CIMON Xpanel

Touch Panel Mountable HMI Device

User’s Manual

CIMON Xpanel

WINDOWS CE BASED NEW TOUCH

INSTALLATION GUIDE

SYSTEM OVERVIEW

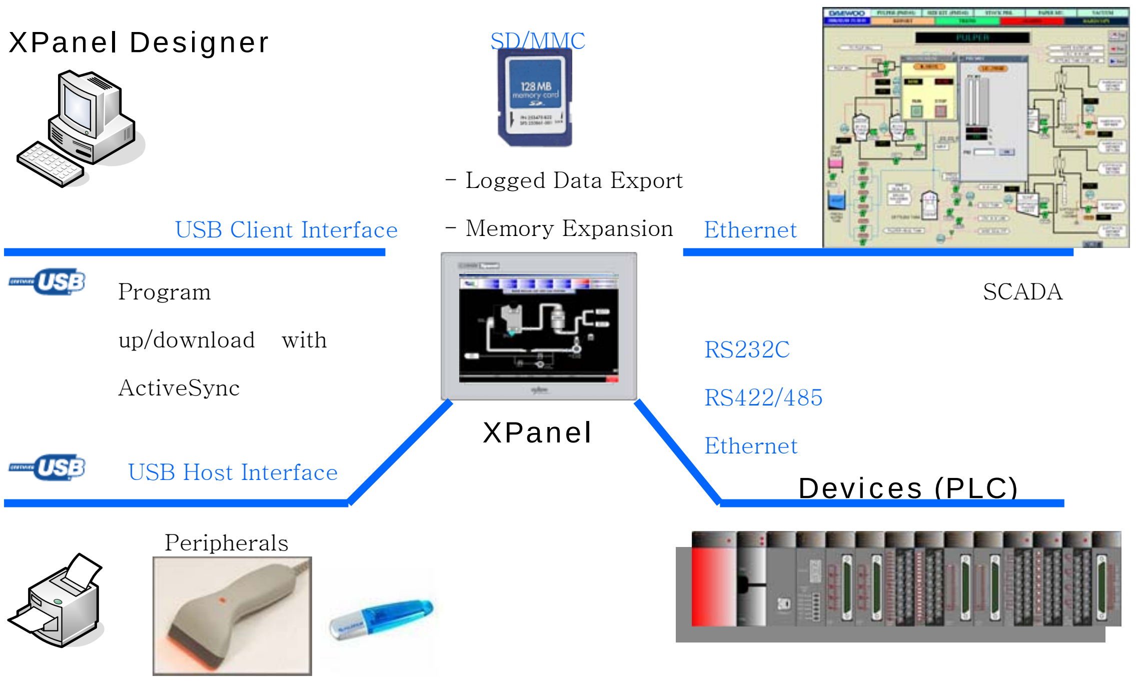
DATA LOGGER

RECIPE

SCRIPT

System Overview

1. Peripherals



XPanel supports various kinds of peripherals. For those peripheral connection, XPanel provides following interfaces :

(1) USB Client

This interface is used for XPanel Designer interface. By using this port, all configuration data files including the screen drawings can be up and downloaded. XPanel Designer software is a free software. It is provided by CD and can be downloaded through the internet. (http://www.kdtsys.com)

XPanel Designer is a Microsoft Windows based software. And for connection with XPanel, it uses another software on PC, the Microsoft ActiveSync. The ActiveSync is provided by CD, but also can be downloaded from Microsoft web sites.

(2) USB Host

This interface can be used for many kinds of peripherals such as printers, memory sticks, barcode scanners and keyboards. For connection of multiple of them, the USB hub can be utilized.

(3) SD/MMC Memory Slot

SD/MMC memory slot can be used for storage expansion and/or upgrading project files. The XPanel provides standard 8MB internal flash memory. If this storage was not enough for storing all project data files, external SD/MMC or USB memory can be used.

XPanel detects SD/MMC or USB memory automatically. And if there are required folder

1. Safety Precautions

Essential Safety Precautions

⚠ WARNING

System Design

- Do not create Xpanel graphic objects that could possibly endanger the safety of equipment and personnel. Damage to the Xpanel can cause an output signal to remain ON or OFF continuously and can cause a major accident. Therefore, design all monitoring circuits using limit switch to detect incorrect device movement.
- Do not create Xpanel graphic objects that control machine safety operations, such as an emergency stop. Switches to control machine safety operations must be installed as separated hardware switches.
- Design your system so that a communication fault between the Xpanel and the controller of the equipment can not make the equipment malfunction.
- Do not use the Xpanel as a warning device for critical alarms that can cause serious operator injury, machine damage or production stoppage.
- The Xpanel is not appropriate for use with aircraft control devices, aerospace equipments, central trunk data transmission (communication) devices, nuclear power control devices, or medical life support equipment, due to these devices' inherent requirements of extremely high levels of safety and reliability.
- When using the Xpanel with transportation vehicles (trains, cars and ships), disaster and crime prevention devices, various type of safety equipment, non-life support related medical devices, etc. redundant and/or failsafe system designs should be used to ensure the proper degree of reliability and safety.
- After the backlight of the Xpanel burns out, the touch panel of the Xpanel is still active. If an operator does not notice that the backlight burned out and touches the panel, a potentially dangerous machine miss-operation can occur. Therefore, do not use the Xpanel graphic objects for the control of any equipment safety mechanisms, such as Emergency Stop switches, etc. that protect humans and equipment from injury and damage.
- If the backlight of LCD suddenly turns OFF, use the following steps to determine if the backlight is actually burned out.

  Step1. If your Xpanel is not set to "Standby Mode" and the screen has gone blank, the backlight of LCD is burned out.

Step2. Or, if your Xpanel is set to “Standby Mode” but touching the screen can not wake Xpanel up, your backlight is burned out.

Installation

- High voltage runs through the Xpanel. Do not disassemble the Xpanel. otherwise an electric shock can occur.
- Do not modify the Xpanel unit, since the modified Xpanel cause a fire or an electric shock.
- Do not use the Xpanel in flammable gasses, since operating the Xpanel in flammable gasses may cause an explosion.

Wiring

- To prevent an electric shock, be sure to confirm that the power cord of the Xpanel is not connected to the main power before connecting any cord, cables or line to the Xpanel.
- Do not use power beyond the specified voltage range of the Xpanel. Doing so may cause a fire or an electric shock.

Maintenance

- The Xpanel uses a lithium battery to back up its internal clock data. If the battery is incorrectly replaced, the battery may explode. To prevent this, please do not replace the battery yourself. When the battery needs to be replaced, please contact your local Xpanel distributor.

⚠ CAUTIONS

Installation

- Be sure to securely connect all cable connectors to the Xpanel. A loose connection may cause incorrect input or output.

Wiring

- Ground the FG line of the Xpanel separately from FG lines of other units. Putting these FG lines too close may cause an electric shock or unit malfunction. Be sure to use a grounding resistance of 100Ω or less and a 2 ㎟ or thicker wire, or applicable standard of your country.
- Correctly wire the Xpanel, be sure that the rated voltage and terminal layout are within the designated range. If the voltage supplied differs from the rated voltage, or incorrect wiring or grounding is performed, it may cause a fire or unit malfunction.
- Use only the designated torque to tighten terminal block screws of the Xpanel. If these screws are not tightened firmly, it may cause a short circuit, fire or Xpanel malfunction.
- Be careful that the metal filings and wiring debris do not fall into the Xpanel, since they can cause a fire, Xpanel malfunction or incorrect operation.

Maintenance

- The LCD contains a powerful irritant and if for any reason the panel is damaged and this liquid contacts any part of your body, be sure to wash that area with running water for 15 minutes. If any of this liquid enters your eye, flush your eye for 15 minutes with running water and contact a physician.

Unit Disposal

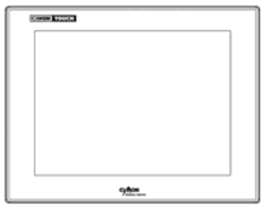
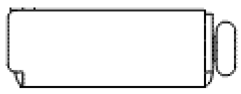
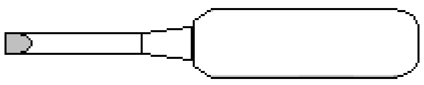
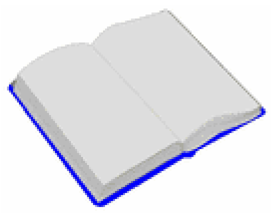
- When this unit is disposed of, it should be done so according to your country’s regulation for similar types of industrial waste.

