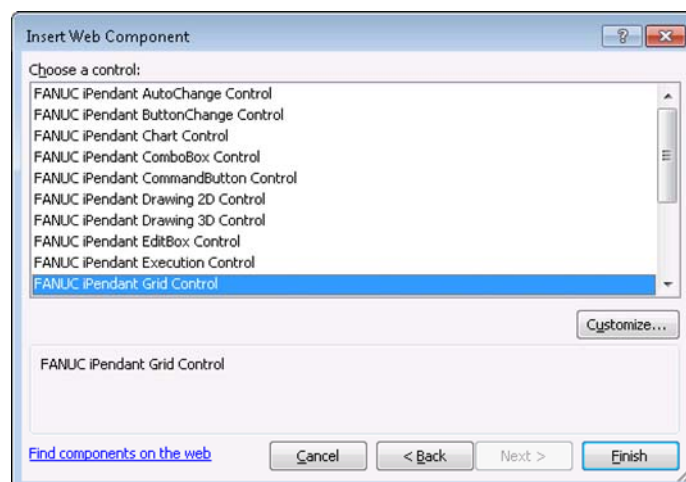
To add a Grid Control to your web in SharePoint Designer 2007, follow these steps:

STEP

- 1) Position your cursor where you want the control to appear
- 2) Select **Insert | Web Component | Advanced Controls| ActiveX Control** from the menu bar.



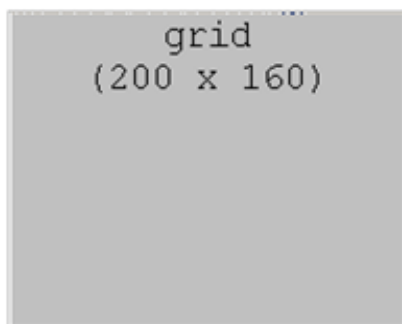
- 3) Press **NEXT >**
- 4) Choose the **FANUC Grid Control** from the list of available FANUC *i*Pendant controls, and click Finish. The Grid control is now inserted into your page.



NOTE

If the **FANUC Grid Control** is not an option in the list, use the **Customize...** button to enable it. If the option is not available under the **Customize...** setup, then the proper software for the FANUC controls has not been installed

- 5) Once the control is inserted into your web page you will see a blank control, with the size in pixels as:

**NOTE**

You can grab the edges of the Grid control and stretch it to resize.

- 6) The particular properties can be configured from the ActiveX property pages. Either double click on the control or right click and select ActiveX Control Properties.



8.2 COMMON GRID CONTROL PROPERTIES

A Grid control has the following properties.

From here on we'll use the term of attributes, since these can be more closely equated to XML content.

8.2.1 Object Tag

The Object Tag dialog allows you to specify some standard attributes associated with your control.

Name is used when an error occurs.

Width and **Height** can be specified in pixels or percentage (%). Of course, you can resize the control in SharePoint Designer 2007 by dragging the control's handles with the mouse.

The remaining properties are grouped by their property page tabs.

8.2.2 Fonts

The font properties associated with the GRID as the "defaults" are modified from the **Fonts** property page. These "default" settings are used by the **Text** string entities when no other font attributes are specified.

Enhanced font control has been introduced with the R-30iB controller. This includes being able to specify:

- Fonts by name
- Strikeout
- Underline

New pages created will automatically take advantage of this enhanced control. To edit or create web pages that exhibit font characteristics previous to these R-30iB enhancements you must edit the HTML source. Replace the line:

```

    <param name="TrueFont" value="1">
with
    <param name="TrueFont" value="0">

```

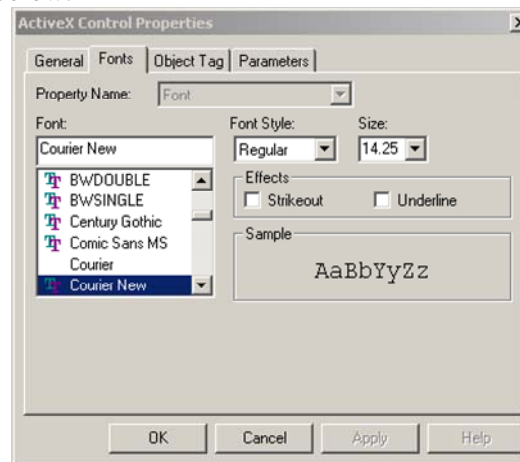
Font control beyond that configured by the ActiveX Control Properties pages described below can be made by directly using the **fontname** and **font** attributes in the following XML elements sent via the pipe:

```

<GRID />
<DISPLAY />
<TILE />
<MODTILE />
<TEXT />
<MODTEXT />

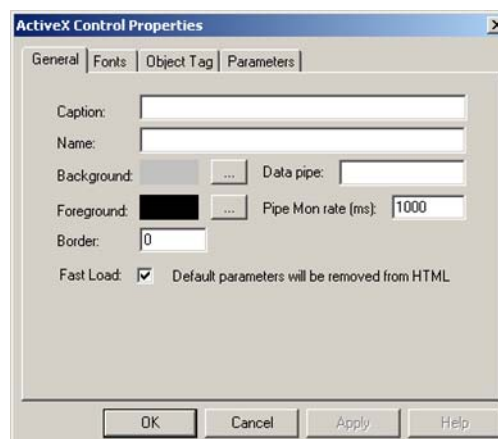
```

Font characteristics not specifically set on an element are inherited from its parent. TEXT elements inherit from TILES which inherit from the GRID or DISPLAY elements which inherit from the Grid control as defined by the property page below.



Item	Description
Font	Font specifies the name of the font passed in the Font property.
Font Style	Font Style can specify regular, bold, italic and italic bold. This affects whether the properties FontBold and FontItalic are optionally passed.
Size	Size specifies the size of the font. It is passed in the FontSize property.
Strikeout	Strikeout specifies whether the FontStrikethrough property is passed.
Underline	Underline specifies whether the FontUnderline property is passed.

8.2.3 General tab



8.2.3.1 Caption

Caption Is a text string that is passed as a property but not used. It is there as an aid to the developer since it is displayed on the SharePoint Designer 2007 design and preview views.

8.2.3.2 Name

Name Is a text string that specifies an association to a KAREL program on the controller.

Currently, when the Grid control is instantiated, the control checks for the existence of two program variables on the controller. These variables can be used to pass text commands in the format and with the content of other properties to dynamically affect the Grid control's operation.

[name]cmdstr If the variable exists and is of type STRING the control creates a monitor for this variable. Strings of commands can be written (set) to this variable and are delivered to the Grid control to affect 'dynamic' actions such as data updates or changes in some chart properties.

[name]cmdack If the variable exists and is of type BOOLEAN or INTEGER, the Grid control uses this variable to acknowledge that commands have been accepted via the **command** variable and acted on. The **cmdack** variable is set TRUE or 1 when that command is accepted by the plug-in.

NOTE

The **Name** parameter can utilize **!PanelID**. This is recommended for proper functioning in a multi-pane environment.

8.2.3.3 Foreground & Background Colors

The Colors dialog allows you to specify the color of certain elements. The *iPendant* supports 256 colors. Colors associated with the Grid control are:

ForeColor	Specify the "default" color of any entity that does not have an explicit color defined.
BackColor	Specify the background color.

NOTE

Colors are specified as a decimal value represent bbgrr (blue, green, red) format, where if they are set as hexadecimal value preceded by the # sign, they are in rrggbb (red, green, blue) format.

For example; ForeColor = 10531008 (decimal) is the same as ForeColor = #C0B0A0 (hex).

8.2.3.4 Pipe

Pipe Is a text string that specifies a named data file created on the controller and only associated with the PIP: device.

This file is used to dynamically deliver data to the Grid control on the *iPendant*. The name can be any 8.3-formatted name, for example: xgrid.dat.

The name of this file is sent to the controller when the Grid control is instantiated on the *iPendant*. The file name is concatenated to PIP: and the file opened. If the file does not exist then the file is created and

opened for read. If the file does exist prior to this, the file is opened for read access and a seek to the end of file done to eliminate the possibility of stale data.

When the last Grid control using this file is gone, the file is deleted.

NOTE

The **Pipe** parameter can utilize **!PanelID**. Which is recommended for proper functioning in a multi-pane environment? So specifying xgrid!PanelID.dat will allow a unique pipe file to exist on a per pane basis.

8.2.3.5 Pipe Mon Rate

PipeMonRate Is an integer value (>0) that specified the number if millisecond rate at which the controller checks for data in the pipe file.

The controller will enforce a maximum rate (most likely 100ms) as the lower end of the range.

Note: when multiple controls specify the same pipe file the fastest monitoring rate will take effect for all the controls monitoring this file.

8.2.3.6 Border

Border Selects the size of the border surrounding the Grid control.

Allowable values are:

- >0 A 3 dimensional raised border, that number of pixels thick.
- =0 No border line.
- <0 A 3 dimensional depressed border, that number of pixels thick.

8.2.3.7 FastLoad

FastLoad is selected by the checkbox.

Checked Specifies that properties that are set to their corresponding “default” values are not included in the web page. This effectively saves load time.

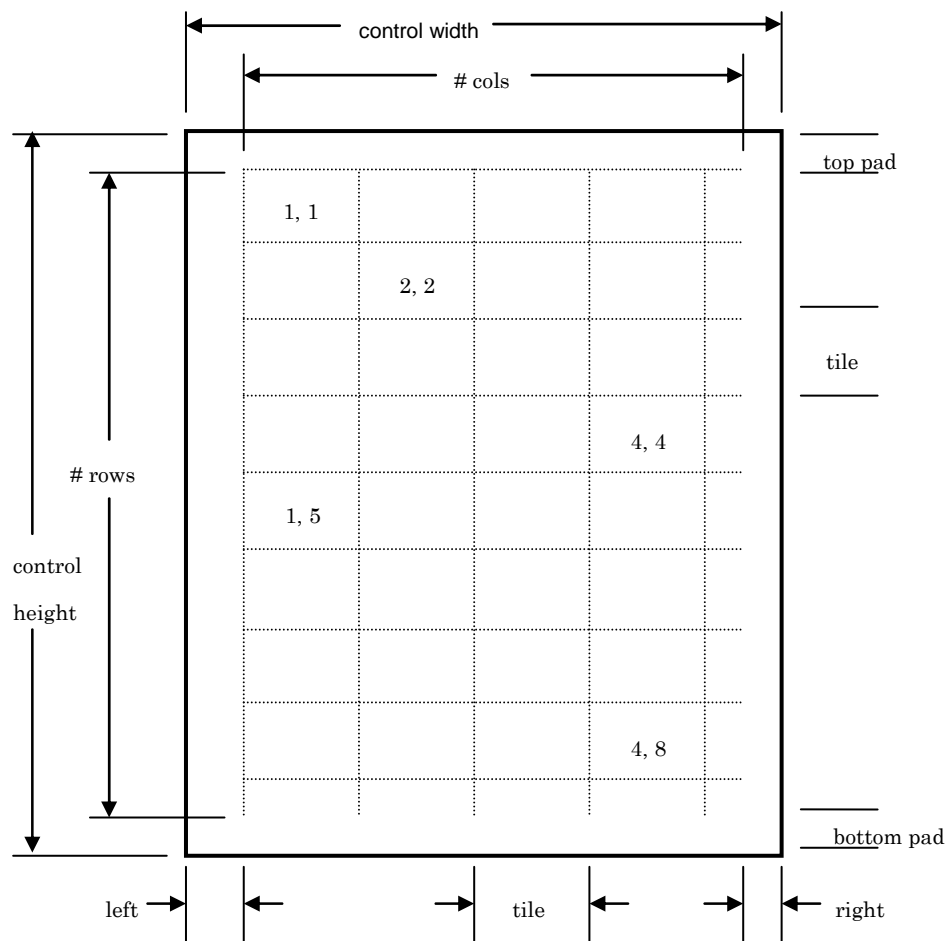
Unchecked Specifies that all parameters are included in the web page, whether they have “default” values or not.

8.3 DISPLAY CONCEPTS

The GRID control is simply a container within a web page to display graphics. It is placed on a web page specified in a .stm file. The .stm file is required so that the OBJECT syntax is properly translated to the EMBEDDED syntax when used with the *i*Pendant.

8.3.1 Conventions

The grid is laid out and referenced as follows:



The user can specify separate and/or optional padding each of the 4 sides independently. This padding is specified in # of pixels.

The GRID is then broken up into TILES, with attributes of:

- # columns wide in the horizontal direction. The origin TILE (1) being at the left,
- # rows high in the vertical direction. The origin TILE (1) being at the top,
- default foreground and or background color, otherwise the colors are inherited from those specified in the Grid Control,
- padding in # of pixels,
- touchable flag,
- version.

Each TILE has attribute(s):

- an assigned id string,
- row location within the grid,
- column location within the grid,
- default foreground and or background color, otherwise the colors are inherited from those specified in the GRID,
- each tile can contain SHAPES: images, text, lines, circles, rectangles, etc.
- each TILE can specify the layer that is currently displayed,.

Each SHAPE within a tile has attributes of:

- an assigned id string,
- location, relative to the tile. Negative offsets are allowed! Location is a % of the TILE dimension,

- size, IMAGES can be sized by width and height, if not specified the corresponding height and width are defaulted to the tile height or width,
- color, which if not specified is inherited from the TILE,
- and optionally, the SHAPE can belong to a TILE's layer 0 through 9. Layer 0 is the default and is always displayed.

NOTE

png files should not be used for images due to deficiencies in the Internet Explorer ActiveX components.

8.3.2 Alignment


SHAPES associated with a TILE are aligned relative to that TILE. The pixels are referenced relative to the tile and the origin (1,1) within the tile is selected as:

- X: 1 at top left going positive to the right
- Y: 1 at the top left going positive down.

For instance consider the following example tile that is 10 pixels wide by 9 pixels high.

Alignment presents issues depending on the size of the tile in pixels, especially if the height and width is an even number of pixels.

So the pixel at (A) in the TILE is the origin.

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
8										

8.3.3 Display, Pan and Zoom

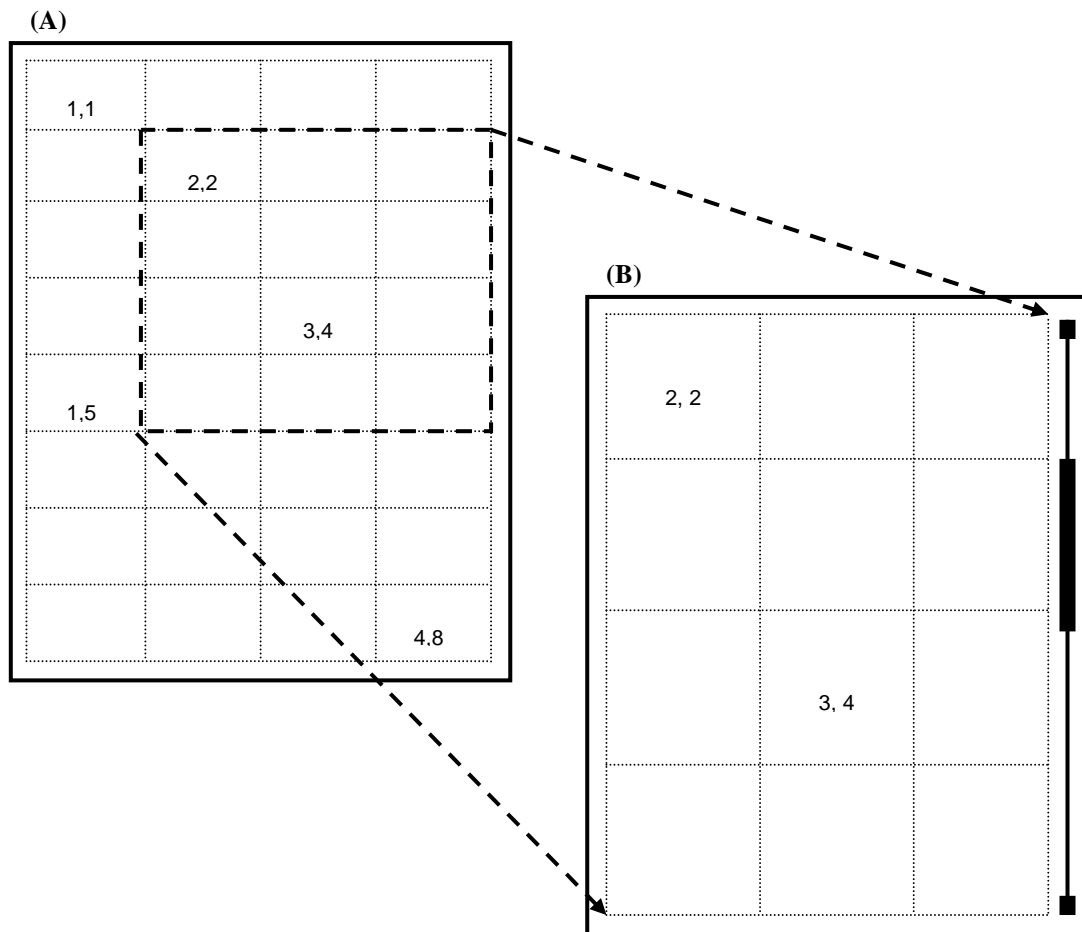
Since the SHAPES belong to TILES, and TILES belong to the GRID, the simple matter of moving around in the display is accomplished by specifying an origin TILE and dimensions to display.

Consider the following; a grid is initially created as 4 tiles wide by 8 tiles high is in example (A).

The XML string:

```
<DISPLAY pos="2,2" dims="3,4"/>
```

will cause the a subset of the GRDI to be display as in example (B).



The GRID may be displayed in its entirety as in (A). Here the whole GRID occupies most of the display real estate.

But as the GRID tile count is increased, TILES become smaller; the items in these TILES shrink. At some point there is too much information and the TILES become too small to provide recognizable information. So we need to display a part of the grid.

In (B) a part of the grid is selected for display. What parts and how much is determined by the management software as part of the application on the robot controller.

NOTE

Example (B) also has a scroll bar indicator to aid the user in navigating.

8.3.4 Dynamic Display

The GRID control can be dynamically updated. Either through the PIPE file mechanism or KAREL strings. Refer to section [8.4.15 Dynamic data](#)

There are many methods:

- TILES contain layers 0 through 10.
 - Layer 0 is always displayed.
 - Layers 1 through 10 can be turned on or off by the user using the MODTILE tag.
 - A SHAPE can be in any layer 0 through 10
- TILES may be deleted

- All SHAPES belonging to the TILE are automatically deleted.
- TILES may be modified
 - Changing parameters such as color and font characteristics
- SHAPES may be modified
 - Changing all parameters associated with the SHAPE

8.3.5 Rendering

As previously described the GRID contains TILES and each TILE contains SHAPES.

The actual rendering is not handled simply as a multi-dimensional array. There is a specific order to rendering, but it can be used to the programmer's advantage!

When a TILE is created, it is put into the tail of a linear list. TILES are processed in order in this list from beginning to end. So it is important when architecting the display to consider the order in which TILES are placed in the XML.

Next, a TILE itself can be thought of as a linear list of SHAPES. As each SHAPE is specified (created) on the TILE it is placed at the end of this list.

When the rendering for the tile is done, the linear list of SHAPES is processed. The SHAPES are individually handled in the following order:

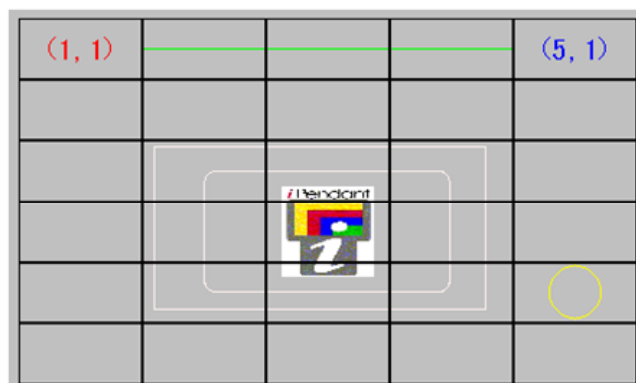
1. VT320
2. Button
3. Image
4. Circle
5. Rectangle
6. Text
7. Line

8.4 XML TO THE GRID

XML is standard Extensible Markup Language.

The Grid control understands a fairly strict XML syntax, a set of tags and attributes, as detailed in the following sections. Although the Grid control ignores unknown attributes, the user should be cautioned that, unknown attributes cause errors to be reported and can introduce delays in rendering or other *iPendant* problems.

As an example, the display produced in section 8 USING THE GRID CONTROL.



Is produced by the following XML:

(a)	<?xml version="1.0"?>
(b)	<GRID>
(c)	<TILE pos="1,1">
(d)	<TEXT font="+2" clr="#ff0000">(1,1)</TEXT>
(e)	<LINE clr="#00ff00" loc="100,50" eloc="400,50"/>
(f)	</TILE>
(g)	<TILE pos="5,1">
(h)	<TEXT clr="#0000ff" font="+2">(5,1)</TEXT>
(i)	</TILE>
(j)	<TILE pos="3,4">
(k)	<IMAGE size="75,150">ipend.gif</IMAGE>
(l)	</TILE>
(m)	<TILE pos="2,3">
(n)	<RECTANGLE clr="#fff0f0" align="top,left" size="200,200" radius="6"/>
(o)	< RECTANGLE clr="#fff0f0" align="top,left" loc="10,10" size="270,270"/>
(p)	</TILE>
(q)	<TILE pos="5,5">
(r)	<CIRCLE clr="#ffff00" diameter="90"/>
(s)	</TILE>
(t)	<DISPLAY Pos="1,1" DiMS="5,6" verbose="98" pad="6"/>

- (a) Is XML header information. This is not required for by GRID but is included for completeness.
- (b) Is the opening XML tag for the GRID.
- (c) Is the opening TILE tag for the TILE located at 1,1
- (d) Is TEXT "(1,1)" with font +2 more than the Grid control defaults, and in red
- (e) Is a LINE in green from the right side of the TILE (loc="100,50") for 400%.
- (f) Is the end tag for the TILE at 1,1
- (g) Is the opening XML tag for the TILE at 5,1
- (h) Is TEXT "(5,1)" with font +2 more than the Grid control defaults, and in blue
- (i) Is the end tag for the TILE at 5,1
- (j) Is the opening XML tag for the TILE at 3,4
- (k) Specifies and IMAGE ipend.gif, with a size of 75% the width of the TILE and 150% of the height
- (l) Is the end tag for the TILE at 3,4
- (m) Is the opening XML tag for the TILE at 2,3
- (n) Specifies a rounded RECTANGLE (corner radius="6"), aligned top and left, 200% of the TILE'S height and width
- (o) Specifies a RECTANGLE, aligned top and left, 270% of the TILE'S height and width
- (p) Is the end tag for the TILE at 2,3
- (q) Is the opening XML tag for the TILE at 5,5
- (r) Specifies a CIRCLE, in yellow, 90% of the TILE'S height
- (s) Is the end tag for the TILE at 5,5
- (t) Is the DISPLAY tag that causes the actual rendering of the TILES. This specifies that the TILE on pos="1,1" is the origin (upper left hand corner) and the display is 5 tile wide by 6 tiles high. Padding around the borders is 6 pixels. Note: verbose is the flag that turns on the TILE outlining.

The typical hierarchy of XML syntax, which the Grid control expects, is:

```

<?xml version="1.0"?>
<GRID>
  <TILE ...>
    <TEXT ...>string</TEXT>
    <LINE .../>
    <RECTANGLE .../>
    <CIRCLE .../>
    <IMAGE...>file</IMAGE>
    <BUTTON ... />
    <VT320 ... />
  </TILE>
  <DISPLAY .../>
  <DELTILE .../>
  <MODTILE/>
  <MODTEXT>string</MODTEXT>
  <MODLINE .../>
  <MODRECTANGLE .../>
  <MODCIRCLE .../>
  <MODIMAGE...>file</IMAGE>
  <MODBUTTON ... />
  <MODVT320 ... />
  <SETFOCUS ... />
</GRID>

```

8.4.1 XML tag

The following standard xml tag should be sent first:

```
<?xml version="1.0"?>
```

8.4.2 GRID tag

The GRID tag in the XML file data encapsulates a GRID and all of its sub-members. A GRID is composed of TILES and is X columns by Y rows. It has optional characteristics of color and padding to the display edges.

Prototype grid tag is:

```

<GRID id="userid" bclr="backcolor" fclr="forecolor" font="size" pad="#pixels" padl="#pixels"
padt="#pixels" padr="#pixels" padb="#pixels" padv="#pixels" padh="#pixels" versn="n"
verbose="#">
  .....
</GRID>

```

Optional parameters:

Attribute	Description
id="userid"	Specifies an optional user supplied ASCII identifier.
bclr="backcolor"	Specifies the background color. Supersedes the controls background color. If not specified then the control's background color is inherited.
fclr="forecolor"	Specifies the background color. Supersedes the controls foreground color. If not specified then the control's foreground color is inherited.
fontname="name"	The name of a Windows font recognized by the iPendant.

Attribute	Description
font="list"	The value for this property is a comma separated list decoded as follows: The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used. The remaining items are optional <ul style="list-style-type: none"> • b or -b to turn bold on/off • i or -i to turn italic on/off • s or -s to turn strikethrough on/off • u or -u to turn underline on/off • wxx to set the weight to xx • r to set bold, italic, strikethrough and underline off and weight to 0.
pad="#pixels"	Specifies general padding, for all sides of the display.
padl="#pixels"	Specifies padding for the left side of the display. Optional, if specified it overrides the general padding above.
padt="#pixels"	Specifies padding for the top of the display. Optional, if specified it overrides the general padding above.
padr="#pixels"	Specifies padding for the right side of the display. Optional, if specified it overrides the general padding above.
padb="#pixels"	Specifies padding for the bottom of the display. Optional, if specified it overrides the general padding above.
padv="#pixels"	Specifies vertical padding between TILES.
padh="#pixels"	Specifies horizontal padding between TILES.
touch="method"	Specifies that the GRID is touchable. Method is a bit mask: 1 = event on touch or mouse down, 2 = event on release or mouse up, 4 = event on BUTTON
verbose="#"	Specifies debug level. This is internal. 99 will put a 1 pixel wide outline around TILES for ease of alignment.

The <GRID> tag is the only permissible way to begin a grid.

Upon ending the grid with the </GRID> tag, the display is cleared and all TILES are freed.

NOTE

The user should always specify, **versn="1"** unless they are supporting a display prior to V7.30 and wish to keep exiting functionality.

8.4.3 TILE tag

A GRID contains TILES. Each tile is placed at a specific row and column within the GRID.

TILES contain SHAPES to be displayed. All SHAPES placed on a tile are located relative to the tile's origin, which is the upper left hand corner pixel of the tile. Columns go positive right and rows go positive down.

Prototype tile tag is:

```
<TILE id="user_ID" pos="col#,row#" layeron="layerlist" layeroff="layerlist" bclr="backcolor"
fclr="forecolor" font="size" sclr="color" slct="flag">
.....
</TILE >
```

Required parameters:

Attribute	Description
pos="col#,row#"	Specifies the position as column and row in the grid. Both rows and cols must be > 0.

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
layeron="layerlist"	Specifies a comma separated list of layers that are ON for this tile
layeroff=" layerlist"	Specifies a comma separated list of layers that are OFF for this tile. Layerlist = "*" is a special case that turns off all layers. Except of course layer 0 which is always displayed.
bclr="backcolor"	Specifies the background color. Supersedes the GRID's background color. If not specified then the GRID's background color is inherited.
fclr="forecolor"	Specifies the foreground color. Supersedes the GRID's foreground color. If not specified then the GRID's background color is inherited.
fontname="name"	The name of a Windows font recognized by the iPendant.
font="list"	The value for this property is a comma separated list decoded as follows: The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used. The remaining items are optional <ul style="list-style-type: none"> • b or -b to turn bold on/off • i or -i to turn italic on/off • s or -s to turn strikethrough on/off • u or -u to turn underline on/off • wxx to set the weight to xx • r to set bold, italic, strikethrough and underline off and weight to 0.
sclr=" color"	Specifies the tile's default select color. If not present DarkGray is assumed.
slct=" flag"	Flag designating the tile as selected. If not present the tile is not selected.

In general:

- The <TILE> tag must be enclosed within a <GRID> tag.
- A <TILE> tag can only encapsulate SHAPes as outlined in the following sections.
- A <TILE> can encapsulate multiple SHAPes.
- TILES can be deleted using the <DELTILE> tag.
- TILES can be modified using the <MODTILE> tag.

NOTE

Layerlist = "*" is a special case that turns off all layers. Except of course layer 0, which is always displayed.

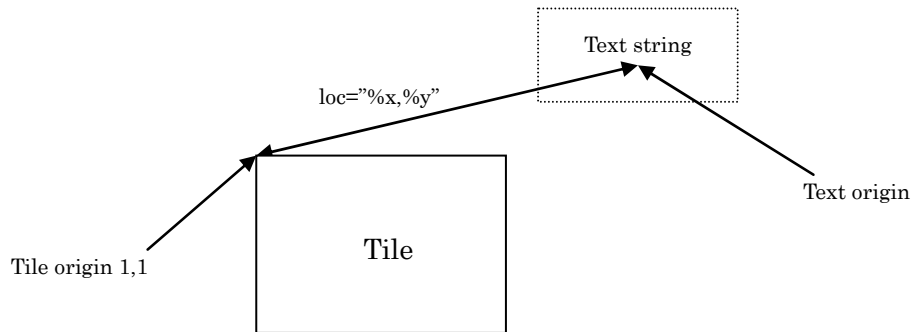
8.4.4 TEXT tag

A TEXT tag declares a text string to be displayed relative to the tile that it belongs to.

There are two origins to be concerned with:

- The TILE origin pixel (1,1) is at the upper left hand corner.
- The TEXT string origin, which the user can specify by the alignment argument. This argument specifies the vertical origin of the string as top, center or bottom and the horizontal origin as left, middle or right.

After that the string is offset by the % of the tile dimensions specified in the offset.



Prototype text tag:

```
<TEXT id="userid" clr="color" align="vert,horiz" font="size" loc="%x,%y" layer="layer#"
curs="index">Text string</TEXT>
```

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
clr="color"	Specifies text color. If not specified then the TILE's foreground color is inherited.
align="vert,horiz"	Specifies the vertical and horizontal alignment of the text string. Vertical values can be: <ul style="list-style-type: none"> • Top • Middle (default if not specified) • Bottom (These can be abbreviated as "t", "m" or "b") Horizontal values can be: <ul style="list-style-type: none"> • Left • Center (default if not specified) • Right (These can be abbreviated as "l", "c" or "r")
fontname="name"	The name of a Windows font recognized by the <i>iPendant</i> .
font="list"	The value for this property is a comma separated list decoded as follows: The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used. The remaining items are optional <ul style="list-style-type: none"> • b or -b to turn bold on/off • i or -i to turn italic on/off • s or -s to turn strikethrough on/off • u or -u to turn underline on/off • wxx to set the weight to xx • r to set bold, italic, strikethrough and underline off and weight to 0.
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
layer=" layer#"	Specifies the layer that the TEXT belongs to relative to the TILE. Values can be 0 to 9, if not specified then layer 0 is used.
curs="index"	Cursor index, will draw an underline at the specified index of the string. 0 = cursor off. 1 to strlen = cursor position. > strlen = cursor 1 character past the end of the string.
Text string	Is the text string to be displayed. If the string contains "¥n" (not a new line character), then the line is broken there and multi-line text can be written.

NOTE

In the case of multi-line text, the general horizontal and vertical alignment is extended on to each line.

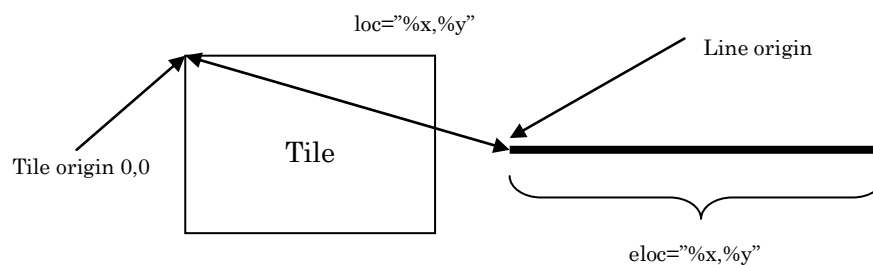
8.4.5 LINE tag

A LINE tag declares a line to be displayed relative to the tile that it belongs to.

There are two origins to be concerned with:

- The TILE origin pixel (1,1) is at the upper left hand corner.
- The LINE string origin.

After that the LINE is offset by the % of the tile dimensions specified in the offset.



Prototype text tag:

```
<LINE id="userid" loc="%x,%y" layer="layer#" clr="color" eloc="%x,%y" />
```

Required parameters:

Attribute	Description
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
eloc="%x,%y"	Specifies the end of the LINE relative to the TILE's origin. These values are % relative to the TILE's height and width.

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
layer="layer#"	Specifies the layer that the LINE belongs to relative to the TILE. Values can be 0 to 9, if not specified then layer 0 is used.
clr="color"	Specifies line color. If not specified then the TILE's foreground color is inherited.