General Safety Precautions

- Do not strike the touch panel with a hard or pointed object, or press on the touch panel with too much force, since it may damage the touch panel or the display.
- Do not install the Xpanel where the ambient temperature can exceed the allowed range. Doing so may cause the Xpanel to malfunction or shorten its operation life.
- Do not restrict or limit naturally occurring rear-face ventilation of the Xpanel, or storing or using the Xpanel in an environment that is too hot.
- Do not use the Xpanel in areas where large, sudden temperature changes can occur. These changes can cause condensation to form inside the unit, possibly causing the unit to malfunction.
- Do not allow water, liquids, metal or charged particles to enter inside the Xpanel, since they can cause either a Xpanel malfunction or an electrical shock. The allowable pollution degree is 2.
- Do not store or use the Xpanel in direct sunlight, or in excessively dusty or dirty environments.
- Do not store or use the Xpanel where strong jolting or excessive vibration can occur.
- Do not store or use the Xpanel where chemicals (such as organic solvents, etc.) and acids can evaporate, or where chemicals and acids are present in the air.
- Do not use paint thinner or organic solvents to clean the Xpanel.
- Do not store or operate the LCD display in areas receiving direct sunlight, since the sun's UV rays may cause the LCD display's quality to deteriorate.
- If you store the Xpanel in areas where the temperature is lower than allowed level, the liquid of the LCD will congeal and the LCD can be damage. Conversely, if the storage area's temperature becomes higher than the allowed level, the liquid of the LCD will become isotropic, causing irreversible damage to the LCD. Therefore, be sure to store the panel only in areas where temperatures are within those specified in this manual.
- After turning the Xpanel OFF, be sure to wait a few seconds before turning it ON again. If the Xpanel started too soon, it may not start up correctly.
- Due to the possibility of unexpected accidents, you must back up the project data of the Xpanel regularly.

2. Package Contents

The following items are contained in the package of the Xpanel. Before using the Xpanel, please confirm that all items listed here are present.

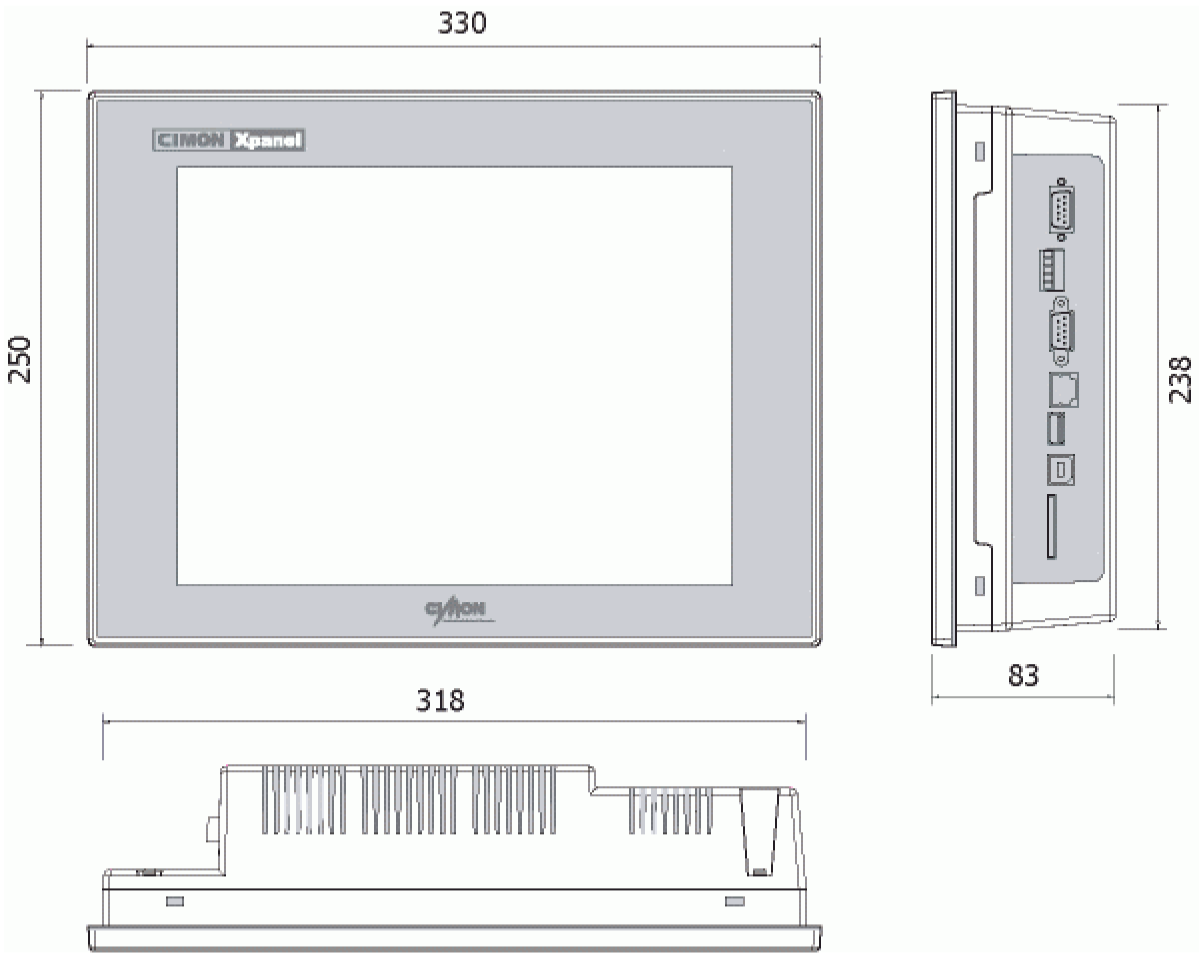
■ Xpanel Unit	■ Fastener: 8 ea.
	
■ 5P Connector	■ Screw Driver
	
■ USB Cable	■ CD
■ Installation Guide	
	

This unit has been carefully packed, with special attention to quality. However, should you find anything damaged or missing, please contact your local Xpanel distributor immediately.