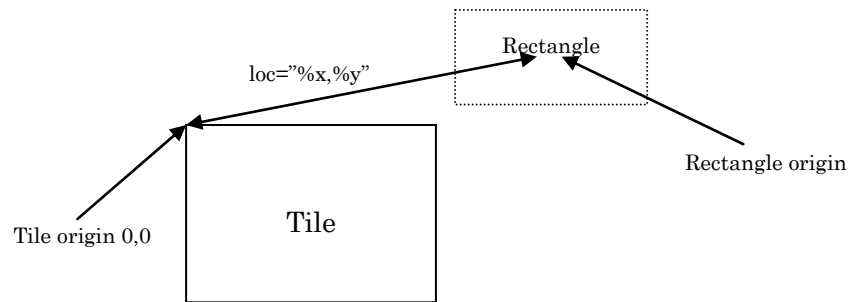
8.4.6 RECTANGLE tag

A RECTANGLE tag, which can be abbreviated as RECT, declares a rectangle to be displayed relative to the tile that it belongs to.

There are two origins to be concerned with:

- The TILE origin pixel (1,1) is at the upper left hand corner.
- The RECTANGLE origin, which the user can specify by the alignment argument. This argument specifies the vertical origin of the string as top, center or bottom and the horizontal origin as left, middle or right.

After that the string is offset by the % of the tile dimensions specified in the offset.



Prototype rectangle tag:

```
<RECTANGLE id="userid" clr="color" align="vert,hORIZ" loc="%x,%y" layer="layer#"
size="%width,%height[,radius]" wt="#pixels" fill="color" radius="radius"/>
```

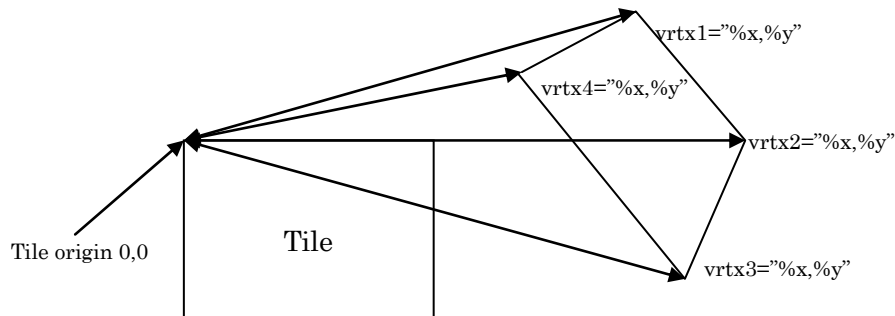
Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
clr="color"	Specifies line color. If not specified then the TILE's foreground color is inherited.
align="vert,hORIZ"	Specifies the vertical and horizontal alignment of the text string. Vertical values can be: <ul style="list-style-type: none"> • Top • Middle (default if not specified) • Bottom Horizontal values can be: <ul style="list-style-type: none"> • Left • Center (default if not specified) • Right
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
layer="layer#"	Specifies the layer that the RECTANGLE belongs to relative to the TILE. Values can be 0 to 9; if not specified then layer 0 is used.
size="%width,%height[,radius]"	Specifies the width and height as a % of the tile. And optional radius of the rectangle. Width and height are % relative to the tile's width and height. Optional radius makes this a rounded rectangle with corner radius of #pixels.
wt="#pixels"	Specifies the line weight in #pixels. If not present the default is 1.
fill="color"	Specifies the fill color for the RECTANGLE, also serves as a flag. If not present then the rectangle is not filled.
radius="#pixels"	Specifies a corner rounding radius in # of pixels. If not present then corners are not rounded.

8.4.7 POLYGON tag

A POLYGON tag declares a 3 or 4 sided polygon to be displayed relative to the tile that it belongs to.

The polygon is defined by three or four vertices, each located as a % of the tile dimensions relative to the tile's origin (upper left hand corner). A straight line is drawn between each successive vertex and from the last vertex to the first.



Prototype POLYGON tag:

```
<POLYGON id="userid" clr="color" align="vert,hORIZ" loc="%x,%y" layer="layer#"
size="%width,%height[,radius]" wt="#pixels" fill="color" radius="radius"/>
```

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
clr="color"	Specifies line color. If not specified then the TILE's foreground color is inherited.
vrtx1="%x,%y"	Specifies the offset to the first vertex relative to the TILE's origin.
vrtx2="%x,%y"	Specifies the offset to the second vertex relative to the TILE's origin.
vrtx3="%x,%y"	Specifies the offset to the third vertex relative to the TILE's origin.
vrtx4="%x,%y"	Specifies the offset to the fourth vertex relative to the TILE's origin. This is optional and if not present, a triangle is drawn.
layer="layer#"	Specifies the layer that the POLYGON belongs to relative to the TILE. Values can be 0 to 9; if not specified then layer 0 is used.
fill="color"	Specifies the fill color for the POLYGON. If not present then the rectangle is not filled.

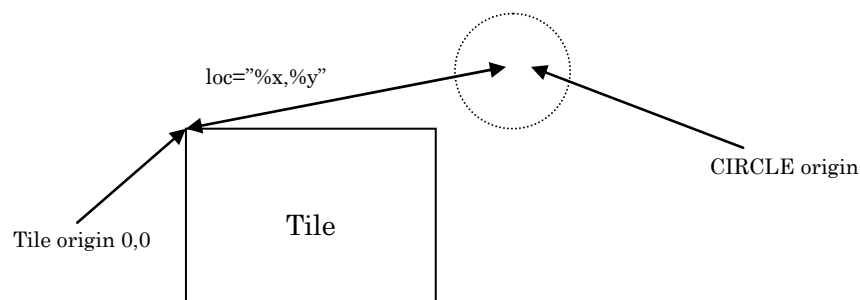
8.4.8 CIRCLE tag

A CIRCLE tag declares a circle to be displayed relative to the tile that it belongs to.

There are two origins to be concerned with:

- The TILE origin pixel (1,1) is at the upper left hand corner.
- The CIRCLE origin, which the user can specify by the alignment argument. This argument specifies the vertical origin of the string as top, center or bottom and the horizontal origin as left, middle or right.

After that the string is offset by the % of the tile dimensions specified in the offset.



Prototype circle tag:

```
<CIRCLE id="userid" clr="color" align="vert,hORIZ" loc="%x,%y"
```

```
layer="layer#" wt="#pixels" fill="color" diameter="%diameter"/>
```

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
clr="color"	Specifies line color. If not specified then the TILE's foreground color is inherited.
align="vert,horiz"	Specifies the vertical and horizontal alignment of the text string. Vertical values can be: <ul style="list-style-type: none"> • Top • Middle (default if not specified) • Bottom Horizontal values can be: <ul style="list-style-type: none"> • Left • Center (default if not specified) • Right
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
layer=" layer#"	Specifies the layer that the TEXT belongs to relative to the TILE. Values can be 0 to 9, if not specified then layer 0 is used.
wt="#pixels"	Specifies the line weight in #pixels. If not present the default is 1.
fill="color"	Specifies the fill color for the RECTANGLE, also serves as a flag. If not present then the rectangle is not filled.
diameter="%diameter"	Specifies the CIRCLE's diameter as a % of the TILE it is contained in. If not present then 100% is assumed.

8.4.9 IMAGE tag

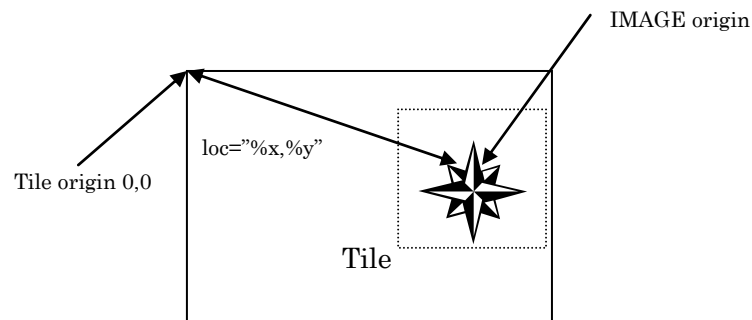
An IMAGE tag places an IMAGE (.jpg or .gif) into a tile for display.

There are two origins to be concerned with:

- The TILE origin pixel (1,1) is at the upper left hand corner.
- The IMAGE origin, which the user can specify by the alignment argument. This argument specifies the vertical origin of the string as top, center or bottom and the horizontal origin as left, middle or right.

After that the string is offset by the % of the tile dimensions specified in the offset.

The size of the IMAGE relative to the tile can be specified as percentages of the size of the TILE. If not specified the IMAGE is sized to 100% of the TILE.



Prototype IMAGE tag:

```
<IMAGE      id="userid"      clr="color"      align="vert,horiz"      loc="%x,%y"
size="%width,%height">image.gif</IMAGE>
```

Required parameters:

Attribute	Description
image.gif	Specifies the file name of the image. This can be absolute or URL relative. Allowable file types are .gif and .jpg

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
clr="color"	Specifies the color show if there is transparency in the image.
align="vert,horiz"	Specifies the vertical and horizontal alignment of the text string. Vertical values can be: <ul style="list-style-type: none"> • Top • Middle (default if not specified) • Bottom Horizontal values can be: <ul style="list-style-type: none"> • Left • Center (default if not specified) • Right
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
layer="layer#"	Specifies the layer that the TEXT belongs to relative to the TILE. Values can be 0 to 9, if not specified then layer 0 is used.
size="%width,%height"	Specifies the width and height as a % of the tile. Width and height are % relative to the tile's width and height. If the size is not specified then the 100% is assumed.

NOTE

- .png files should not be used for images due to deficiencies in the Internet Explorer ActiveX components.
- For proper color rendering image files should be created with a Standard Windows Color Pallet.
- For a .gif file specifying a transparency color, the transparent color can be specified by the **clr** parameter, or inherited from TILE's **bclr** parameter.

8.4.10 BUTTON tag

BUTTONs are intended to be used just like a panel push button. A BUTTON can be constructed in such a way that it conveys information or state as well as accepting an action.

The information or state is pretty much under the programmer's control. The BUTTON tag defines the basic framing around the button shape. The content can be other SHAPES placed on the BUTTON. These include TEXT, RECTANGLES, CIRCLES, IMAGES, etc. The programmer then has control of these SHAPES as well as colors.

If the *iPendant* is capable of touch then touch sense is maintained by the *iPendant* control itself.

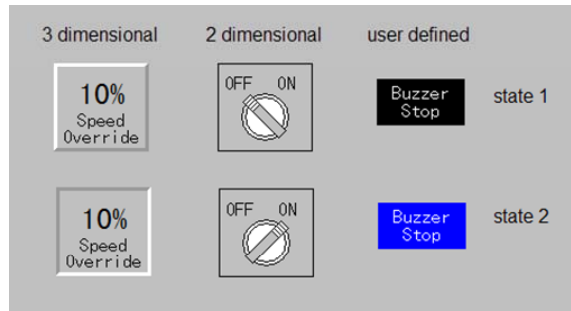
- Release or mouse up is sensed and can be sent back to the controller in a manner specified by any or all of the 3 parameters: kcod, tprq or url.

- Press or mouse down is sensed and can be sent back to the controller in a manner specified by any or all of the 3 parameters: kcod2, tprq2 or url2.

If the *i*Pendant is not capable of TOUCH then the programmer can monitor the TP arrow keys as a means of navigating and selecting a BUTTON on the display. The ENTER key can then be used to select the BUTTON.

Buttons have the same concept of alignment as the other SHAPES.

A few examples of BUTTONS and different states are:



Prototype BUTTON tag:

```
<BUTTON id="userid" bclr="color" align="vert,hORIZ" loc="%x,%y" layer="#"
size="%width,%height" latch="state" kcod="#" tprq="#1,#2" url="urlstring" kcod2="#"
tprq2="#1,#2" url2="urlstring"/>
```

Optional parameters:

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.
bclr="color"	Specifies the background color.
align="vert,hORIZ"	Specifies the vertical and horizontal alignment of the text string. Vertical values can be: <ul style="list-style-type: none"> • Top • Middle (default if not specified) • Bottom Horizontal values can be: <ul style="list-style-type: none"> • Left • Center (default if not specified) • Right
loc="%x,%y"	Specifies the offset relative to the TILE's origin. If not specified then the TILE's origin (1,1) is used. These values are % relative to the TILE's height and width.
layer=" layer#"	Specifies the layer that the TEXT belongs to relative to the TILE. Values can be 0 to 9, if not specified then layer 0 is used.
size="%width,%height"	Specifies the width and height as a % of the tile. Width and height are % relative to the tile's width and height.
look="style"	Specifies the button style: <ul style="list-style-type: none"> 0 = not a button 1 = user defined border 2 = 2-dimensional border 3 = 3-dimensional border

Attribute	Description
latch="state"	Specifies that the button is latchable, or controlled by the user. State is one of: 0 = button is not currently pressed 1 = button is currently pressed
nolatch="state"	Specifies that the button is not latchable. State is meaningless but must be supplied for XML completeness.
kcod="#"	Specifies a 1 byte key code value that is sent back to the controller on the release or mouse up event
tprq="#1,#2"	Specifies a 2 byte value that is sent back to the controller on the release or mouse up event. This is intended for system use and should not be used otherwise.
url="urlstring"	Specifies a URL string to be sent back to controller's the web server on the release or mouse up event. Some examples are: url="KCL/show%20var%20\$version" url="/KCLDO/reset" url="/KCLDO/set%20port%20tpout[1]=1">KCLDO set port tpout[1]=1 url="/KCLDO/set%20var%20[program]somevar%20"
kcod2="#"	Specifies a 1 byte key code value that is sent back to the controller on the press or mouse down event.
tprq2="#1,#2"	Specifies a 2 byte value that is sent back to the controller on the press or mouse down event. This is intended for system use and should not be used otherwise.
url2="urlstring"	Specifies a URL string to be sent back to controller's the web server on the press or mouse down event. Examples can be found under the description of url, above.

8.4.11 DISPLAY tag

The display tag defines what part of the whole grid is visible on the display. This is how pan and zoom is implemented. (See section [8.3.3 Display, Pan and Zoom](#)).

For example, a GRID may have been defined as 100 by 200 and its tiles populated. This is too large to display any amount of detail that the user might make sense of.

The DISPLAY tag defines the origin tile and a X,Y extent, as a dimension, to display. The origin tile will be the top left TILE actually displayed. And the ending extent TILE will be in the lower right.

Additionally the DISPLAY tag can supersede the default font, pad, foreground and background colors, and specify display of scroll bars to aid the user in visualizing where they are within the whole grid.

Prototype DISPLAY tag:

```
<DISPLAY pos="col#,row#" dims="#cols,#rows" bclr="color" fclr="color" font="size"
pad="pixels" padt="pixels" padr="pixels" padb="pixels" padl="pixels" padh="pixels"
padv="pixels" scrollbar="style" hscroll="thick,pos,amount" vscroll="thick,pos,amount"/>
```

Required parameters:

Attribute	Description
pos="col#,row#"	Specifies the position as column and row in the grid that is the origin to be displayed.

Optional parameters:

Attribute	Description
dims="#cols,#rows"	Specifies the dimensions, # columns and # rows to be displayed.
bclr="color"	Specifies the tile's background color for all tiles in the range of pos by dims. It supersedes the TILE's background color. If not specified then the TILE's background color is not affected.
fclr="color"	Specifies the tile's foreground color for all tiles in the range of pos by dims. It supersedes the TILE's foreground color. If not specified then the TILE's foreground color is not affected.
fontname="name"	The name of a Windows font recognized by the iPendant.
font="list"	The value for this property is a comma separated list decoded as follows: The first item is the font size. If a + or – precedes the number, then the value is saved as a separate increment/decrement to be applied to the font size when it is used. The remaining items are optional <ul style="list-style-type: none"> • b or -b to turn bold on/off • i or -i to turn italic on/off • s or -s to turn strikethrough on/off • u or -u to turn underline on/off • wx to set the weight to xx • r to set bold, italic, strikethrough and underline off and weight to 0.
pad="#pixels"	Specifies general padding, for all sides of the display.
padl="#pixels"	Specifies padding for the left side of the display.
padt="#pixels"	Specifies padding for the top of the display.
padr="#pixels"	Specifies padding for the right side of the display.
padb="#pixels"	Specifies padding for the bottom of the display.
padh="#pixels"	Specifies horizontal padding between TILES.
padv="#pixels"	Specifies vertical padding between TILES.
scrollbar="style,bgcolor,fgcolor"	Specifies the style 1, through 10. 0 = No scrollbar.
hscroll="thick,offset,amount"	Specifies the horizontal scroll bar characteristics. Refer to section 8.4.11.1 Scroll bars with the DISPLAY tag.
vscroll="thick,offset,amount"	Specifies the vertical scroll bar characteristics: Refer to section 8.4.11.1 Scroll bars with the DISPLAY tag.

So we might want to only display a 10 by 12 area within the grid. The DISPLAY tag does that.

An example would be:

```
<DISPLAY pos="1,4" dims="10,12"/>
```

This starts at the tile located at 1,4 within the GRID and displays TILES 10 columns wide by 12 rows high.

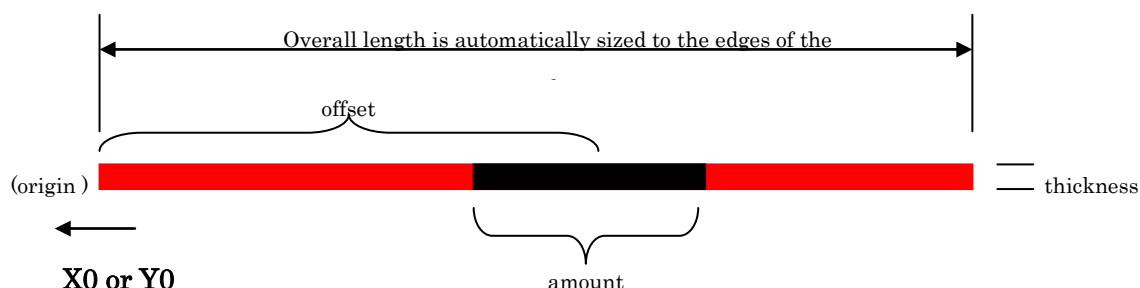
8.4.11.1 Scroll bars with the DISPLAY tag

Additionally the display tag allows the user to display scroll bars along the right side of the control and at the bottom.

Scroll bars are normally used to represent where the current offset is within the total possible display.

The horizontal scrollbar and vertical scrollbar are completely independent and under the control of the user, as XML content.

Scroll bars are specified as part of the <DISPLAY> tag because at the time the DISPLAY command is issued the user should have a good handle of what portion and offset within the total they are showing.



Prototype DISPLAY tag:

```
<DISPLAY . . . scrollbar="style,bgcolor,fgcolor" hscroll="thickness,offset,amount"
vscroll="thickness,offset,amount"/>
```

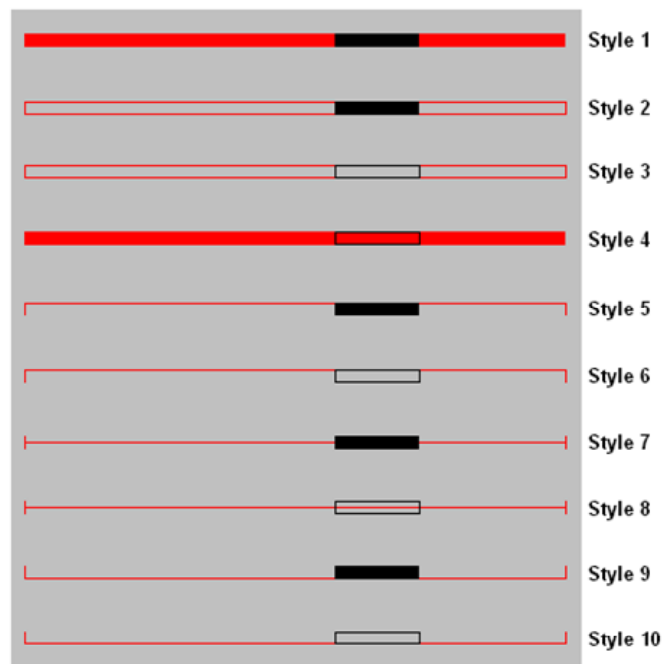
Optional parameters to the DISPLAY tag:

Attribute	Description
scrollbar="style,[bgcolor[,fgcolor]]"	Specifies a style of scroll bar and optionally a foreground and background color. Valid styles are 1 through 10 as depicted below.
hscroll="thickness,[offset[,amount]]"	Specifies the horizontal scrollbar characteristics: <ul style="list-style-type: none"> • thickness is # of pixels wide. • offset is the center of the scroll slider from the origin in % • amount is the length of the scroll slider in %
vscroll="thickness,[offset[,amount]]"	Specifies the vertical scrollbar characteristics: <ul style="list-style-type: none"> • thickness is # of pixels wide. 0 indicates no scrollbars. • offset is the center of the scroll slider from the origin in % • amount is the length of the scroll slider in %

A **thickness** of 0 on any scrollbar means, do not display it.

Offset must be between 0 and 100%.

Amount must be between 0 and 100%.



8.4.12 DELTILE tag

The GRID plug-in maybe sent many TILES that are not being displayed or may not even be relevant anymore.

The DELTILE tag is implemented to delete unneeded TILES. This also frees up internal memory that the plug-in is holding onto.

Prototype DELTILE tag:

```
<DELTILE id="userid" pos="col#,row#" dims="#cols,#row"/>
```

Required parameters:

Either

Attribute	Description
id="userid"	Specifies a user supplied ASCII identifier.

or

Attribute	Description
pos="col#,row#" cant="0,0"	Specifies the position as column and row in the grid that is the origin to be deleted.
dims="#cols,#rows"	Specifies the dimensions, # columns and # rows to be deleted.

or both

As with the previous example the GRID control may have been sent hundreds of TILES, for example a display of 100 by 200.

- If we want to clear row 10 entirely:
<DELTILE pos="1,10" dims="100,1"/>
- If we want to clear column 5 entirely:
<DELTILE pos="5,1" dims="1,200"/>

- If we want to clear all tiles from column 20 & row 10 through column 90 & row 190:
<DELTILE pos="20,10" dims="71,181"/>

8.4.13 Modifying SHAPES

When a SHAPE is created, it is done in the context of being associated with a TILE. A requisite to the GRID control being useful is to be able to modify or change the characteristics of SHAPES being displayed. And to do so dynamically!

One way to accomplish this would be to delete the TILE that the SHAPE is associated with and recreate the necessary TILE and SHAPE. But remember the TILE may have many associated SHAPES and it is most likely a waste of time and processor since we would have to recreate many SHAPES. And of course there are more complication when rendering order is important.

Recall that all the SHAPE tags have a user specifiable identifier, **id="userid"**. This userid string is:

- a non-case sensitive string that should uniquely identify a shape.
- it can contain space characters.

Each SHAPE tag has a corresponding <MODxxxx.....> tag that will accept the same parameters as the definition tag.

For example, the TEXT tag :

```
<TEXT id="string 1" clr="#0000ff" font="10" loc="50,50" />Normal</TEXT>
```

Can be modified by :

```
<MODTEXT id="string 1" clr="#ff0000 font="+2,i" />Alarm</TEXT>
```

This changes the black text from "Normal" to "Alarm" in red in a font 2 points larger and Italic.

The following is the correlation of the definition tag to the modifier tag.

Definition tag	Modification tag
<TEXT ...	<MODTEXT ...>
<LINE ...	<MODLINE ...>
<RECTANGLE ...	<MODRECTANGLE ...>
<RECT ...	<MODRECT ...>
<CIRCLE ...	<MODCIRCLE ...>
<IMAGE ...	<MODIMAGE ...>
<BUTTON ...	<MODBUTTON ...>

Please consult the corresponding section of the definition tag to verify the proper parameters. In the modification tag if a parameter is not specified, then the original parameter setting is left unchanged.

8.4.14 Modifying TILES

In the same way that SHAPES can be modified, TILE can be too. Remember that a TILE is a container that can have many SHAPES associated with it. Careful structuring of the whole display can mean that a simple modification to a TILE can affects a large number of SHAPES, either through the use of layers or using inherited characteristics such as font, colors, etc.

The TILE definition tag has two ways of identifying the TILE. Using the **id="userid"** and the combination **pos="col#,row#"** and **dims="#cols,#rows"** parameters.

Like the SHAPE tag, the userid string is a non-case sensitive string that should uniquely identify a TILE. The id parameter is optional, the pos parameter is not.

Using these two parameters we can modify a TILE in one of 3 ways:

- 1) Specifying only the id parameter:

```
<MODTILE id="tile #12" layeroff="*" layeron="1" />
```

Will modify the TILE with the id of "tile #12" to turn off all layers 1 through 10, and turn on layer 1.

- 2) Specifying only the pos:

```
<MODTILE pos="1,12" layeroff="*" layeron="1"/>
```

Will modify the TILE at position 1,12 to turn off all layers 1 through 10, and turn on layer 1.

```
<MODTILE pos="2,2" dims="2,3" bclr="ff0000"/>
```

Will modify all the TILES starting at position 2,2 in a rectangular array down to the TILE at position 2,3 modifying their background color to RED.

- 3) Specifying both the id and pos:

```
<MODTILE id="panel" pos="1,1" dims="2,5" layeron="2"/>
```

This will turn on layer 2 for all tiles with the id "panel" in the TILES starting at position 1,1 through position 2,5.

8.4.15 Dynamic data

The following parameters are available but have limited use, except for dynamic rendering.

8.4.15.1 Using the Name Parameter

If a **Name** is designated for a chart, then KAREL variables as described for the **Name** parameter can be used to deliver dynamic commands to the Grid control. For example, a Grid with Name = grid1

```
KCL> set def grid1
KCL> create var command string[127]
KCL> create var cmdack integer
```

Go to the web page that creates grid1

```
KCL> set var command = '....' -- Will deliver this data to the drawing.
```

8.4.15.2 Using the Pipe parameter

The **Pipe** parameter of the Grid control specifies a data file located on the PIP: device. Once created, the controller will attempt to deliver data written to this pipe file to the control(s) using it.

8.4.15.3 Using the Data parameter

The data parameter of the web page can also be used to get XML content to the Grid control.

For example within the <OBJECT . . . in the .stm file:

```
<param name="data" value='<GRID><TILE pos='1,1'><TEXT font='+2'
clr='#ff0000'>(1,1)</TEXT><LINE clr='#00ff00' loc='100,50' eloc='400,50'></TILE><TILE
pos='5,1'><TEXT clr='#0000ff' font='+2'>(5,1)</TEXT></TILE><TILE pos='3,4'><IMAGE
size='75,150'>ipend.gif</IMAGE></TILE><TILE pos='2,3'><RECTANGLE clr='#fff0f0'
align='top,left' size='200,200' radius='6'>< RECTANGLE clr='#fff0f0' align='top,left'
loc='10,10' size='270,270'></TILE><TILE pos='5,5'><CIRCLE clr='#ffff00'
diameter='90'></TILE><DISPLAY Pos='1,1' DiMS='5,6' verbose='98' pad='6'>'>
```

Notice that several special things must be done:

- The data value itself must be framed in double quotes “
- Since the data value must be framed in double quotes, any double quotes in the XML content should be converted to single quotes. The GRID control internally will translate single quotes to double quotes before processing the “data” parameter.
- Since single quotes are translated to double quotes, single quotes cannot be used.

Additionally the entire content of the data value parameter could be a separate file that is included with server side includes:

```
<param name="data" value="<!--#include file=data.xml -->">
```

Here, the file data.xml is included is the data stream to the Grid control.

9 PANE LINKING

9.1 INTRODUCTION

A set of utilities and features is provided for enhancing connecting together Web pages on the *i*Pendant. These features fall into a number of different categories:

- Display Menu Functions
- Generic Linking Functionality
- Context Sensitive Help
- Related Views

This document includes some information on examples. It is much easier to understand the functionality by looking at some of these examples. These are simple to set up and should provide a good basis for understanding the potential of this functionality.

Generic documentation of the display menu and its use can be found in the operations manual. This document has some redundant information but most of this is more detailed to allow a user to tailor system operation in this area.

9.2 DISPLAY MENU FUNCTIONS

All of the entries in the display menu can be displayed or masked by setting the system variable \$UI_CONFIG.\$DSPMEN_MASK. For the default case not all of these entries are available.

“Related Views” is a new concept that is explained in a later section. It is fully implemented but there is no related view content at this time. Therefore it is masked out.

“Help/Diagnostics” provides the same functionality as the help key plus allows the user to bypass context sensitive help and get to the root help pages.

“User Views” allows the specification of a specific *i*Pendant configuration. That is the number of windows and the menu page that is displayed in each window. These user views can be set up via system variable settings. If you run the procedures outlined in the example section you can see a set of user views that are set up via system variables. User Views are cleared in the *i*Pendant setup screen

For menu history and menu favorites a menu is considered to be any page of the robot menu system. This is a bit different than a browser favorite or browser history that provides this information or Web pages. A menu page is both a specific softpart and a specific screen, or for the case of editing, a specific program. Any time the user selects an edit page from favorites or history he will be in the teach pendant editor in the program that he was editing when the favorite or history entry was created.

“Menu Favorites” allows any user to select a menu to put into his favorites. These are then accessible for a quick link to any menu in the system. This feature is only available at COLD start with FULL menus in force. Favorites are cleared in the *i*Pendant setup screen.

“Menu History” provides a list of the eight most recently accessed menus. This allows a savvy user to quickly return to a recently accessed menu.

9.3 COMMANDING LINKS FROM KAREL

A built-in `FORCE_LINK` is provided to command these links from a KAREL program. This built-in is fully described in Appendix E: KAREL built-ins.

This allows the KAREL user to have complete control over the display screen from a programming standpoint. This allows for Wizards and other complex Web based programs to be written in KAREL.

9.4 GENERIC LINKING FUNCTIONALITY

All of these capabilities are built on top of generic linking. Generic linking provides the following key capabilities:

- Specification of any menu page in the system as a URL.
- Specification of any program edit as a URL.
- Specification of any of the three “panes” on *iPendant* as the destination for a link.
- Specification of links into multiple panes from a single complex link
- Specification explicitly or implicitly of any display configuration.

By bringing our menus in as URL’s we can now specify links in a generic way that can either be Web pages in the traditional sense or our legacy menus. This provides some very interesting capabilities.

The exact syntax of these is in a section to follow. The examples are full of these links.

9.5 CONTEXT SENSITIVE HELP

Context sensitive help allows the application to specify a particular Web page as a help page at a particular instance in time. This means that help key will bring up the Web page specified for help at that instant. This is called “context sensitive” help.

If the user does not specify a help page the default page be brought up when the user hits the help key. This default page is currently specified based on screen ID and softpart ID. Context sensitive help is menu page specific. This means that when the user selects a different menu or Web page, the context sensitive help information is cleared.

Placing a `HELP` control on a Web page will specify context sensitive help for that page. This causes the URL specified in the `HELP` control to be activated when the user hits the help key. The default help in this case would be “Web Browser Help”. For a simple Web page the context is the page itself.

More typically context sensitive help is dependent on the exact place in the menu that the user is looking at. For example, context sensitive help in the `CELLIO` screen might provide help information on the specific I/O point the cursor is on. For this type of dynamic help content the program running the screen needs to make system calls to update the dynamic help URL.

From a KAREL program you can set the context help URL via:

```
$UI_PANELINK[connect_id].$HELP_URL[device_id] = 'some help url'
$UI_PANELINK[connect_id].$HELP_FLAG[device_id] = 1
```

The C system call is:

```

void tplink_set_help(UBYTE conn_id, /* Zero based connect ID */
                    UBYTE device_id,
                    char *partial_url, /* Partial URL to be forced */
                    ULONG flags, /* URL fully qualified? */
                    ULONG context_id) /* context information if applicable
*/

```

See the section “About identifiers” for details on connect_id and device_id.

9.6 ABOUT IDENTIFIERS

When working with links within multiple panes there is some terminology that is used to indicate which “instantiation” of a particular application we are talking about. With Internet Explorer connections it is possible to have up to 4 Internet Explorer sessions connected. Each Internet Explorer session can have a total of three windows active. Conceivably, your application can appear in all windows on all connections. The total number of displayed windows is actually limited to nine.

It is possible for a single KAREL program to manage all of these connections if the KAREL program always knows which instance of the menu is of interest. For example, when you set the context sensitive help system variables, they need to be set based on your connection.

In order to manage these three indexes are used in the system. They are: connect_id, device_id and pane_id. The connect_id is *i*Pendant or a particular browser attachment. For *i*Pendant the connect_id is always 0. For Internet Explorer connections the connect_id is in the range of 1 to 4 but you can’t tell by looking at the Internet Explorer session which connect ID is being used.

The device_id identifies one of the three panes that can be displayed on a connection. In single pane, the device_id is always zero. In dual pane the right hand pane is 1 and the left pane is zero. The third pane in triple mode is 2.

The pane_id is a number between one and nine which uniquely identifies the instance. This allows an application to have arrays of length nine where the pane_id is the index into that array. The real interpretation of pane_id is that this designates which system task (there are nine). The system variable \$UI_PANEDATA[9] contains information about the status of all pages.

The pane_id is the same value as the device_stat. If a menu is built into the system then the device_stat KAREL variable will be set by the system. You can also set the device_stat variable in order to instruct the system how to treat your menu.

The built-in GET_DEVINFO is provided to return all the information to the KAREL user. This is fully described in Appendix E: KAREL Built-ins.

In C all three of these numbers are stored in tpglobals as:

```

UBYTE conn_id; /* CRT model, TP model or remote browse */
UBYTE device_id; /* Device which this task services */
UBYTE uif_task_idx; /* Assigned index for task/pane tables */

```

In KAREL the pane_id is always provided by the execution control. From the pane_id you can determine the connect_id and the device_id. This is a bit complicated but it is easy to find the pane_id from the connect_id and the device_id (in KAREL) via:

```

Pane_id = $UI_CONFIG.$PANEMAP[device_id*3 + pane_id + 1]

```

In C:

```
UBYTE tplink_panenum(BYTE conn_id, /* Zero based connect ID */
                     UBYTE device_id)
```

9.7 NEW/MODIFIED SYSTEM VARIABLES

Several new system variables have been added in support of this functionality. The details of specific fields are covered in the reference manual sections. The variable \$UI_USERVIEW is not covered in a manual section so is fully documented here.