3. Dimensions

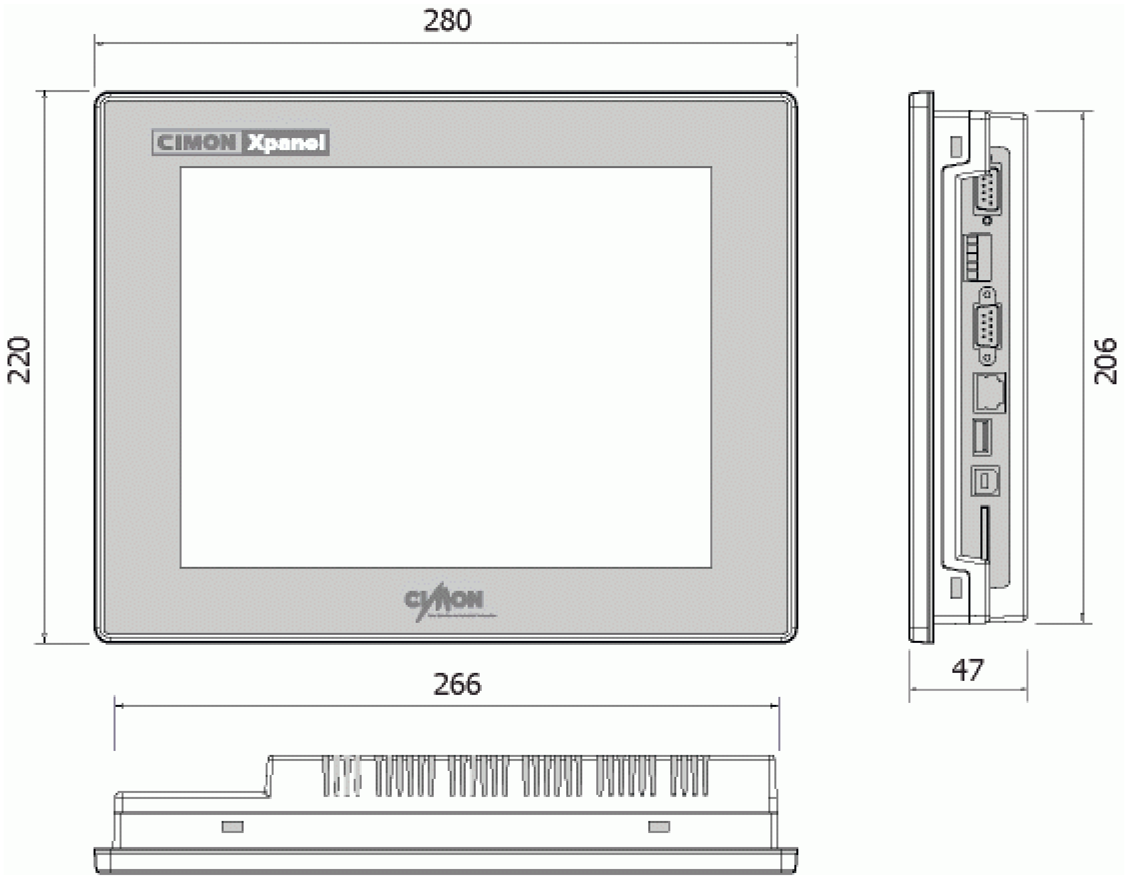
Unit: mm

- XT12CA



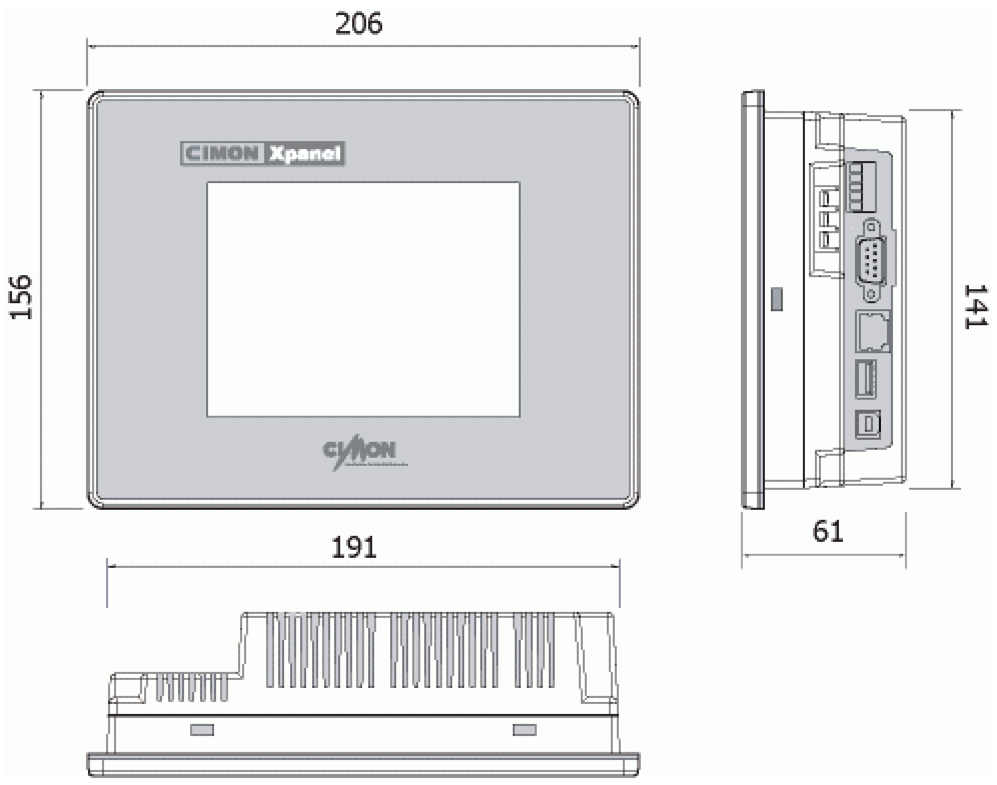
Unit: mm

- XT10CA



Unit: mm

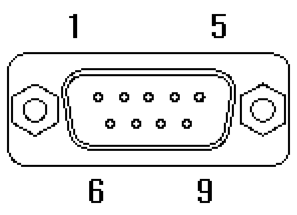
- XT06CA, XT05SA, XT05MA



4. Interfaces

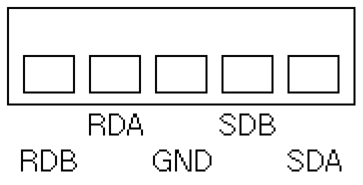
■ COM1: RS-232C

This interface is used to connect the Xpanel to the host (PLC), via an RS-232C cable. When you use COM1 RS-232C port, you must not use COM1 RS-485/422 port.

Connector	Pin No	Name	Description
	1	DCD	Data Carrier Detect
	2	RD	Receive Data
	3	TD	Transmit Data
	4	DTR	Data Terminal Ready
	5	SG	Signal Ground
	6	DSR	Data Set Ready
	7	RTS	Request To Send
	8	CTS	Clear To Send
	9	RI	Ring Indicator

■ COM1: RS-422/485

This interface is used to connect the Xpanel to the host (PLC), via an RS-422/485 cable. When you use COM1 RS-422/485 port, you must not use COM1 RS-232C port.

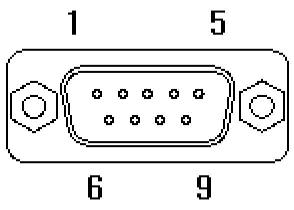
Connector	Pin No	Name	Description
	1	RDB	Receive Data B
	2	RDA	Receive Data A
	3	GND	Ground
	4	SDB	Send Data B
	5	SDA	Send Data A

- If you connect the Xpanel to the host via an RS-485 cable (2wire), you must connect SDA and SDB lines.
- The RS-485 of the Xpanel runs under auto toggle mode.
- To reduce the risk of damaging the RS-422 circuit, be sure to connect the SG terminal.

■ COM2: RS-232C

XT10A and XT12A series units have this interface.

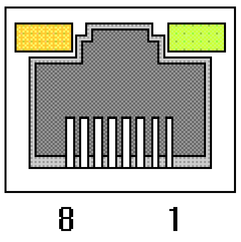
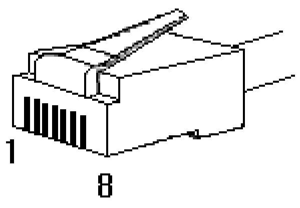
This interface is used to connect the Xpanel to the host (PLC), via an RS-232C cable. You can use pin 2, 3, 5 of this interface.

Connector	Pin No	Name	Description
	1		
	2	RD	Receive Data
	3	TD	Transmit Data
	4		
	5	SG	Signal Ground
	6		
	7		
	8		
	9		

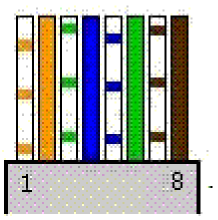
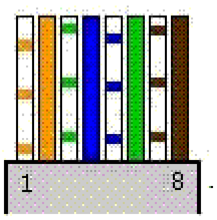
- The serial port of the Xpanel is not isolated. When the host(PLC) unit is also not isolated
- Inside the Xpanel unit, the SG (Signal Ground) and FG (Frame Ground) terminals are connected to each other.
- When connecting an external device to the Xpanel with the SG terminal, ensure that no short-circuit loop is created when you setup the system.

■ Ethernet

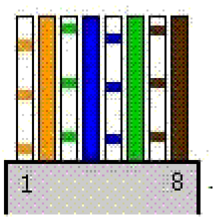
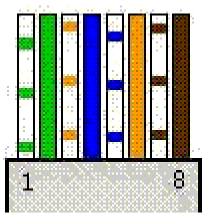
This interface complies with the IEEE802.3 for Ethernet (10BaseT/100BaseTX).

RJ45 Connector	RJ45 Jack
	

Direct Cable: Host <-> HUB

Cable	No	Color	Color	No	Cable
	1	Orange/W	Orange/W	1	
	2	Orange	Orange	2	
	3	Green/W	Green/W	3	
	4	Blue	Blue	4	
	5	Blue/W	Blue/W	5	
	6	Green	Green	6	
	7	Brown/W	Brown/W	7	
	8	Brown	Brown	8	

Crossover Cable: Host <-> Host

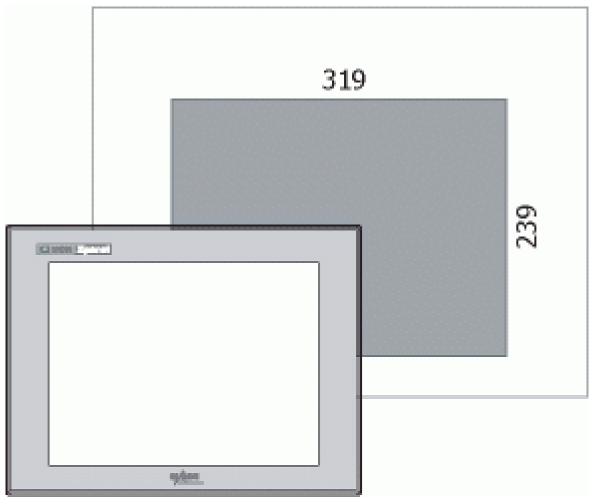
Cable	No	Color	Color	No	Cable
	1	Orange/W	Green/W	1	
	2	Orange	Green	2	
	3	Green/W	Orange/W	3	
	4	Blue	Blue	4	
	5	Blue/W	Blue/W	5	
	6	Green	Orange	6	
	7	Brown/W	Brown/W	7	
	8	Brown	Brown	8	

5. Installation

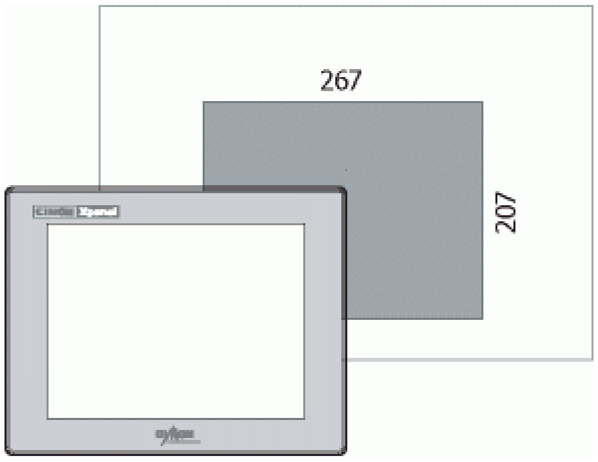
■ Create a Panel Cut and insert the Xpanel into the panel from the front

Unit: mm

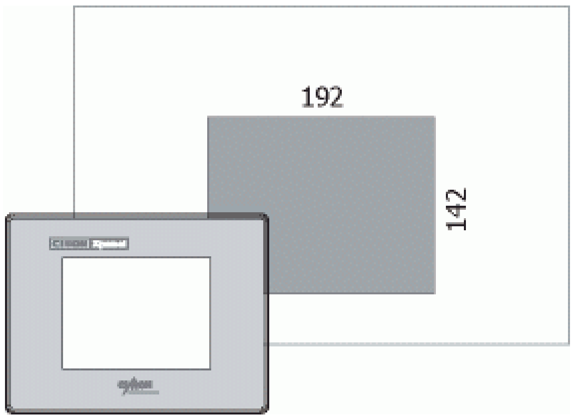
- XT12CA



- XT10CA

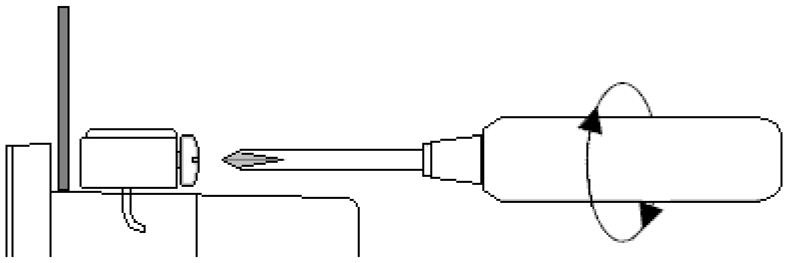


- XT06CA / XT05SA / XT05MA



■ Attach the Installation Fasteners from inside the panel

Insert each the hook of faster into slot and tighten it with a screwdriver.