9.7.1 \$TX

\$TX system variable contains information on the Teach Pendant.

System variable	Description
\$TX.\$CONNECTED	If TRUE, some version of the <i>i</i> Pendant is connected. If FALSE, the legacy pendant is connected.
\$TX.\$TP3D	If TRUE, the R-30iB <i>i</i> Pendant is connected. The virtual <i>i</i> Pendant on a PC will also have this setting. If FALSE, the R-30iA <i>i</i> Pendant or legacy pendant is connected.
\$TX.\$PC	If TRUE, the virtual <i>i</i> Pendant on a PC is connected. If FALSE, the real <i>i</i> Pendant is connected.

9.7.2 \$ALM_IF

\$ALM_IF system variable can be used to display the last alarm or last user alarm from an *i*Pendant Control

System variable	Description
\$ALM_IF.\$ENABLE	If TRUE, the current alarm text is copied to \$ALM_IF.
\$ALM_IF.\$LAST_ALM	The text for the most current alarm.
\$ALM_IF.\$LAST_UALM	The text for the most current user alarm or facility code KALM
\$ALM_IF.\$KALM_MAX	The maximum alarm number of KALM alarm that is output to \$ALM_IF.\$LAST_UALM. For example, if \$ALM_IF.\$KALM_MAX is 100, then text of KALM-100 is output to \$ALM_IF.\$LAST_UALM when the alarm happens. Text of KALM-101 is not output.

9.7.3 \$UI_USERVIEW

This variable contains information on the content of all three-user windows and the configuration. By setting this variable a user can set up a pre-configured window configuration. This variable is used to contain the user views that are set by the —Add Current— selection in the User Views menu under display. The *i*Pendant setup menus can be used to clear the user views.

To better understand this system variable you can set it from the menu and then examine what the system sets as the value.

System variable	Description
\$MENU	The ASCII text which appears in the fly out menu.
\$CONFIG	Configuration, Whole, Double, Triple or Status/Single
\$FOCUS	Pane which has focus.
\$PRIM	URL for left pane.
\$DUAL	URL for right pane if double, upper left pane if triple or status area for single status.
\$TRIPLE	URL for lower right pane.

Note that the URL's can be standard Web URLs or special menu URL's as described above.

9.7.4 \$UI_CONFIG

This is a preexisting variable that contains some new fields for this release. Of specific interest are:

System variable	Description
\$PANEMAP	Provides relationship between pane_id, context_id and device_id.
\$MENU_FAVS	URL's for menus in favorites fly out.
\$HLPMEN_*	Definition of entries in HELP fly out which are by default not occupied
\$DSPMEN_MASK	Mentioned above configures what items are displayed in the display menu. This is bit mapped, setting it to 511 will enable all menus.
\$BACKCOLOR	<i>i</i> Pendant background color. Use the <i>i</i> Pendant Setup <i>i</i> Pendant Color Setup menu to change the background color. A new SSI is created to use the system variable, but only if your web page is .STM. <body bgcolor="<!-- #echo var=_BGCOLOR -->"> becomes <body bgcolor="#DCF0FF">

9.7.5 \$UI_MENHIST[5]

The index into this variable in the connect index. So \$UI_MENHIST[1] contains menu history information for *i*Pendant. 2-5 contains the history information for the other connections.

History is saved as a URL to the menu plus a head index.

9.7.6 \$UI_PANEDATA[9]

The index into this variable is the pane_id. Up to 9 panes system wide can be active at any one time. This variable contains output information about what is currently executing in that pane. When the KAREL execution control is started up, the pane ID is provided to the KAREL program.

Note that the execution control passes up to 8 string parameters via this system variable. Fields in panedata are:

System variable	Description
\$PAGEURL	URL specification of the pane currently loaded into the frame
\$FRAME	Name of the frame the URL is in
\$HELPURL	URL which is posted when user hits help key (set by HELP control)
\$PARAMETER1	String parameter to execution partner
\$PARAMETER2	String parameter to execution partner
\$PARAMETER3	String parameter to execution partner
\$PARAMETER4	String parameter to execution partner
\$PARAMETER5	String parameter to execution partner
\$PARAMETER6	String parameter to execution partner
\$PARAMETER7	String parameter to execution partner
\$PARAMETER8	String parameter to execution partner
\$INTERVAL	Dynamic update interval
\$PANESTATE	State of the pane, UI_LOADED means the page is completely loaded. You should not interact with controls on the page until this is set.

9.7.7 \$UI_PANELINK[5]

The index into this variable in the connect index. So \$UI_PANELINK [1] contains link information for *i*Pendant. 2-5 contains the link information for the other connections.

System variable	Description
\$HELP_URL[3]	Context help URL, one per device
\$HLP_CONTEXT[3]	Context identifier for help one per device
\$HELP_FLAG [3]	Flags relating to help one per device
\$RELV_INDEX	Index of currently displayed relative view
\$RELV_URL[8]	Related view URLs which might be posted
\$RELV_MTEXT[8]	Text displayed in Related view menu
\$RELV_CONTEX [8]	Related view context identifier
\$RELV_FLAGS [8]	Related view flags

9.8 GENERIC LINKING DETAILED INFORMATION

All of the generic links are of the form:

href="/SOFTPART/GENLINK?parameter=value&etcetera"

There can be as many parameter=value pairs as are required to specify the link. In this case there are a fixed number of potential parameters with a fixed number of potential values. The order of the parameter value pairs is not relevant.

The parameter definitions are:

System variable	Description
PRIM	Put the URL in the parameter in the primary (left) pane.
DUAL	Put the URL in the parameter in the dual (right) pane.
TRIPLE	Put the URL in the parameter in the triple (lower right) pane.
CURRENT	Put the URL in the parameter in the pane, which currently has focus.
OTHER	Put the URL in the parameter in the primary (left) pane when linked from dual or triple; put it in the dual pane when linked from the primary pane.
HELP	Put the URL in the help pane and treat it as HELP. That is dismiss it with PREV or HELP.
FOCUS	Set the focus pane
CONFIG	Set the screen configuration
ADDRELV	Add a relative view link (primarily for testing)
EXECRELV	Execute a relative view link (primarily for testing)
DELRELV	Delete a relative view link (primarily for testing)
CTXHELP	Add a context help link (primarily for testing)

The parameters and values are:

- PRIM | DUAL | TRIPLE | CURRENT | OTHER =
MENUPAGE,spid,scid |
EDITPAGE,prog_name<,line_no> (current editor) |
WINTPE,1,prog_name<,line_no> (icon editor) |
MAINEDIT,prog_name<,line_no> (main editor) | IOPAGE,spid,scid,IN,port_no |
IOPAGE,spid,scid,OUT,port_no |
'ArbitraryURL' |
BROWSER (Forces the browser menu first. Typically used before one of the other parameters, such as "?prim=browser&prim=/fr/mypage.stm")
- HELP= 'ArbitraryURL'
- FOCUS= PRIM | DUAL | TRIPLE
- CONFIG= SINGLE |
STATSING (Status/Single) |

- DOUBLE |
- TRIPLE |
- WIDE (Single Wide) |
- DOUBLEH (Double Horizontal) |
- TRIPLEH (Triple Horizontal)
- ADDRELV= 'ArbitraryURL Menu Text'
ADDRELV= ' /SOFTPART/GENLINK?dual=menupage,930,1%20Program%20Adjust '
- EXECRELV= related_view_no
ADDRELV= ' /SOFTPART/GENLINK?dual=menupage,18,1%20Alarms' &EXECRELV=1
- DELRELV (deletes all related view links)
- CTXHELP= 'ArbitraryURL'
- TARGET= frame_name&TARGETURL='ArbitraryURL'
TARGET=user1&TARGETURL= ' /fr/user1.stm '

The operations are case insensitive. In the case of the MENUPAGE links, the softpart ID and screen ID of the target menu is also specified. In the case of the EDITPAGE, WINTPE, and MAINEDIT links, the name of the program and optionally the line number to launch in the editor is also supplied. In the case of the IOPAGE links, the softpart ID and screen ID of the I/O menu is specified along with IN or OUT and optionally the port_no.

9.9 RELATED VIEWS

Many related views are automatically provided by the system. In addition, you can set a related view via the link command ADDRELV and execute it via EXECRELV.

Related views is a concept which really only applies in two pane mode. This is a means to designate Web pages or other menus that are related to the current operation. These related menus are then selectable from the related view pop-up (*i* + FCTN). When related views are available, *i* is displayed on the focus bar.

For example you might want to look at a TPP program using some different display like a node map. In this case node map could be specified as the related view and the system would provide "node map" as a selection in the related view menu. This related view could be tied to the operation of the editor in the left pane. As the user cursors through the program and moves the robot the "related view" of the program as a gob of points is updated as well. Another example of related information is process data. The process parameters can be graphically displayed as a related view using this capability.

In some cases the related view is more specifically a related menu. For example editing a register line in a TPP program a related menu would be the register-editing screen where you could look at more detailed information about the register. It is also logical to provide access to the schedule editing of process parameters via this connection.

Almost all of the driving applications for related views are attached to the editor. This is generic functionality that could be applied to other menus as well. For example a related view of a system variable might be a chart of its values vs. time. The same sort of view of I/O point might also be utilized.

Related views are always related to the information in the left pane of the pendant. This is device_id 0 or the primary pane. This is where TPMM runs and the only pane that can run the editor.

The related views menu entries go away when the menu goes away or when the application decides it's no longer related to what the user is doing. For example, in the editor the register setup screen is NOT

related to a DIO instruction. So that entry is removed when the cursor is no longer on a register instruction.

This is similar to context sensitive help in the way in which it is set up. The user designates that a particular menu has particular related information and then that information is displayed as a choice in the related views menu. This information goes away when the menu goes away or when the application decides it's no longer related to what the user is doing.

A related view can be specified from a Web page or a KAREL program. From a Web page a related view can be set up via a link (see the detailed information and the examples).

9.10 EXAMPLES

Working examples are provided in /linking in the examples zip set. The files there can be copied to a memory card or FRVRC MC: directory and exercised relatively easily. You can also look at the files and determine the exact syntax for a wide range of available links.

Once you have copied all of these files to your memory card execute the command file links.cf from KCL to set up the system variables and other parameters. This will set up the [TYPE] menu on the browser screen and links in the browser favorites menu. You can exercise the links in these files and look at the HTML to get a good understanding of this functionality.

APPENDIX

A EXTENDED STATUS TEMPLATE

```

<html>
<head>
<title>Extended Status</title>
<script language="JavaScript"><!--
function NotifyTPTX() {
    // Notify TPTX softpart that extended status #EXTNUM is loaded in browser
    // TPTX must be loaded in SID_HTTPPEX
    // #define tpcf_cgt_p_ext_c (SSC_TP*0x10000 + 44)
    window.location.href =
        "/SOFTPART/tptx?fc=0x9002C&idx=#EXTNUM";
}
//--></script>
</head>
<body onload="NotifyTPTX()">
<font face="Arial">
    <table border="0" cellpadding="0" cellspacing="5" width="100%">
        <tr>
            <td width="2%">
                <!-- ***** -->
                <!-- Add your image here -->
                <!-- ***** -->
                
            </td>
            <td width="98%">
                <!-- ***** -->
                <!-- Add your title here -->
                <!-- ***** -->
                <a href="#EXTPAGE.stm"><font color=black>Title</font></a>
            </td>
        </tr>
        <tr>
            <td width="100%" bgcolor="#C6F9E3" colspan="2">
                <!-- ***** -->
                <!-- Add your code here -->
                <!-- To avoid scroll bars -->
                <!-- width = 200, height = 280 -->
                <!-- ***** -->
            </td>
        </tr>
    </table>
</font>
</body>
</html>

```

NOTE

#EXTNUM and #EXTPAGE are placeholders. They will be replaced with the correct values after the Extended Status pages are generated.

B CUSTOM SCREEN EXAMPLES

B.1 USING TABLES TO SET SIZE

The following example is also included with the *i*Pendant Controls setup. (wtest.stm). This file sets up a table with 2 columns. To add your information and controls, insert a second table into the cell which contains the “Insert a new...”, replacing this text.

(wtest.stm)

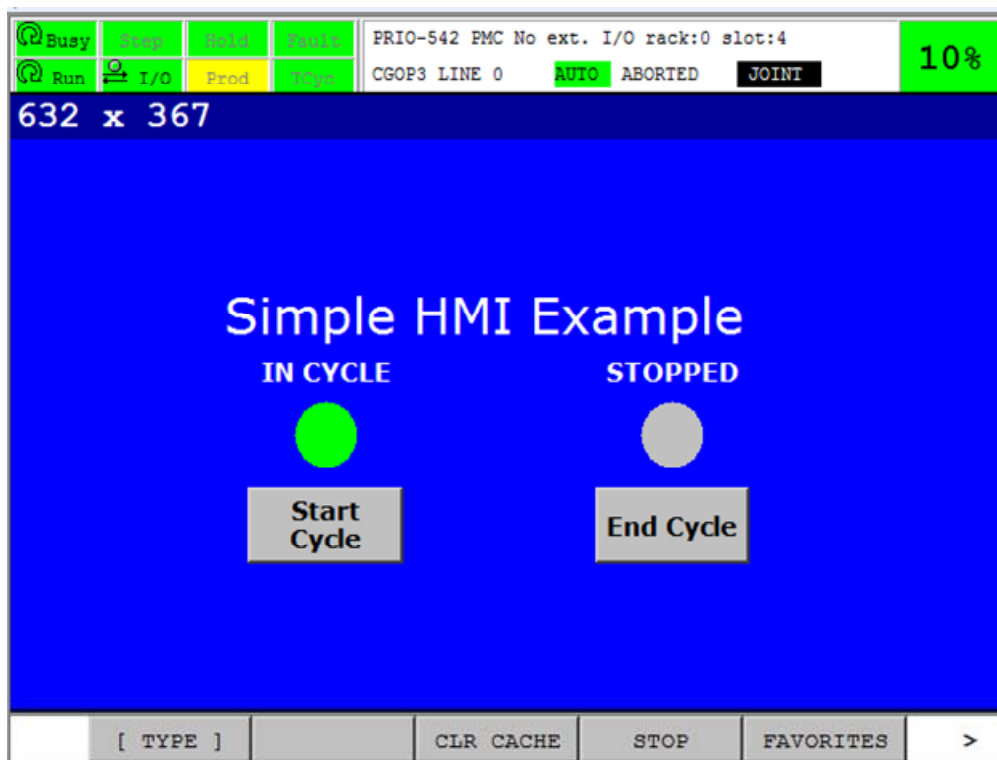
```
<html>
<head>
<title>632 x 367</title>
</head>
<body bgcolor="<!-- #echo var=_BGCOLOR -->" style="font-family: Verdana">
<div align="left">
  <table style="width: 632; height: 367" bgcolor="<!-- #echo var=_BGCOLOR -->" cellpadding="0" cellspacing="0">
    <tr>
      <td valign="middle" align="center">
        <p align="center"><font size="6">
          <object classid="clsid:7106065C-0E45-11D3-81B6-0000E206D650" id="FRIPLabel1">
            <param name="Caption" value="">
            <param name="FontSize" value="14">
            <param name="width" value="100">
            <param name="height" value="50">
            <param name="DataType" value="100">
            <param name="DataIndex" value="">
            <param name="Interval" value="250">
            <param name="TrueFont" value="-1">
            <param name="FastLoad" value="-1">
            <param name="Border" value="4">
          </object>
          Insert a new table in this</font></p>
          <p align="center"><font size="6">&nbsp;space for your controls</font></td>
        </tr>
      </table>
    </div>
  </body>
</html>
```


**NOTE**

The R-30iA *i*Pendant did not support style sheets so tables were used to position information and controls. The R-30iB *i*Pendant supports style sheets as defined by Internet Explorer 7 so positioning can be done using style sheets instead of tables.

B.2 SIMPLE HMI EXAMPLE

The following example is also included with the *i*Pendant Controls setup. (examp_1.stm). Refer to this file for the actual source code. This file uses the example above and creates a simple HMI screen by adding a second table within the main table and then inserting the *i*Pendant Controls and Text into this new table.



B.3 FORM EXAMPLE

B.3.1 Overview

The following example illustrates how to use standard HTML form components like buttons to interact with a KAREL program running on the robot to cause actions to occur. These can be anything that can be done from a KAREL program, including running programs, setting variables, registers and I/O. This is intended to be a simple example of what can be done. Many of these operations can be accomplished using the *iPendant* Controls without the need to have a KAREL program, however, the methods illustrated here can be used to perform complex functions that cannot be done with the *iPendant* Controls.

The basic operation of the example is: The FORM components are set up as SUBMIT buttons. The SUBMIT action sends the parameters to the web server (in this case the robot controller). One of the parameters passed from the HTML page is the KAREL program that is run to process the other parameters (in this case mpnlsvr). The Web Server starts the KAREL program and passes the parameters (via variables declared in the KAREL program). The KAREL program parses the parameters, performs the appropriate action, sends a status back to the Web Server, and EXITS. (For more information on using KAREL with the Web Server, see the Web Server Chapter in the R-30*iB* Internet Options Manual)

Both the HTML file (forms.stm) and the source and executable files for the KAREL Program (mpnlsvr.kl and mpnlsvr.pc) are included with the *iPendant* Controls setup for your reference.

B.3.2 Web Page

The following is the HTML page shown on the *iPendant*.

The screenshot shows the *iPendant* interface. At the top, there's a status bar with buttons: Busy (yellow), Stop (green), Hold (green), Fault (green), Run (yellow), I/O (green), Prod (green), and a green box showing 10%. To the right of these buttons, text reads: 'PRIO-542 PMC No ext. I/O rack:0 slot:4' and 'CGOP3 LINE 0 AUTO ABORTED JOINT'. Below the status bar is a blue header bar with the text 'Form Example' and 'FORM EXAMPLE' in large blue letters. The main area is light blue and contains four buttons: 'Reset Parts Counter (Numreg[1])', 'Run TP Program Part1', 'Auto Mode (DOUT[1])', and 'GenOverride=100'. At the bottom is a navigation bar with buttons: '[TYPE]', 'CLR CACHE', 'STOP', 'FAVORITES', and a right arrow '>'.

Below is the portion of the HTML code that defines the Reset Parts Counter Button.

```
<form action="/KAREL/mpnlsvr" method="GET">
  <div align="center">
```

```

        <input type="hidden" name="object" value="numreg">
        <input type="hidden" name="operate" value="setint">
        <input type="hidden" name="index" value="1">
        <input type="hidden" name="value" value="0">
        <input type="submit" value="Reset Parts Counter (Numreg[1])">
    </div>
</form>

```

The form “action” is defined to run the program `mpnlsvr` (which has been loaded on the controller). All of the parameters that are to be passed to the KAREL program are defined as “hidden” types. Note the “Name” of each of the hidden parameters is the variable in the KAREL program. Pushing this button on the *i*Pendant will cause `NUMREG[1]` to be set to 0.

B.3.3 KAREL Program

The following are key sections of the example KAREL program (`mpnlsvr.kl`). This program is run whenever a button on the HTML form is “pressed”. It shows how the different commands that can be input, are processed.

-- Example KAREL program

```
program mpnlsvr
```

```

.
.
.

```

```
var
```

-- Declare HTML parameter names and value

```

object  : string[12]
pname   : string[12]
operate : string[12]
index   : string[12]
value   : string[12]
URL     : string[128]
vname   : string[128]

```

-- These are duplicates that will be used to

-- convert the input parameters to Upper case

```

uobject : string[12]
uoperate: string[12]
uindex  : string[12]
uvalue  : string[12]
upname  : string[12]
uvname  : string[128]

```

-- Misc Variables

```

.
.
.

```

-- Convert input string to Uppercase for consistent comparison

```
routine toupper(p_char: integer): string
```

```
begin
  if (p_char > 96) and (p_char < 123) then
    p_char = p_char - 32
  endif
  return (chr(p_char))
end toupper

begin

-- Good practice to check for uninitialized variables before using them
.
.
.

-- Change all character of input parameters to uppercase for string comparison
.
.
.

-- Handle setting DOUTs

if (uobject = 'DOUT') then
  if (uoperate = 'SET') then
    cnv_str_int(uindex, index_i)
    if (uvalue = 'ON') then
      DOUT[index_i] = ON
    endif
    if (uvalue = 'OFF') then
      DOUT[index_i] = OFF
    endif
  endif
endif
.
.
.

-- Handle Setting Numreg values

if (uobject = 'NUMREG') then
  cnv_str_int(uindex, index_i)
  if (uoperate = 'SETINT') then
    cnv_str_int(uvalue, value_i)
    set_int_reg(index_i, value_i, status)
  endif

  if (uoperate = 'SETREAL') then
    cnv_str_real(uvalue, value_r)
    set_real_reg(index_i, value_r, status)
  endif
endif

-- Handle Running and Aborting a program

if (uobject = 'PROG') then
  if (uoperate = 'RUN') then
    kcommand = 'RUN '+ upname
    KCL_no_wait (kcommand, status)
```

```

    else
    kcommand = 'ABORT ' + upname
    KCL_no_wait (kcommand, status)
    endif
endif

-- Handle Setting a System Variable

if (uobject = 'SYSVAR') then
  if (uoperate = 'SETINT') then
    cnv_str_int(uvalue, value_i)
    set_var(entry, '*SYSTEM*', uvname, value_i, status)
  endif

  if (uoperate = 'SETREAL') then
    cnv_str_real(uvalue, value_r)
    set_var(entry, '*SYSTEM*', uvname, value_r, status)
  endif

  if (uoperate = 'SETSTR') then
    set_var(entry, '*SYSTEM*', uvname, uvalue, status)
  endif
endif

-- Return a NO RESPONSE Required code

return_code = 204

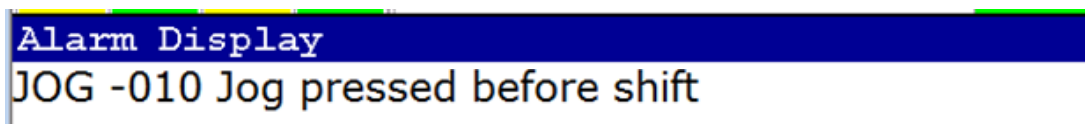
end mpnlsvr

```

B.4 AJAX EXAMPLE

B.4.1 Overview

The following example illustrates how to use Ajax to update the alarm on a custom web page. The web page displays the alarm and updates it every second.



B.4.2 Web Page

The following is the HTML page shown on the *iPendant*. You can include `ajax.js` which has ajax routines already written. You must create a global variable called `xmlhttp` and set it to null. You call `ajax_send_url` to send the URL to the web server and provide a callback routine. Use a sequence number so the URL is always unique and the browser will not cache it. In the example, the callback routine, `AlarmEvent`, gets executed when the response is returned from the URL. It calls `AlarmRefresh` again after 1 second.

```

alarm.htm
<html>
<head>
<script language=javascript src="/frh/cgtp/ajax.js"></script>

```

```

<script language=javascript>
var xmlhttp = null;

// Request alarm refresh
function AlarmRefresh() {
    ajax_send_url("/KAREL/GETALARM?&_seq=" + new Date().getTime(),
AlarmEvent);
}

// Execute alarm refresh
function AlarmEvent() {
    if (xmlhttp.readyState == 4) { // if readyState == complete
        // Execute JavaScript returned from GETALARM
        ajax_execute();
        window.setTimeout("AlarmRefresh()", 1000);
    }
}
</script>

<title>Alarm Display</title>

<style type="text/css">
.text {
    font-family: Verdana;
    font-size: large;
}
</style>

</head>
<body>
<div id="lblAlarm" class="text"></div>

<script language=javascript>
AlarmRefresh();
</script>
</body>
</html>

```

B.4.3 KAREL Program

The following is the KAREL program which is executed from URL. The KAREL program creates a response file which contains JavaScript which will be executed by `ajax_execute`. The JavaScript sets the `lblAlarm` <div> to the value of `$ALM_IF.$LAST_ALM`. This system variable can be used to get the text for the last alarm.

```

getalarm.kl
PROGRAM getalarm
%NOLOCKGROUP
%NOABORT=ERROR+COMMAND
%NOPAUSE=ERROR+COMMAND+TPENABLE
%NOBUSYLAMP

VAR
    file1      : FILE -- temp file for web browser
    respfile   : STRING[40] -- web server argument

BEGIN

```

```
-- Turn on $ALM_IF
$ALM_IF.$ENABLE = TRUE

IF UNINIT(respfile) THEN
  respfile = 'TD:RESPONSE.HTM'
ENDIF
OPEN FILE file1 ('RW', respfile)
-- Write JavaScript which will be executed as a response
WRITE file1('document.getElementById("lblAlarm").innerHTML = "')
WRITE file1($ALM_IF.$LAST_ALM)
WRITE file1('";', CR)
CLOSE FILE file1

END getalarm
```

B.4.4 Web Page using *i*Pendant Control Instead

The above example is used to show Ajax. However, if you just want the alarm displayed, using an *i*Pendant control is much simpler or more efficient. In this case, the KAREL program is not required, and only the following web page would be used. The controller monitors the system variable and only sends changes to the *i*Pendant Control when they occur.

```
alarm.stm
<html>
<head>
<title>Alarm Display</title>
</head>
<body>
<div>
  <object          classid="clsid:7106065C-0E45-11D3-81B6-0000E206D650"
id="FRIPLabell1" style="width: 100%">
    <param name="Caption" value="">
    <param name="FontSize" value="18">
    <param name="width" value="739">
    <param name="height" value="50">
    <param name="DataType" value="102">
    <param name="DataIndex" value="$ALM_IF.$LAST_ALM">
    <param name="Interval" value="250">
    <param name="TrueFont" value="-1">
    <param name="FastLoad" value="-1">
    <param name="Border" value="1">
    <param name="HAlign" value="0">
    <param name="VAlign" value="0">
  </object>
</div>
</body>
</html>
```

C USING DISCTRL_FORM TO DISPLAY A WEB PAGE

C.1 OVERVIEW

The KAREL built-in DISCTRL_FORM can be used to display and control a web page on the teach pendant screen. This is an existing built-in that is documented in the KAREL Reference Manual. It is normally used to display and control a dictionary form. When using it to display a web page, a dictionary form is not used and many of the parameters are not used.

Syntax :

DISCTRL_FORM(dict_name, ele_number, value_array, inact_array, change_array, term_mask, def_item, term_char, status)

Input/Output Parameters :

[in] dict_name : STRING
 [in] ele_number : INTEGER
 [in] value_array : ARRAY OF STRING
 [in] inact_array : ARRAY OF BOOLEAN
 [out] change_array : ARRAY OF BOOLEAN
 [in] term_mask : INTEGER
 [out] def_item : INTEGER
 [out] term_char : INTEGER
 [out] status : INTEGER

%ENVIRONMENT Group :PBcore

Details:

- dict_name is the URL of the web page.
- ele_number is not used.
- value_array is not used. It can be declared as an ARRAY[1] of STRING[1] and left uninitialized.
- inactive_array is not used. It can be declared as an ARRAY[1] of BOOLEAN and left uninitialized.
- change_array is an array of Booleans that corresponds to each control in the web page.
 - If the corresponding value is set, then the Boolean will be set to TRUE, otherwise it is set to FALSE. You do not need to initialize the array.
 - The array size can be greater than or less than the number of controls in the web page.
 - If change_array is not used, an array size of 1 can be used.
- term_mask is a bit-wise mask indicating conditions that will terminate the web page. This should be an OR of the constants defined in the include file klevkmsk.kl.
 - kc_func_key -- Function keys
 - kc_enter_key -- Enter and Return keys
 - kc_prev_key -- PREV key

If either a ButtonChange Control or a new menu is selected, the web page will always terminate, regardless of term_mask.
- def_item is not used on input. def_item returns the item that was currently highlighted when the termination character was pressed. def_item will be dynamically changed as the controls receive focus.

- term_char receives a code indicating the character or other condition that terminated the web page. The codes for key terminating conditions are defined in the include file klevkeys.kl. Keys normally returned are pre-defined constants as follows:
ky_undef -- No termination character was pressed
ky_select -- A ButtonChange Control was selected
ky_new_menu -- A new menu was selected
ky_f1 -- Function key 1 was selected
ky_f2 -- Function key 2 was selected
ky_f3 -- Function key 3 was selected
ky_f4 -- Function key 4 was selected
ky_f5 -- Function key 5 was selected
ky_f6 -- Function key 6 was selected
ky_f7 -- Function key 7 was selected
ky_f8 -- Function key 8 was selected
ky_f9 -- Function key 9 was selected
ky_f10 -- Function key 10 was selected
- DISCTRL_FORM will display the web page on the teach pendant device.
- status explains the status of the attempted operation. If status returns a value other than 0, an error has occurred.

NOTE

DISCTRL_FORM will only display the web page if the USER2 menu is the selected menu. Therefore, use
FORCE_SPMENU(device_stat, SPI_TPUSER2, 1)
before calling DISCTRL_FORM to force the USER2 menu.

C.2 WEB PAGE EXAMPLE 1

The following is a simple HTML page (excekex1.stm) shown on the *i*Pendant, which allows the selection of a menu item.

Form Example

KAREL FORM EXAMPLE 1

Menu Item 1

Menu Item 2

Menu Item 3

Menu Item 4

Menu Item 5

An Execution control is placed on the web page with the following properties. The Name must be webexec and the type must be SOFTPART. Otherwise DISCTRL_FORM will not operate correctly.

If Click Events is checked, then the ButtonChange objects will cause the form to exit with *term_char* = *ky_select* and *def_item* will equal the Object Tag selected. (i.e. If M2 is selected, *def_item* = 2.)

If Change Events is checked, then the objects that change a value will cause an element in *change_array* to be set TRUE. The element set will be the Object Tag that was changed. (i.e. If V5 is changed, *change_array*[5] is TRUE.)

If Focus Events is checked, then *def_item* will be updated with the currently focused object. (i.e. If M4 has focus, then *def_item* = 4.)

The screenshot shows the 'General' tab of a configuration window. The 'Name' field is set to 'webexec'. The 'Type' dropdown is set to '1 - SOFTPART'. Three checkboxes are checked: 'Click Events', 'Change Events', and 'Focus Events'.

In the example, there are 5 ButtonChange Controls for the menu items. The PageName is not used. The id's for the ButtonChange Controls are labeled "M1" to "M5". When computing the number from the Object Tag, the letters are ignored.

The screenshot shows the 'Indirect' tab of a configuration window. The 'Caption' field is set to 'Menu Item 1'. The 'ViewType' dropdown is set to '0 - Normal'. The 'PageName' field is empty, with a small button to its right.



C.3 KAREL PROGRAM EXAMPLE 1

The following is the KAREL program (execkex1.kl).

```
--
-- KAREL program to run web pages using DISCTRL_FORM
--
PROGRAM execkex1
%NOLOCKGROUP
%ENVIRONMENT UIF

%INCLUDE klevccdf

VAR
    device_stat: INTEGER
    value_array: ARRAY[1] OF STRING[1]
    inact_array: ARRAY[1] OF BOOLEAN
    change_array: ARRAY[1] OF BOOLEAN
    def_item: INTEGER
    def_item_sav: INTEGER
    term_char: INTEGER
    status: INTEGER

ROUTINE notify_user
    BEGIN
        def_item_sav = def_item
        WRITE TERROR (CHR(cc_clear_win))
        WRITE TERROR ('Focus on ', def_item::1)
    END notify_user

BEGIN
    condition[1]:
    when def_item <> def_item_sav do
        notify_user
        enable condition[1]
    endcondition

--
--      /MC/EXECKEX1.STM:  contains selectable menu items
--
FORCE_SPMENU(device_stat, SPI_TPUSER2, 1)
def_item = 1  -- start with menu item 1
def_item_sav = 1
enable condition[1]
DISCTRL_FORM(
    '/MC/EXECKEX1.STM', -- dict_name contains web page
    0,                  -- ele_number not used
    value_array,        -- not used
    inact_array,        -- not used
```

```

change_array,      -- not used in this web page
0,                 -- term_mask not used in this web page
def_item, term_char, status)
disable condition[1]
WRITE TPERROR (CHR(cc_clear_win))
IF status <> 0 THEN
  WRITE TPERROR ('DISCTRL_FORM status ', status)
  ABORT
ELSE
  IF term_char = ky_select THEN
    WRITE(CHR(cc_clear_win))
    FORCE_SPMENU(device_stat, SPI_TPUSER, 1)
    WRITE ('Menu item ', def_item::1, ' was selected.')
  ENDIF
ENDIF
ENDIF

END execkex1

```

C.4 WEB PAGE EXAMPLE 2

The following is an HTML page (execkex2.stm) shown on the *i*Pendant, which allows the editing of different types of values.

The screenshot shows a web page titled "Form Example" with a blue header. The main content area is light blue and contains the title "KAREL FORM EXAMPLE 2". Below the title are several input fields for different data types: Integer (12345), Integer (*****), Real (12.340000), Boolean (TRUE), String (This is a test), and String (*****). At the bottom, there is a Byte field with the value 255. The top status bar shows "Focus on 1" and "10%". The bottom navigation bar has buttons for "F2", "F3", "F4", "EXIT", and a right arrow.

An Execution control is placed on the web page with Name = webexec, Type = SOFTPART, Check Events, Change Events, and Focus Events are checked. Each EditText Control has a name starting with V1 and ending with V18.