- Tightening the screws with too much force can damage the case of the Xpanel.
- The necessary torque is 0.5 N•m.
- Depending on the thickness of installation panel, the number of installation fasteners used may need to be increased to provide the desired level of moisture resistance.

6. Wiring

WARNINGS

- To avoid an electric shock, when connecting the Xpanel power cord terminals to the power terminal block, confirm that the Xpanel power supply is completely turned OFF, via a breaker, or similar unit.
- Since there is no power switch on the Xpanel unit, be sure to attach a breaker-type switch to its power cord.

- To avoid a short caused by loose ring terminals, be sure to use ring terminals with an insulating sleeve.
- When the FG terminal is connected, be sure the wire is grounded. Not grounding the Xpanel unit will result in excess noise and vibration.

■ Connecting the Xpanel Power Cord

When connecting the power cord, be sure to follow the procedures given below.

- Confirm that the Power Cord is unplugged from the power supply.
- Unscrew the screws from the middle three (3) terminals, align the Ring Terminals and re-attach the screws.

- Confirm that the ring terminal wires are connected correctly.
- The torque required to tighten these screws is 0.5 to 0.6N•m.

1) Power Supply Caution

Please pay special attention to the following instructions when connecting the power cord terminals to the Xpanel unit.

- If the power supply voltage exceeds the GP's specified range, connect a voltage transformer.
- Between the line and the ground, be sure to use a low noise power supply. If there is still an excessive amount of noise, connect a noise reducing transformer.
- The power supply cord should not be bundled with or kept close to main circuit lines (high voltage, high current), or input/output signal lines.

- Connect a surge absorber to handle power surges.
- To reduce noise, make the power cord as short as possible.

2) Grounding Caution

- When grounding to the rear face FG terminal of the Xpanel, (on the Power Input Terminal Block), be sure to create an exclusive ground.
- Inside the Xpanel unit, the SG (Signal Ground) and FG (Frame Ground) terminals are connected to each other.
- When connecting an external device to the Xpanel with the SG terminal, ensure that no short-circuit loop is created when you setup the system.

7. Specification

		XT05MA	XT05SA	XT06CA	XT10CA	XT12CA
Panel Size		5.6"		6.4"	10.4"	12.1"
LCD		Mono LCD	CSTN LCD	Color TFT	Color TFT	Color TFT
Resolution		320 x 240		640 x 480	800 x 600	800 x 600
Color		4-Gray	256	65535	65535	65535
RAM		SDRAM 64MB				
Flash		Flash 64MB				
OS		Microsoft Window CE .NET 4.2				
Interface	Ethernet	10 Base T / 100 Base TX				
	COM1	RS-232C/422/485 (485 Auto Toggle)				
	COM2	None			RS-232	
	USB	None			1 CH (Host)	
	Tool	1 CH				
	SD Card	None			1 CH	
Power		A: 100~240VAC, D: DC24V				
Design Tool		Xpanel Designer				
Dimension		206x156x61			280x220x47	330x250x83
Panel Cut		192x142			267x207	319x239
Protection		IP65 (Front)				

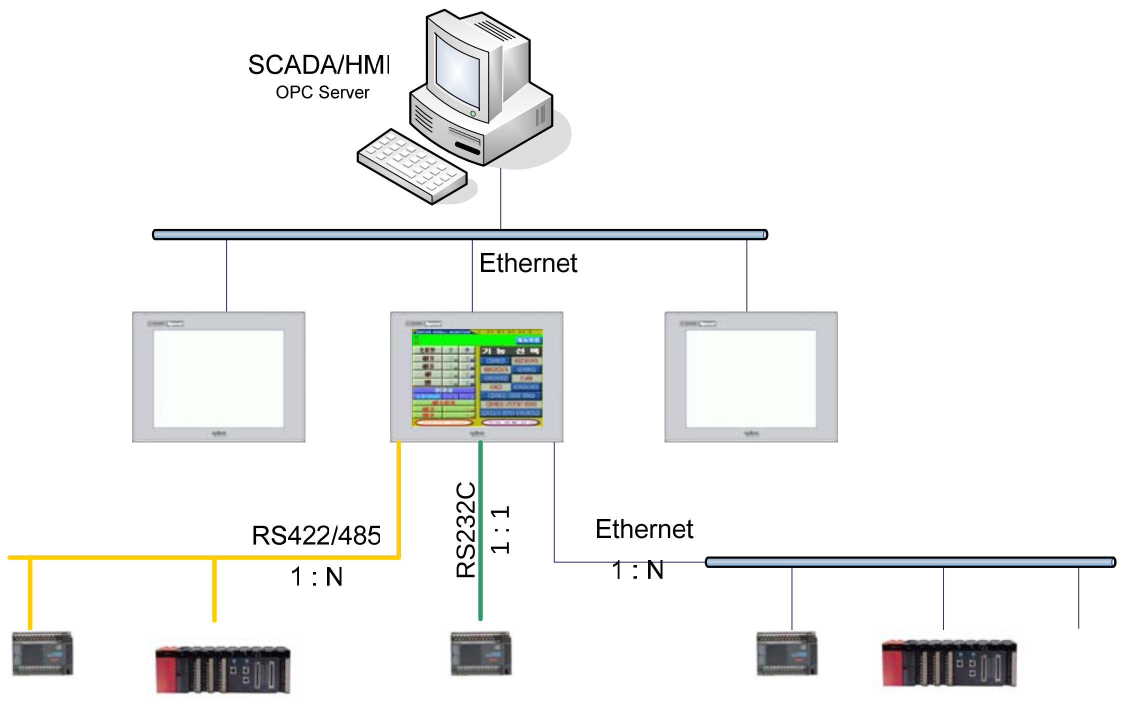
■ Example of model name

05/06/10/12	LCD Size
M/S/C	Initial character of Mono/STN/Color
A	Revision
-A/-D	AC/DC (AC: 100~240, DC: 24V)

- XT05MA-D -> 5.6" Mono DC24V input
- XT10CA-A -> 10.4” Color TFT LCD AC input

and files in external memory, XPanel will load the project files from that external memory at startup. In this case, the project files of internal flash memory will be ignored.

2. Network



XPanel supports not only 1:1 network but also multi-drop topology. And more, XPanel can communicate with several different devices and networks on the same time.

(1) Ethernet Communication Port

Ethernet communication facility of XPanel is used in various purposes. The main usage of Ethernet port is for communicating with devices. Many different kinds of devices can be linked with XPanel in the same network at the same time.

Other usage of Ethernet is network with SCADA system. XPanel can be a data server for SCADA on Ethernet. The OPC server will be provided free.

And one more usage of Ethernet is file sharing with PCs which are located in the same network. Xpanel supports the MS Windows network. That means the XPanel can access the shared files of PCs on Windows network. This functionality is very useful when exporting the CSV formatted file from XPanel.

(2) RS232C(RS422/485) Communication Port

XPanel provides a standard RS232C/RS422/485 communication port. And the mode (RS232C or RS422/485) of port is switched by only software configuration. This port supports full signal lines for standard modem connection.

(3) Additional RS232C Communication Port

XPanel provides additional one RS232C port. This port supports null modem connection.

3. XPanel Designer

XPanel Designer is a free software. It can be run on the standard Microsoft Windows platform such as Windows XP. By using XPanel Designer, following works can be carried out :

(1) Device Configuration

According to the connected devices, all the communication parameters can be configured such as communication media, port, baudrate, station number etc. Every configured devices on network are given it's own name, and this name will be used in TAG database configuration.

(2) TAG Database

The TAG is the basic unit of data processing in XPanel. Three types of TAGs are provided according to the data type.

The first type of TAG is the digital TAG. Digital TAG represents two different states and can be used for general 1 bit input or output point of the device. The second type of TAG is the analog TAG. This type of TAG represents a number. A number can be a binary word, BCD or floating point value and some additional data processing can be accomplished on this type of TAG such as scaling. The last type of TAG is the string TAG. The string TAG represents a string data. This string data can be used for the message display.