C.5 KAREL PROGRAM EXAMPLE 2

The following is the KAREL program (execkex2.kl).

```
--
-- KAREL program to run web pages using DISCTRL_FORM
--
PROGRAM execkex2
%NOLOCKGROUP
%ENVIRONMENT UIF

%INCLUDE klevccdf

TYPE
    mystruc = STRUCTURE
        byte_var1: BYTE
        byte_var2: BYTE
        short_var: SHORT
    ENDSTRUCTURE

VAR
    device_stat: INTEGER
    value_array:  ARRAY[1] OF STRING[1]
    vptr_array:   ARRAY[1] OF INTEGER
    inact_array:  ARRAY[1] OF BOOLEAN
    change_array: ARRAY[20] OF BOOLEAN
    def_item: INTEGER
    def_item_sav: INTEGER
    term_char: INTEGER
    status: INTEGER
    idx: INTEGER
    done: BOOLEAN

    int_var1: INTEGER
    int_var2: INTEGER
    real_var: REAL
    bool_var: BOOLEAN
    str_var1: STRING[20]
    str_var2: STRING[12]
    struc_var: mystruc
    term_text: STRING[12]
    color_sel1: INTEGER
    color_sel2: INTEGER
    color_text1: STRING[12]
    color_text2: STRING[12]
    prog_name1: STRING[12]
    prog_name2: STRING[12]
    prog_name3: STRING[12]
    prog_name4: STRING[12]
    choices: ARRAY[100] OF STRING[12]

ROUTINE notify_user
BEGIN
    def_item_sav = def_item
```

```

WRITE TPERERROR (CHR(cc_clear_win))
WRITE TPERERROR ('Focus on ', def_item::1)
END notify_user

BEGIN
condition[1]:
when def_item <> def_item_sav do
    notify_user
    enable condition[1]
endcondition

--
--      /MC/EXECKEX2.STM:  contains edit data items
--
FORCE_SPMENU(device_stat, SPI_TPUSER2, 1)

-- The combo boxes uses WEBF dictionary
-- Make sure 'WEBF' dictionary is added.
CHECK_DICT('WEBF', 1, status)
IF status <> 0 THEN
    ADD_DICT('WEBF', 'WEBF', dp_default, dp_cmos, status)
ENDIF

-- Initialize variables
int_var1 = 12345
str_var1 = 'This is a test'
choices[1] = "          -- not used
choices[2] = 'Red'      -- corresponds to color_sel2 = 1
choices[3] = 'Blue'     -- corresponds to color_sel2 = 2
choices[4] = 'Green'    -- corresponds to color_sel2 = 3
choices[5] = 'Yellow'   -- corresponds to color_sel2 = 4
FOR idx = 6 TO 100 DO
    CNV_INT_STR(idx - 1, 0, 0, choices[idx])
ENDFOR

done = FALSE
WHILE NOT done DO
    def_item = 1  -- start with menu item 1
    def_item_sav = 1
    enable condition[1]
    DISCTRL_FORM(
        '/MC/EXECKEX2.STM', -- dict_name contains web page
        0,                  -- ele_number not used
        value_array,        -- not used
        inact_array,        -- not used
        change_array,       -- change_array
        0,                  -- term_mask not used in this web page
        def_item, term_char, status)
    disable condition[1]
    done = TRUE
    WRITE TPERERROR (CHR(cc_clear_win))
    IF status <> 0 THEN
        WRITE TPERERROR ('DISCTRL_FORM status ', status)
    ELSE
        IF term_char = ky_select THEN

```

```
        WRITE TPEROR ('Function key ', def_item::1, ' was selected.')
        IF def_item <> 5 THEN -- F5 is EXIT
            done = FALSE
        ENDIF
    ENDIF
ENDIF
ENDWHILE

-- Notify user which values were changed
IF term_char <> ky_new_menu THEN
    WRITE(CHR(cc_clear_win))
    FORCE_SPMENU(device_stat, SPI_TPUSER, 1)
    done = FALSE
    FOR idx = 1 to 20 DO
        IF change_array[idx] = TRUE THEN
            WRITE('Value ', idx, ' changed', CR)
            done = TRUE
        ENDIF
    ENDFOR
    IF done = FALSE THEN
        WRITE('No values were changed', CR)
    ENDIF
ENDIF
END execkex2
```

D

SYSTEM VARIABLES

From a KAREL program, it may be useful to know the web page that is being displayed in the *i*Pendant browser. The web page is in the form of a URL.

There are a maximum of 5 *i*Pendant connections and a maximum of 9 tasks. Each task controls a separate pane. The system variable, \$UI_PANEDATA[PANEID] will store for each pane; the current web page, the container frame, and the pane state.

[*SYSTEM*] \$UI_PANEDATA Storage: DRAM Access: RW : ARRAY[9] OF UI_PANEDAT_T

\$UI_PANEDATA[1] will always contain the primary *i*Pendant pane information.

\$UI_PANEDATA[2] will always contain the dual *i*Pendant pane information.

\$UI_PANEDATA[3] will always contain the third *i*Pendant pane information.

\$UI_PANEDATA[4-9] will contain pane information for Internet Explorer connections. \$UI_CONFIG.\$PANEMAP can be used to find the PANEID for a specific connection.

[*SYSTEM*] \$UI_CONFIG.\$PANEMAP Storage: CMOS Access: RW : ARRAY[16] OF SHORT

\$UI_PANEDATA is stored in DRAM so the values are not retained between power up. The values do not need to be saved.

[*SYSTEM*] \$UI_PANEDATA[1] Storage: DRAM Access: RW : UI_PANEDAT_T
 Field: \$UI_PANEDATA[1].\$PAGEURL Access: RW : STRING[125] = '/mc/fmt.stm'
 Field: \$UI_PANEDATA[1].\$FRAME Access: RW : STRING[41] = 'prim'
 Field: \$UI_PANEDATA[1].\$PANESTATE Access: RW : BYTE = 2
 Field: \$UI_PANEDATA[1].\$HELPURL Access: RW : STRING[125] = Uninitialized
 Field: \$UI_PANEDATA[1].\$PARAMETER1 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER2 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER3 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER4 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER5 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER6 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER7 Access: RW: STRING[41] = "
 Field: \$UI_PANEDATA[1].\$PARAMETER8 Access: RW: STRING[41] = "

\$PAGEURL will store the web page URL.

\$FRAME will store the frame name. This could be a user frame.

A user frame may be put into a pane and this may contain many web pages. Only the last requested URL and frame will be stored.

\$PANESTATE will store the document state:

UI_DISCONNECT = 0 /* connection has been disconnected */

UI_CONNECTED = 1 /* document has been requested */

UI_LOADED = 2 /* document is loaded */

\$HELPURL will be set by the Help Control. If \$HELPURL is set, then the Help system will use this URL instead of the default help file.

\$PARAMETER1 - \$PARAMETER8 will be set by the Execution Control and can be used as parameters to your KAREL program.

E KAREL BUILTINS

E.1 FORCE_LINK BUILT-IN

From a KAREL program, it may be useful to display a web page in the BROWSER menu. Use FORCE_LINK built-in.

FORCE_LINK(pane_id: integer; url: string)

pane_id	Description
tp_panel = 1	Primary iPendant pane
tp2_panel = 2	Dual iPendant pane
tp3_panel = 3	Third iPendant pane
4 – 9	Panes for Internet Explorer connections

url can be any valid URL location.

Some examples:

Force fr:¥pw_op1.stm into primary pane of iPendant:

```
FORCE_LINK(tp_panel, '/fr/pw_op1.stm')
```

Force fr:¥pw_op1.stm into the currently focused frame of iPendant:

```
FORCE_LINK(tp_panel, 'current=/fr/pw_op1.stm')
```

Force other robot home pages into second and third pane of iPendant:

```
FORCE_LINK(tp2_panel, 'http://robot05')
FORCE_LINK(tp3_panel, 'http://robot08')
```

Force the page in the appropriate pane to be refreshed:

```
FORCE_LINK(tp_panel, 'refresh=prim')
FORCE_LINK(tp_panel, 'refresh=dual')
FORCE_LINK(tp_panel, 'refresh=third')
```

E.2 GET_DEV_INFO BUILT-IN

From a KAREL program running as a UIF task, it may be useful to know what the display device is and what it is capable of. Use the GET_DEV_INFO built-in to get this information.

GET_DEV_INFO(device_type, dev_id, pane_id, connect_id, parent, status)

All parameters are integers.

device_type returns the device type.

device_type	Description
no_uif_dev = 0	No Teach Pendant is connected
tp_uif_dev = 1	Legacy Teach Pendant is connected

device_type	Description
ip_uif_dev = 2	iPendant is connected
ie_uif_dev = 3	KAREL program was run from Internet Explorer connection
crt_uif_dev= 254	KAREL program was run from CRT Menus

dev_id returns the Device ID.

dev_id	Description
1	Primary (left window)
2	Dual (right window)
3	Third (lower right window)

pane_id returns the Pane ID.

pane_id	Description
tp_panel = 1	Primary iPendant pane
tp2_panel = 2	Dual iPendant pane
tp3_panel = 3	Third iPendant pane
4 – 9	Panes for Internet Explorer connections

Connect_id returns the connection index.

connect_id	Description
tp_panel = 1	Teach Pendant connection
crt_panel = 254	CRT connection
2-5	Internet Explorer connections

parent returns the parent 'C' UIF task id. Internal use only.

Status returns:

SUCCESS

-1 for invalid

-2 inactive connection

Some examples:

Use device_type to determine if this KAREL program was run from an Internet Explorer connection:

```
GET_DEV_INFO(device_type, dummy, dummy, dummy, dummy, status)
IF (status = 0) AND (device_type = ie_uif_dev) THEN
  -- this is an Internet Explorer connection
ENDIF
```

Use pane_id as a parameter to FORCE_LINK.

```
GET_DEV_INFO(dummy, dummy, pane_id, dummy, dummy, status)
IF (status = 0) THEN
  FORCE_LINK(pane_id, '/fr/pw_op1.stm')
ENDIF
```

E.3 DISPLAY WEB PAGE MACRO

A Teach Pendant macro called DSP_WEBP.MR is included with the Menu Utility option. It can be called from the Teach Pendant programs to display any web pages created by users for the iPendant.

User can call this macro with a parameter in the TP programs.

Usage:

```
CALL DSP_WEBP(1)      -- Displays the Favorites menu in the BROWSER
CALL DSP_WEBP(2)      -- Displays the first user created web page in the BROWSER [TYPE]
pull-up ...
CALL DSP_WEBP(11)    -- Displays the tenth user created web page in the [TYPE] pull-up
```

Parameters 2-11 are for displaying user-created web pages (created with or without the Panel Wizard).

The menu will be displayed only if the teach pendant is *iPendant*. It will always be displayed on the active pane on the *iPendant*.

E.4 DISCTRL_DIAG

The KAREL built-in DISCTRL_DIAG can be used to display and control a dialog box on the *iPendant* screen.

Syntax :

```
DISCTRL_DIAG(file_spec, term_mask, term_char, status)
```

Input/Output Parameters:

```
[in] file_spec : STRING
[in] term_mask : INTEGER
[out] term_char : INTEGER
[out] status : INTEGER
```

%ENVIRONMENT Group :UIF

Details:

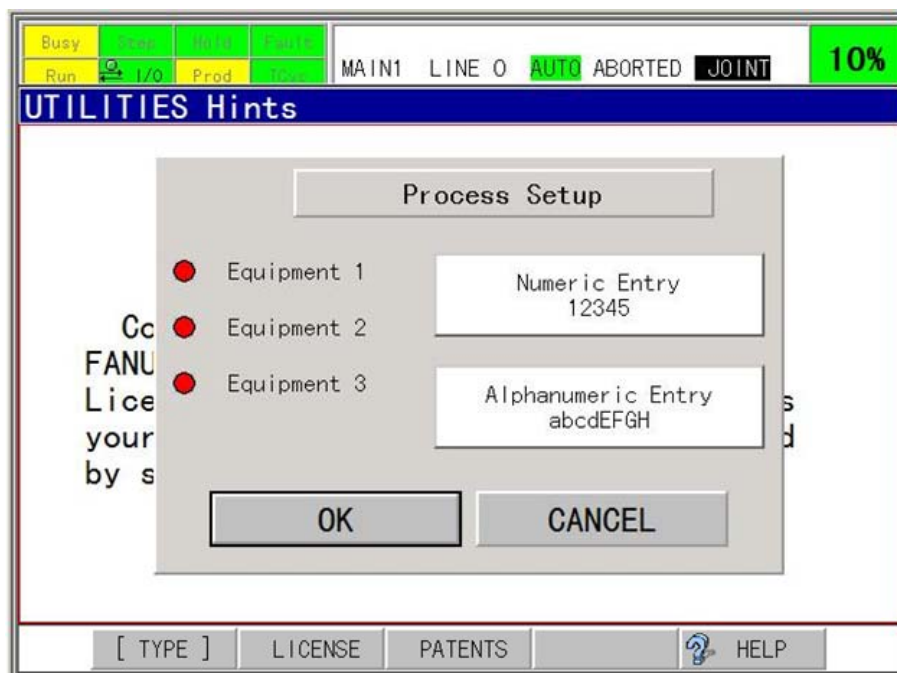
- file_spec specifies the device and filename of the XML file defining the dialog box. If no device name is specified, the default device is used.
- term_mask is a bit-wise mask indicating conditions that will terminate the dialog box. This should be an OR of the constants defined in the include file `klevkmsk.kl`.
 kc_prev_key -- PREV key
 kc_enter_key -- Enter key
 kc_func_key -- Function keys
 0 is also a valid term_mask if one or more of the controls on the dialog box are used to dismiss it.
- term_char receives a code indicating the character or other condition that terminated the dialog box. The codes for key terminating conditions are defined in the include file `klevkeys.kl`. Keys normally returned are pre-defined constants as follows:
 ky_cancel -- Dialog box was dismissed
 ky_prev -- Prev key was pressed (only if kc_prev_key mask is used)
 ky_enter -- Enter key was pressed (only if kc_enter_key mask is used)
 ky_new_menu -- A new menu was selected
- DISCTRL_DIAG will display the dialog box on the teach pendant device over top of the menu being displayed.
- status explains the status of the attempted operation. If status returns a value other than 0, an error has occurred.
- If the KAREL program is aborted, the dialog box is automatically dismissed.

A Dialog Box can contain a single item like a string entry or it can be a collection of several items such as string entry, radio buttons and check boxes. The following are some of the items that may be included in a Dialog Box.

- Discrete buttons (Yes, No, OK, Cancel etc.)
- Labels
- String entry
- Numeric entry
- Check boxes
- Choice boxes (popup lists)
- Radio buttons
- Images
- Charts

The Dialog Box uses standard *i*Pendant Controls on an independent window to provide these items.

The size and location of the Dialog Box are configurable and it can be displayed across multiple panes.



E.4.1 Dialog Box XML File

To make a Dialog Box, you must first create an XML file to define the box and then store that file in controller memory (FR:, MC:, etc.) where it can be accessed by the KAREL built-in. The XML file is an ASCII file with .xml extension. Later, to display the box, you call the KAREL built-in DISCTRL_DIAG and provide it the name of the XML file.

The XML file defines the following items:

- General layout of the Dialog Box and position of all items
- Definition of the items on the Dialog Box (checkbox, radio button etc.)
- Standard *i*Pendant Control information including:
 - Size and color
 - Label text
 - Data type
 - Data index

Tags and attributes are used to define these items.

The typical layout of a dialog box XML file is:

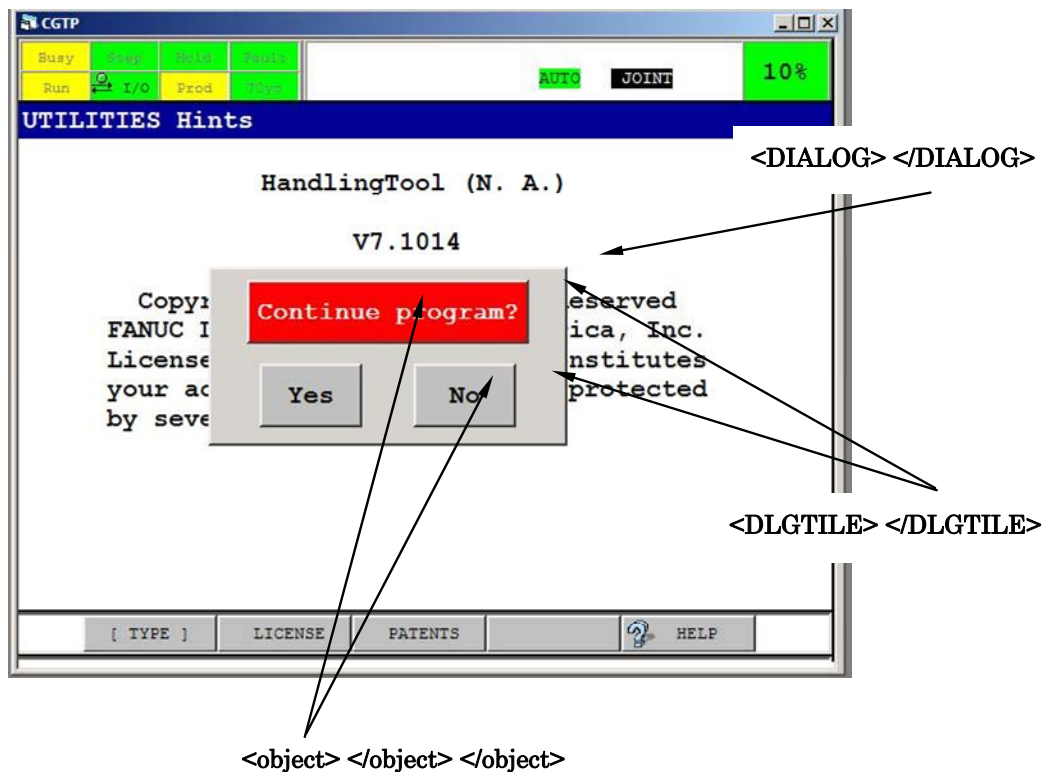
<DIALOG [DIALOG attributes] >	- One of these required in the file
<TEXT [TEXT attributes]>title</TEXT>	- An optional title
<DLGTILE [DLGTILE attributes] >	- One for each control
<object [object attributes] >	- One per DLGTILE
<object parameter>	- <i>i</i> Pendant Control parameters
.	
.	
.	
<object parameter>	
</object>	- End of object
</DLGTILE>	- End of tile
<DLGTILE [DLGTILE attributes] >	- Next tile in dialog box
<object [object attributes] >	
<object parameter>	
.	
.	
.	
<object parameter>	
</object>	
</DLGTILE>	
</DIALOG>	- End of Dialog Box

There can only be one Dialog Box definition per XML file.

E.4.1.1 Tags and Attributes

Several XML tags are defined to describe a dialog box and the *i*Pendant Controls that make it up.

- **<DIALOG> </DIALOG>** - The overall dialog box
- **<TEXT> </TEXT>** - An optional dialog box title
- **<DLGTILE> </DLGTILE>** - The individual tiles that contain each *i*Pendant Control
- **<object> </object>** - The actual *i*Pendant Controls that make up the dialog box



<DIALOG> </DIALOG>

This tag defines the overall dialog box window including the size and its location on the screen. The location is based on the whole screen and not an individual pane.

```
<DIALOG posx="col#" posy="row#" width="#cols" height="#rows" bgcolor="color#"
border="type#" bordercolor="color#">
```

```
.....
</DIALOG>
```

posx="col#" Specifies the column position of the dialog box.
posy="row#" Specifies the row position of the dialog box.
width="#cols" Specifies the width of the dialog box in pixels.
height="#rows" Specifies the height of the dialog box in pixels.
bgcolor="color#" Specifies the background color in RGB values.
border="type#" Specifies the border type where:
0 - Thin3D Create a thin 3D line.
1 - None No border line.
2 - Black Create a thin black line.
3 - BorderColor Create a thin line whose color is the equal to the bordercolor.
4 - Bold3D Create a bold 3D line.
bordercolor="color#" Specifies the border color in RGB values.

<TEXT> </TEXT>

This is an optional tag that defines a title for the dialog box. The location is based on the overall dialog box window. Only one TEXT tag is supported.

```
<TEXT clr="color#" font="font#" align="horiz,vert" loc="x,y" dims="#cols,#rows">
.....
</TEXT>
```

clr="color#" Specifies the foreground color in RGB values.
font="font#" Specifies the font size.
fontname="font" Specifies the font name.
align="horiz,vert" Specifies the horizontal (left,center,right) and vertical (top,middle,bottom) alignment within the text area.
loc="x,y" Specifies the column and row within the dialog box.
dims="#cols,#rows" Specifies the width and height in pixels of the text area.

When **fontname** is specified, then **TrueFont** is automatically used. Otherwise **TrueFont** is false and the fonts will be compatible with R-30iA *iPendant*. See additional information in the **Common Controls Properties, Fonts** section.

<DLGTILE> </DLGTILE>

This tag defines the size and location of individual tiles on the dialog box that will contain the *iPendant* Controls. Their location is based on the overall dialog box window. This tag also contains a special attribute that defines whether selecting this control will dismiss the dialog box. There must be at least one <DLGTILE> tag with this attribute set to 1 when calling DISCTRL_DIAG with term_mask = 0. If one of more of the *iPendant* Controls need to use enter or prev to operate than term_mask should be 0.

<DLGTILE posx="col#" posy="row#" width="#cols" height="#rows" type="dismiss">

.....

</DLGTILE>

posx="col#" Specifies the column position of the entity within the dialog box.
posy="row#" Specifies the row position of the entity within the dialog box.
width="#cols" Specifies the width of the entity in pixels.
height="#rows" Specifies the height of the entity in pixels.
type="dismiss" Specifies whether the dialog box will be dismissed when this entity is selected. 1 = dismiss, 0 or not specified = not dismissed

<object> </object>

This tag is the normal tag for an *iPendant* Control. It defines one control in each tile on the dialog box. The format of the object tag is the standard format for *iPendant* Controls. The information can be copied verbatim from the HTML view of a web page editor.

This is an example of a typical object tag:

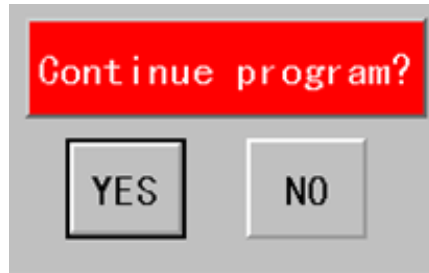
```
<object classid="clsid:7106065C-0E45-11D3-81B6-0000E206D650"
      id="FRIPLabel1" width="300" height="35">
  <param name="_Version" value="65536">
  <param name="_ExtentX" value="2646">
  <param name="_ExtentY" value="1323">
  <param name="_StockProps" value="6">
  <param name="Caption" value="Dialog Box">
  <param name="FontName" value=" Courier New ">
  <param name="FontSize" value="16">
  <param name="DataType" value="100">
  <param name="DataIndex" value>
  <param name="Interval" value="250">
  <param name="FastLoad" value="1">
  <param name="BackColor" value="13554646">
</object>
```

Note the parameters that start with `_`, such as `“_Version”`, are not required and can be omitted. In addition, the width and height on the object tag is not used. The width and height on the DLGTILE is used instead.

E.4.1.2 XML content example

To create the Dialog Box XML file, you first create the *iPendant Controls* using a web page editor such as SharePoint Designer. Then you copy and paste the `<object> .. </object>` sections into the Dialog Box XML as shown below.

Below is an example of an XML file that defines the following dialog box.



This box contains three *iPendant Controls*, a Label (“Continue program?”) and two Toggle Buttons (“YES” and “NO”). When a Toggle Button is pressed, the KAREL variable is set and the dialog box is dismissed.

dlgyesno.xml

```
<DIALOG posx="220" posy="170" width="220" height="140" bgcolor="#C0C0C0">

<!--Continue program? -->
<DLGTILE posx="10" posy="10" width="200" height="50">
<object classid="clsid:7106065C-0E45-11D3-81B6-0000E206D650" id="LABEL1">
  <param name="Caption" value="Continue program?">
  <param name="DataType" value="100">
  <param name="FontSize" value="16">
  <param name="ForeColor" value="16777215">
  <param name="BackColor" value="255">
</object>
</DLGTILE>

<!-- Yes -->
<DLGTILE posx="30" posy="70" width="60" height="50" type="1">
<object classid="clsid:7106066C-0E45-11D3-81B6-0000E206D650" id="TGBTN1">
  <param name="Caption" value="YES">
  <param name="DataType" value="103">
  <param name="DataIndex" value="[runyesno]yes">
  <param name="FontSize" value="16">
  <param name="TrueColor" value="12632256">
  <param name="FalseColor" value="12632256">
  <param name="BackColor" value="16777215">
</object>
</DLGTILE>

<!-- No -->
<DLGTILE posx="120" posy="70" width="60" height="50" type="1">
<object classid="clsid:7106066C-0E45-11D3-81B6-0000E206D650" id="TGBTN2">
  <param name="Caption" value="NO">
```



```

    <param name="DataType" value="103">
    <param name="DataIndex" value="[runyesno]no">
    <param name="FontSize" value="16">
    <param name="TrueColor" value="12632256">
    <param name="FalseColor" value="12632256">
    <param name="BackColor" value="16777215">
  </object>
</DLGTILE>
</DIALOG>

```

E.4.1.3 KAREL program example

```

runyesno.kl
--
-- KAREL program to display dialog box
--
PROGRAM runyesno

%NOLOCKGROUP
%ENVIRONMENT uif

%INCLUDE klevkmsk
%INCLUDE klevccdf

VAR
  term_char: INTEGER
  status: INTEGER
  yes: BOOLEAN
  no: BOOLEAN

BEGIN
  yes = FALSE
  no = FALSE
  DISCTRL_DIAG('MC:¥DLGYESNO.XML', kc_prev_key, term_char, status)
  IF term_char = ky_prev THEN
    no = TRUE
  ENDIF
  IF yes = TRUE THEN
    WRITE TERROR (CHR(cc_home) + CHR(cc_clear_win) + 'Continue program')
  ELSE
    WRITE TERROR (CHR(cc_home) + CHR(cc_clear_win) + 'Do not continue program')
  ENDIF
END runyesno

```


INDEX

<\$>

\$ALM_IF	125
\$TX	125
\$UI_CONFIG.....	126
\$UI_MENHIST[5]	126
\$UI_PANEDATA[9]	126
\$UI_PANELINK[5]	126
\$UI_USERVIEW	125

<A>

ABOUT IDENTIFIERS	124
ActiveX Control Shortcut.....	7
Adding script.....	18
Ajax.....	16
AJAX EXAMPLE.....	139
Alignment.....	27,101
Annotation & AnnotateFontSize	68
AutoChange Control	52

BACKUP AND RESTORE OF BROWSER FILES	4
Bar Chart Control	79
Border	26,68,89,99
BROWSER MENUS.....	2
BUTTON tag.....	113
ButtonChange Control	55

<C>

Caption.....	28,87,98
Caption & CaptionFontSize	68
Caution and Limitations.....	40
Ch_Data_N.....	78
Channel config tab	74
Chart config tab.....	69
CHART CONTROL DESCRIPTION.....	79
ChartClear	79
CHARTING CONTROL DESIGN ADVICE	83
ChartType	70
CHECK CONDITION	34
ChN_AutoRange.....	77
ChN_Clear	79
ChN_Color	75
ChN_Data.....	76
ChN_DataGrid	77
ChN_DataMarkerN	78
ChN_DataScale.....	77
ChN_Digital	76
ChN_Name.....	75
ChN_Rate.....	77
ChN_Source	75
ChN_State	76
Circ.....	91
CIRCLE tag.....	111
Clear	93

ClickMe	34
Colors.....	26
ComboBox Control	56
ComboBox Control Limitation	41
CommandButton Control	47
COMMANDING LINKS FROM KAREL	123
COMMON CHART CONTROL PROPERTIES	66
COMMON CONTROL PROPERTIES	24
COMMON DRAWING CONTROL PROPERTIES	86
COMMON GRID CONTROL PROPERTIES.....	96
CONTEXT SENSITIVE HELP	123
CONTROL ARRANGEMENT	23,65,85
CONTROL DESCRIPTION	41
CONTROL DESIGN ADVICE	60
CONTROL FEATURES SUMMARY	7
Conventions	99
CUSTOM SCREEN EXAMPLES	134

<D>

Data.....	90
DataFontSize.....	71
DataFormat	71
DataGrid.....	74
DataScale	73
DataScaleFormat	73,74
DataShowValues	72
DataType and DataIndex	24
Debugging in internet explorer	19
Debugging in source view.....	19
Delete	93
DEL TILE tag	118
Dialog Box XML File	154
Diam.....	92
DISCTRL_DIAG	153
DISPLAY CONCEPTS	99
DISPLAY MENU FUNCTIONS	122
Display other than iPendant	38
DISPLAY tag.....	115
DISPLAY WEB PAGE MACRO	152
Display, Pan and Zoom.....	101
DOM Elements	20
Dynamic data	120
Dynamic Display	102
Dynamic Entity modifications	93

<E>

EditBox Control	43
Error Code Dialog.....	60
Error Code Messages	60
Error Handling	83
Error Message 1	36
Error Message 2	37
EXAMPLES	129
Execution Control	59
Execution Control Limitation	41

EXTENDED STATUS TEMPLATE.....	133	Layer control.....	89
EXTENDED STATUS WINDOW	3	LayerOff.....	93
<F>		LayerOn	93
FastLoad.....	69,89,99	Line	91
File Names	10	Line Chart Control	81
Font Name.....	14	LINE tag	109
Font Size	13	LineScaleActive	71
Fonts.....	26,66,86,96	Links	14
FORCE_LINK BUILT-IN	151	<M>	
Foreground & Background Colors	68,88,98	MAKING A CUSTOM iPendant SCREEN USING	
FORM EXAMPLE.....	136	THE iPendant CONTROLS	23
Forms	15	MenuChange Control.....	54
Frames.....	16	Meta Tags	11
Function Key ViewType	27	Miscellaneous	78
<G>		Miscellaneous Properties	93
General 2 tab	89	Modifying SHAPES.....	119
General Chart Properties	67	Modifying TILES.....	119
General tab	87,97	Monitor	27
GENERIC LINKING DETAILED INFORMATION..	127	Multi Control	51
GENERIC LINKING FUNCTIONALITY	123	<N>	
GET_DEV_INFO BUILT-IN	151	Name	68,88,98
GRID tag	105	NEW/MODIFIED SYSTEM VARIABLES	125
GridColor	74	<O>	
GridType	74	Object Tag.....	24,66,86,96
<H>		Orientation	70
Help Control.....	56	OVERVIEW	2
Help Control Limitation.....	41	Overview.....	136,139
HELP FOR BROWSER MENUS	3	OVERVIEW	142
HTML Editing.....	13	<P>	
<I>		Page Properties.....	11
Imag	92	PANE LINKING.....	122
IMAGE tag.....	112	PANEID	33
Images	14,25	Path	91
Indirect DataType and DataIndex	32	Pipe	33,69,88,98
INSERTING A GRID CONTROL ON A WEB PAGE.	95	Pipe Mon Rate.....	99
INSTALLATION.....	6	PipeMonRate.....	69,89
INTRODUCTION.....	1,122	POLYGON tag.....	110
InvertY	90	Port Simulation	29
iPendant CONTROLS SUMMARY	5	Positioning	13
iPendant WEB BROWSER.....	2	Positions.....	30
<J>		PROPERTIES WITH ADDITIONAL OPTIONAL	
jQuery	17	VALUES	83
<K>		PUBLISHING YOUR WEB.....	21
KAREL BUILTINS	151	Pulse DO	29
KAREL Program.....	137,140	<R>	
KAREL program example	159	RECOMMENDED ENVIRONMENT	5
KAREL PROGRAM EXAMPLE 1	145	Rect	92
KAREL PROGRAM EXAMPLE 2	147	RECTANGLE tag.....	109
<L>		RELATED VIEWS	128
Label Control	41	Rendering.....	103
LabelFontSize	71	<S>	
		SAFETY PRECAUTIONS	s-1

SampleGrid	73
SampleMarkerColor	78
SampleMarkerN	78
SampleScale	72
SampleScaleAspect	72
SampleScaleFormat	73
Scale	90
Scripting Elements	17
Scripting iPendant controls	19
Scroll bars with the DISPLAY tag	116
SetFocus	33
SharePoint Designer 2007	9
SIMPLE HMI EXAMPLE	135
Supported Controls	35
SYSTEM VARIABLES	150

<T>

Tags and Attributes	155
Text	90
TEXT tag	107
The Entities	90
Themes and Styles	16
TILE tag	106
ToggleButton Control	48
ToggleLamp Control	44

<U>

User Interface	35
USING DISCTRL_FORM TO DISPLAY A WEB PAGE	142
USING FANUC iPendant CONTROLS	5
USING TABLES TO SET SIZE	134
USING THE CHART CONTROL	63
Using the Data parameter	120
USING THE DRAWING CONTROL	84
USING THE GRID CONTROL	94
Using the Name Parameter	120
Using the Pipe parameter	120

<W>

Web Page	136,139
WEB PAGE EXAMPLE 1	143
WEB PAGE EXAMPLE 2	146
Web Page using iPendant Control Instead	141
Window Size	11
WORKING WITH PAGES	9
WORKING WITH WEBS	9

<X>

XML content example	158
XML tag	105
XML TO THE GRID	103

REVISION RECORD

Edition	Date	Contents
01	Aug., 2013	

B-83594EN/01



* B - 8 3 5 9 4 E N / 0 1 . 0 1 *