If the type of TAG is a digital or analog, it could be assigned a physical address of device. This kind of TAG is called as 'Real TAG' and XPanel unit polls the value of this TAG from device by communication, when it is needed. A TAG which is not assigned the physical address of device is called as 'Virtual TAG'. Virtual TAG is used for internal data calculation, script programming and string data manipulation.

(3) Screen Editor

XPanel Designer provides it's own graphic editor. It is an object based graphic editor. Each graphic object can be configured with some animation controls. There are many kinds of animation controls supported such as blinking, visualizing, coloring, moving, rotating, digitizing, zooming and more. These all animations are triggered by the value of designated TAG in XPanel unit.

The XPanel provides abundant graphic libraries and supports true type fonts, also.

(4) Script Programming

XPanel provides 'C' style script language of it's own. This script language supports inherent 'C' language keywords such as 'switch-case' statement. And more, XPanel unit provides priority controlled multi-threading environment for that script program.

This script language can be used not only for data processing but also for device control.

(5) XPanel Unit Upgrade

Every XPanel Designer releases have it's own version of binary files for XPanel unit. And when connected first time with XPanel unit, the XPanel Designer detects the version of XPanel unit, and if needed, upgrade the binary files in XPanel unit automatically.

XPanel Designer provides a tool for making memory image of SD/MMC or USB. This image includes project folders and files for XPanel unit. On start-up, XPanel unit searches this image in external memory. If it was found, XPanel unit uses this image as a project data, and the one in internal flash memory will be ignored. This is useful when a user do not want to use a PC or notebook for project data update. The user can change the configuration or the design of screen with a memory card only.

(6) Configure Other Functions

XPanel provides many special functions for XPanel application such as data logger, alarm, trend, recipe and network service. All these functions can be configured in XPanel Designer.

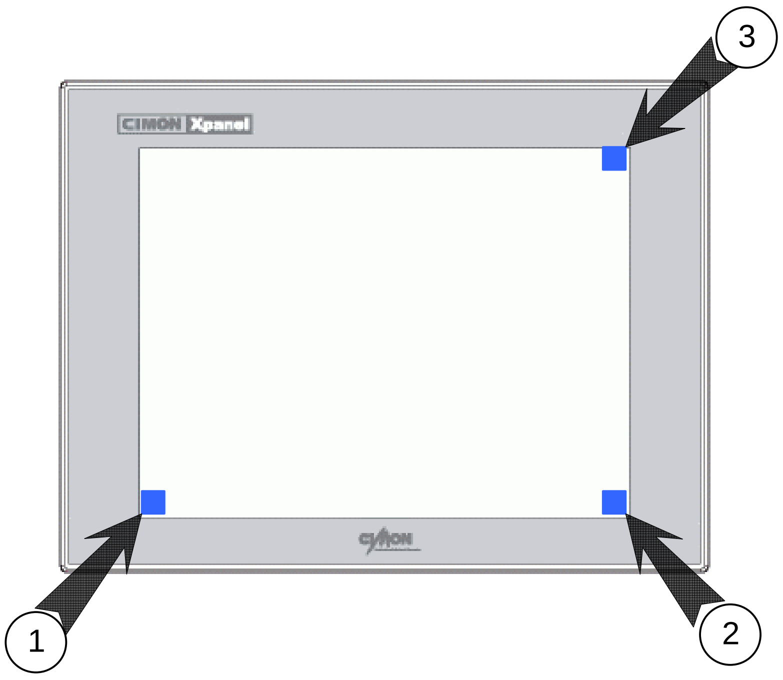
4. XPanel Unit

XPanel unit always searches a project files for execution at start up. Following diagram shows the order of that search. If the project files are found at one of three locations, XPanel unit load those project files to execute.

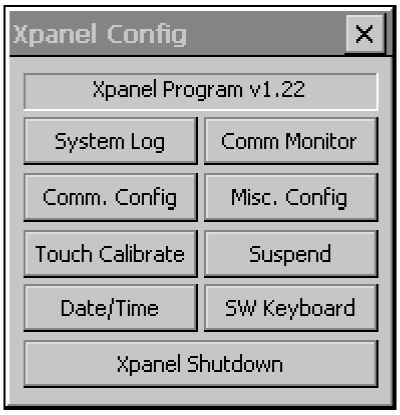
$$\begin{tikzcd}
\boxed{\text{Start-Up}} \arrow[r] & \boxed{\text{SD/MMC}} \arrow[r] & \boxed{\text{USB Memory}} \arrow[r] & \boxed{\text{Internal Flash}}
\end{tikzcd}$$

After the loading of project files, all configured tasks are executed concurrently. The task execution in an XPanel unit is performed by time-sharing and multi-threading with priority. Graphic processing and file manipulation in XPanel are processed by time sharing. But, almost other tasks are executed by means of multi-threading. The priority based multi-threading makes it possible to process real-time data. Communication and script are good examples for priority based multi-threaded task.

On the other hand, XPanel provides some tools for maintenance during runtime. These tools can be visualized by three step touch screen operations. Following picture shows these operations of touch screen.



After these touch operations, a dialog box will be appeared on screen and you can see the following services for maintenance.



(1) System Log

An XPanel logs the system activities in memory during its runtime. This service makes you can see those logged messages.

(2) Communication Monitor

Communication frames between the XPanel and devices can be visualized with this service.

(3) Comm. Port Configuration

Communication parameters can be configured by this service. IP address, baudrate and other parameters of communication port are listed on the dialog box, and can be changed by common control interfaces of Windows.

(4) Misc. Configurations

LCD backlight and beep sound can be configured by this service.

(5) Touch Calibration

This service provides a chance to calibrate touch panel of XPanel.

(6) Clock Setting

The real time clock can be adjusted by this service.

(7) Software Keyboard

The XPanel has a software keyboard interface. This service visualizes or hides the software keyboard on screen.

(8) XPanel Shutdown

This service shutdowns the XPanel program on XPanel unit.

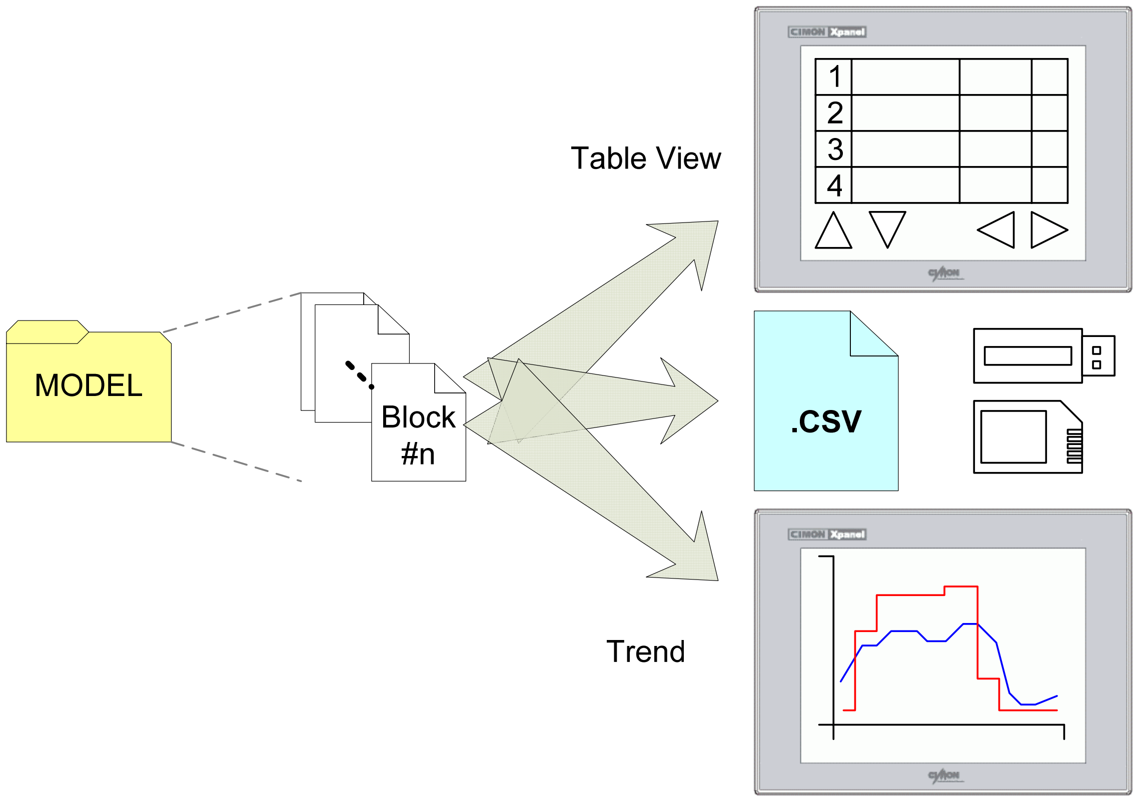
(9) Suspend

This service shutdowns the operating system of XPanel unit. When the IP address is changed, this service must be activated. After the system shutdown, the system will not be restarted automatically. The external power must be turned off and on again for system restart.

Data Logger

1. Overview

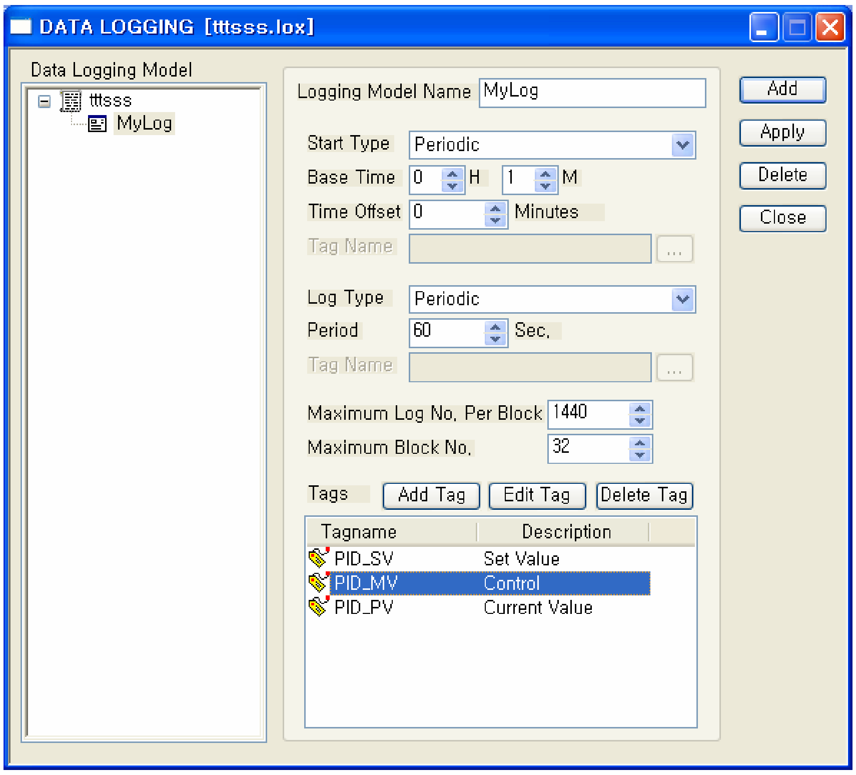
The data logger of XPanel has three kinds of application. The first one is the displaying the logged data on screen in tabular form. The graphic editor in XPanel Designer provides a special object named as 'Data Log' for this purpose. This object displays a part of logged data on screen and makes it possible to examine and scroll all other logged data by touch input. The second one is the data exporting by CSV formatted file. CSV formatted file can be read by application software in PC such as Microsoft Excel. For that exporting, XPanel provides some command and script functions. The third one is the visualizing logged data with the trend. This trending function is under the developing. (This function will be added on April, 2006)



2. Model of Data Logger

For utilizing a data logging function in the XPanel, a data logging 'Model' has to be declared in XPanel Designer. A model is a group of logging configuration and more than one model can be declared in a project. All declared logging models are processed concurrently and independently in an XPanel unit.

The condition of logging-start, trig, size and TAGs are declared in each model. The following dialog screen shot is an example of model configuration.


DATA LOGGING [tttsss.lox]

Data Logging Model
- tttsss
  - MyLog

Logging Model Name: MyLog

Start Type: Periodic
Base Time: 0 H 1 M
Time Offset: 0 Minutes
Tag Name:

Log Type: Periodic
Period: 60 Sec.
Tag Name:

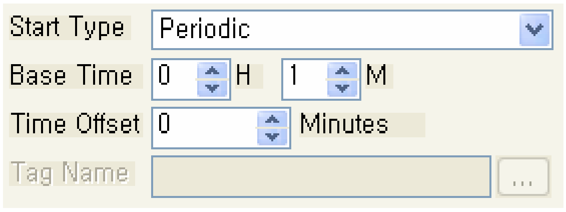
Maximum Log No. Per Block: 1440
Maximum Block No.: 32

Tags

Tagname	Description
PID_SV	Set Value
PID_MV	Control
PID_PV	Current Value

(1) Log Task Invoking Condition

XPanel provides 5 different **start types**. Each of these is based on period, command, tag and time. Whenever a new log task is started, a new block file will be created and the old task will be stopped and closed automatically. The name of a block file is automatically generated based on the started time.


Start Type: Periodic
Base Time: 0 H 1 M
Time Offset: 0 Minutes
Tag Name:

- **Periodic**

  A log task is started periodically. The period of task invoking is configured in the unit of minute. During one day, log task is started on every multiple of designated minute. For example, if the '**Base Time**' was configured as "1H 30M", the log task will be invoked on 00:00, 01:30, 03:00, 04:30, …, 22:30.

This example assumes that '**Time Offset**' is configured as '0'. The 'time offset' is a delay of start. If it is assigned as 10, the log task of previous example will be invoked on 00:10, 01:40 and so on.

- **Called (by command)**

  This type of invoking is triggered by a command. A command can be issued by touch operation or by script program. The command has a following format :

  DataLog("ModelName", 1)

- **Trigger TAG**

  The log task is triggered by the state change of a TAG. When the value of a TAG changes from zero to non-zero, the log task will be invoked. The value change from non-zero to zero does not affect to the log task. Designate the name of TAG in '**Tag Name**' field.

- **Enable TAG**

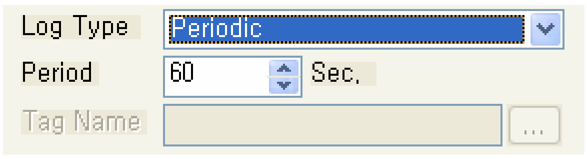
  The log task is triggered by the state change of a TAG. When the value of a TAG changes from zero to non-zero, the log task will be invoked. And when the value changes from non-zero to zero, the log task will be terminated. That is, the log task keeps logging only while the value of TAG is non-zero. Designate the name of TAG in '**Tag Name**' field.

- **On Time**

  The log task is invoked only once in a day. Set the time of log in '**Base Time**' field.

(2) Log Trigger Condition

After being invoked, the log task writes its TAG data on file at one of two different kinds of event. One is a periodic event, and the other is an event from the trigger TAG. Every model has to be assigned one of two events as its log trigger event.



Log Type: Periodic
Period: 60 Sec.
Tag Name:

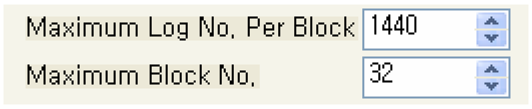
- **Periodic**

  The logging is triggered at every designated period of time. The period has to be set in '**Period**' field. The unit is second.

- **Tag Value**

  The logging is triggered when the value of TAG designated in '**Tag Name**' field is changed from zero to non-zero value.

(3) Log termination



Maximum Log No. Per Block: 1440
Maximum Block No.: 32

Invoked log task is stopped when one or more of following conditions are met.

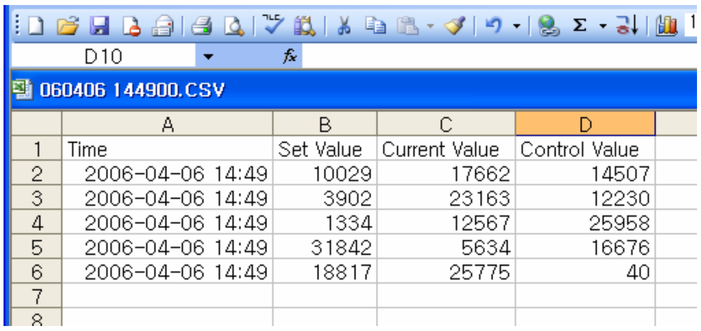
- When the number of logging is reached at the designated count in '**Maximum Log No. Per Block**'.
- When a new log task invoking condition is met, current logging task is forced to terminate and a new task will be invoked.
- A log task can be stopped by external command issued by a touch operation or script program.

DataLog("ModelName", 0)

- In case of the log task invoked by '**Enable TAG**', the task will be stopped by the value of 'Enable TAG'. If the TAG value is changed from non-zero to zero, the task will be terminated.

(4) Block File

Note that every one log task generates one block file. When a new block file is created, old block files are not removed immediately from the storage until the number of total files reaches to the designated number in '**Maximum Block No.**' field. After that the number of block files reaches to the limit, the oldest file is removed from storage.



060406 144900.CSV

	A	B	C	D
1	Time	Set Value	Current Value	Control Value
2	2006-04-06 14:49	10029	17662	14507
3	2006-04-06 14:49	3902	23163	12230
4	2006-04-06 14:49	1334	12567	25958
5	2006-04-06 14:49	31842	5634	16676
6	2006-04-06 14:49	18817	25775	40
7				
8				

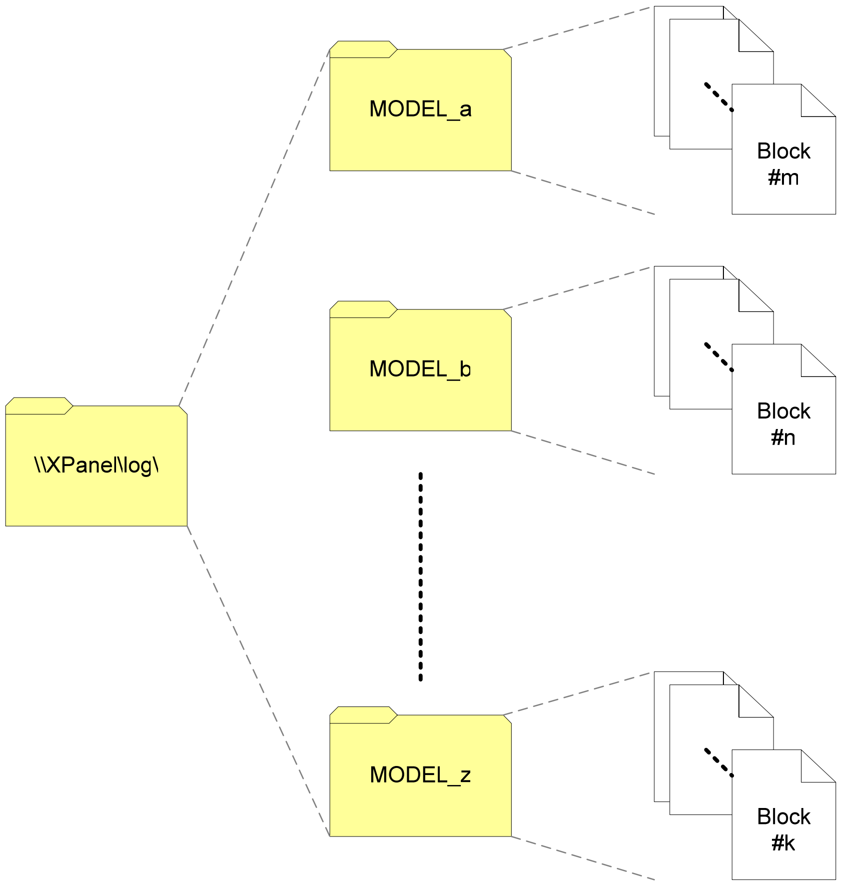
A block file is given its name with the time created. Naming format is "**YYMMDD HHmmSS.LOG**". But, a block file cannot be read with other program or tools, it must be converted to well-known format for further utilization. For this purpose, XPanel provides several conversion commands. These commands can be issued by touch operation or script program. They convert the block files to CSV (Comma Separated Values) formatted text file. The CSV formatted file can be read in Microsoft Excel or other utility programs in PC.

MakeCsv("ModelName", BlkNo)
MakeCsvUsb("ModelName", BlkNo)

'**MakeCsv()**' command converts block file to CSV formatted file, and stores output file in the SD/MMC memory. '**MakeCsvUsb()**' command has the same functionality with

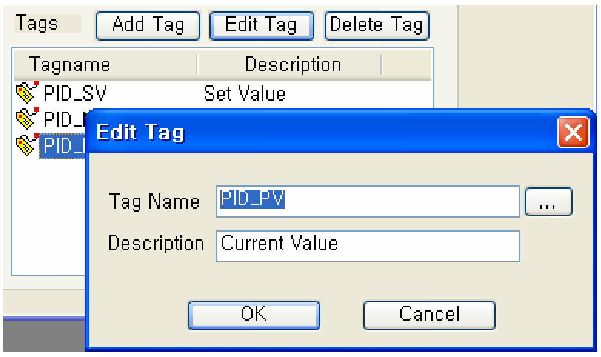
‘MakeCsv()’, but only one difference is that it stores output file in the USB memory. The second parameter ‘**BlkNo**’ can be a number from zero to ‘Maximum Block No.’. If it is assigned as zero, the command converts current logging block file. If there is no current logging task, the command will not make any output file and do nothing. If the ‘BlkNo’ parameter was assigned as a non-zero positive value, it specifies one of logged block file. A value of ‘1’ specifies the latest block file, and ‘2’ specifies the secondly latest block file and so on. Converted file has the same name of the block file. (“**YYMMDD HHmmSS.CSV**”)

Following diagram shows the storage image of XPanel. All the block files in a log model located in the same folder, and this folder is named by its model name. If there are several models already defined, these all model will have their own folders and these are located in “**\\XPanel\Log\**”.



(5) TAGs to Log

Analog and digital TAGs (up to 32 TAGs) can be assigned to a log model. Following dialog box shows the situation of adding a new TAG in a model. The ‘**Description**’ field of the dialog box will be used as a column name of CSV file. As noted before, a block file can be converted to a CSV file by the command from touch operation or a script program.



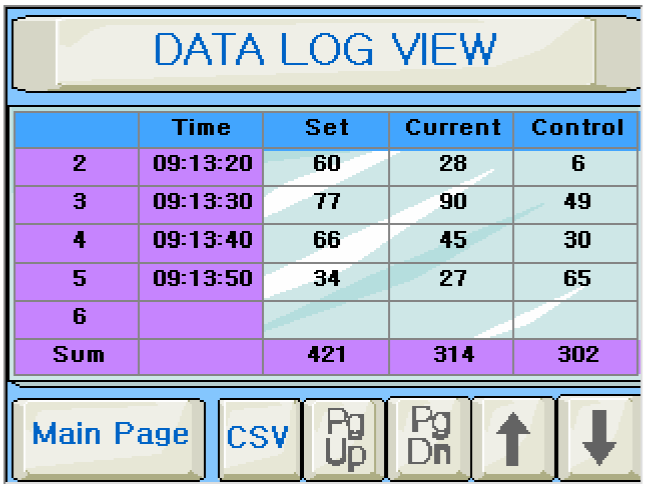
Tagname	Description
PID_SV	Set Value
PID_	
PID_	

Tag Name PID_PV

Description Current Value

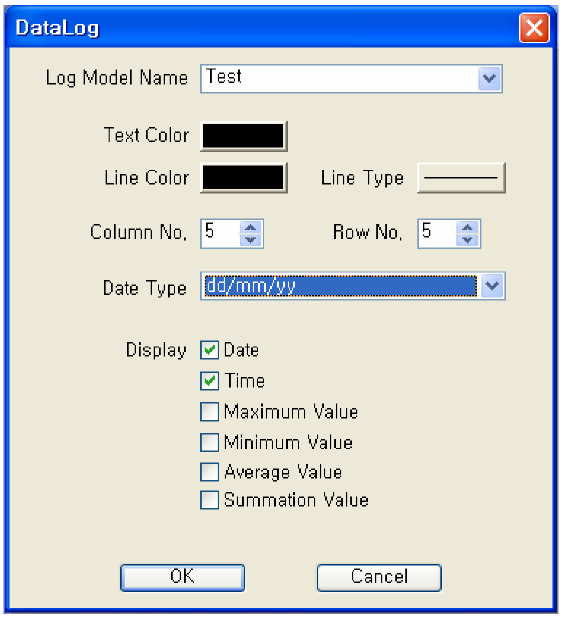
3. Table View : Data Log Object

The logged data can be seen on screen in tabular form. For that, a special graphic object named as '**Data Log**' is provided in graphic editor. Following captured screen is an example of tabular display of logged data.


DATA LOG VIEW

	Time	Set	Current	Control
2	09:13:20	60	28	6
3	09:13:30	77	90	49
4	09:13:40	66	45	30
5	09:13:50	34	27	65
6				
Sum		421	314	302

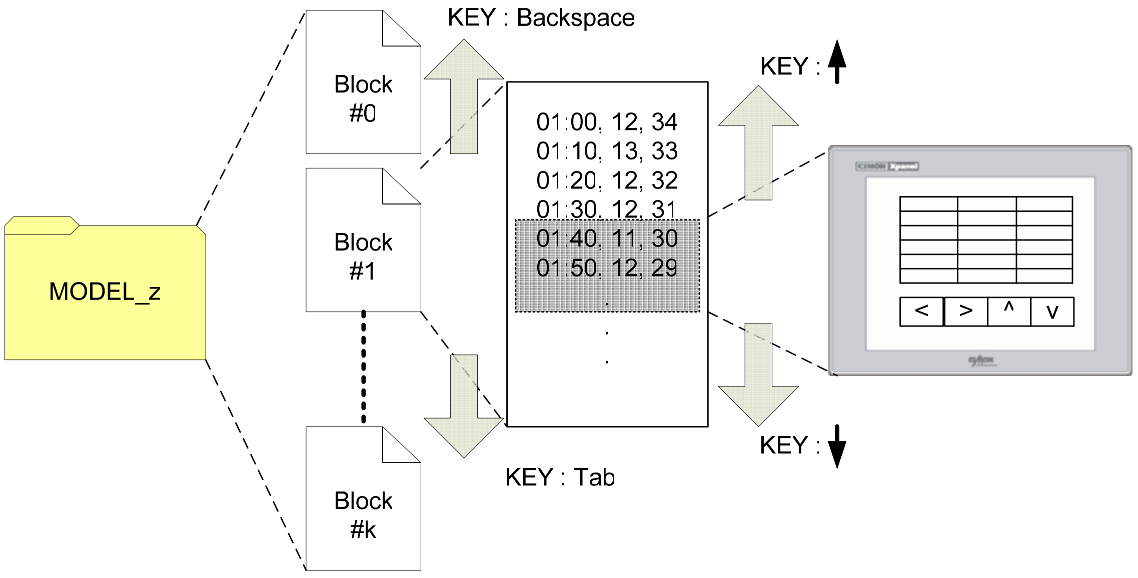
The shape of table can be configured in graphic editor through following dialog box. The colors of text and line, the number of column and row and the format of time and date can be configured in this dialog box. And also, the data log object calculates and displays some statistical data from the block file. Maximum, minimum, average, and summation of logged data in a block file can be displayed on data log object.


DataLog

Log Model Name: Test
Text Color: ■
Line Color: ■
Line Type: ———
Column No. 5
Row No. 5
Date Type: dd/mm/yy

Display
- [x] Date
- [x] Time
- [ ] Maximum Value
- [ ] Minimum Value
- [ ] Average Value
- [ ] Summation Value

A data log object shows a part of one block file. Following diagram explains that. Every data logging model has block files. The number of block files is limited to the assigned number in "Maximum Block No." field of model configuration dialog box. When the tabular view page is opened, XPanel will try to read the latest completely logged block file. And if there is a block file, XPanel displays the contents of it. This file's block number is '1'. The block number '0' file means the block file under the logging at the moment.



The data log object provides a method for traversing between block files to the operator. The traversing is performed by keyboard inputs (see the following table). XPanel can accept the keyboard inputs through the USB port. But also the operator can make a keyboard input by touch operation. Refer to the touch '**action**' type in 'object configuration' dialog box.

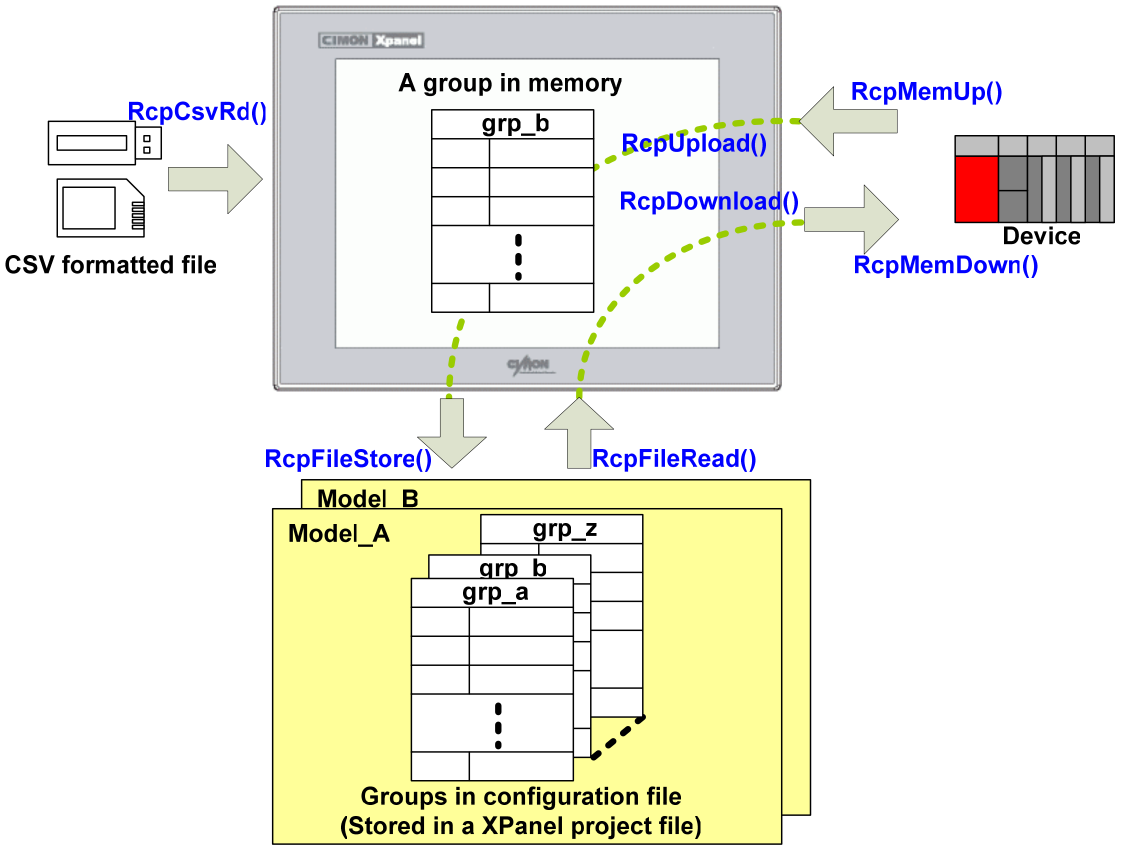
Key Input	Table Movement
Backspace	Change the current displaying block file to the one of more recent.
Tab	Change the current displaying block file to the one of more past.
Left Arrow	Scroll the display cell to the left
Right Arrow	Scroll the display cell to the right
Up Arrow	Scroll the display cell to the up
Down Arrow	Scroll the display cell to the down
Home	Move the display cell to the first logged data of current block file

Recipe

1. Overview

The recipe service of XPanel supports the multiple models. This makes it possible to manipulate different recipe for multiple devices with an XPanel.

In an XPanel, every recipe model has its own name, parameters and group data. Up to 1024 group data can be registered in a recipe model. But at any time, only one group data can be manipulated in XPanel. The reason is that a group data must be located in the memory of XPanel for manipulation. A group data can be transferred between four types of media. And the XPanel memory is located at the center of these transfers. Following picture shows paths of group data movement.



Each of group data movement is issued by the internal functions for recipe operation. More precise explanations can be found on the script section of this manual.

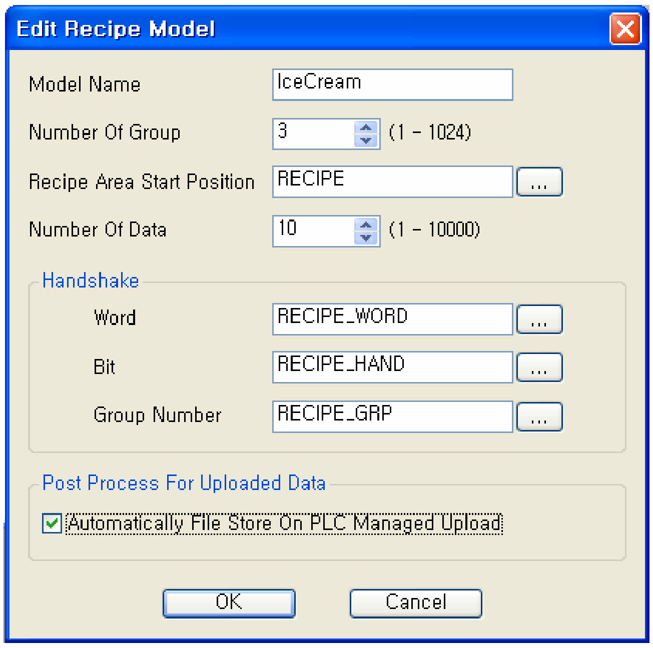
- A group data can be imported from external memory through CSV formatted file. CSV formatted file can be generated by the spreadsheet utilities such as MS-Excel on PC. : **RcpCsvRd()**
- A group data can be uploaded from the device through communication. For normal recipe operation, the communication driver must support the recipe functions. This can be

known from the manual of the communication driver. Refer to the driver’s manual. : **RcpMemUp(), RcpMemDown()**

- The group data in memory can be stored to the recipe configuration file. The stored group data will be maintained while power off. And the function for reading a group data from configuration file is provided also. : **RcpFileStore(), RcpFileRead()**
- Compounded functions are provided for convenience. **RcpUpload()** function is a compounded form of RcpMemUp() and RcpFileStore(). And **RcpDownload()** function is a compounded form of RcpFileRead() and RcpMemDown().

2. Model Configuration

A recipe model is defined by following dialog box of XPanel Designer.



Edit Recipe Model

Model Name	IceCream
Number Of Group	3 (1 – 1024)
Recipe Area Start Position	RECIPE
Number Of Data	10 (1 – 10000)

Handshake

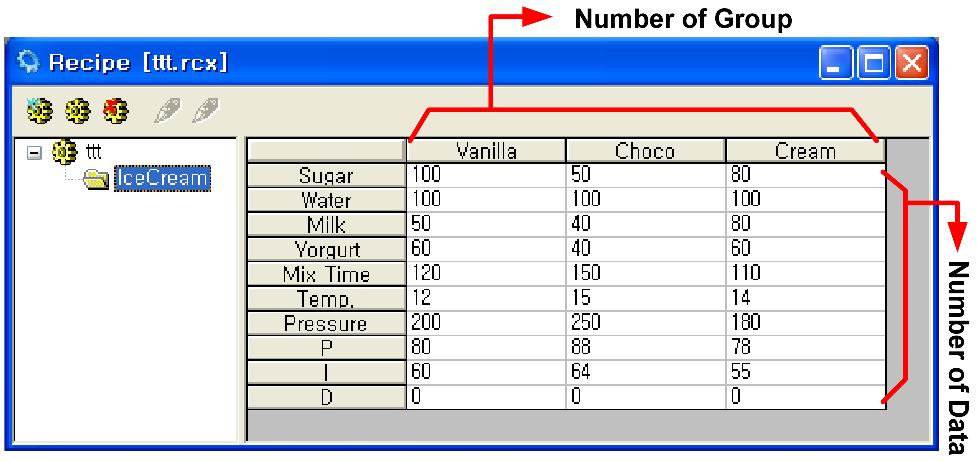
Word	RECIPE_WORD
Bit	RECIPE_HAND
Group Number	RECIPE_GRP

Post Process For Uploaded Data

☑ Automatically File Store On PLC Managed Upload

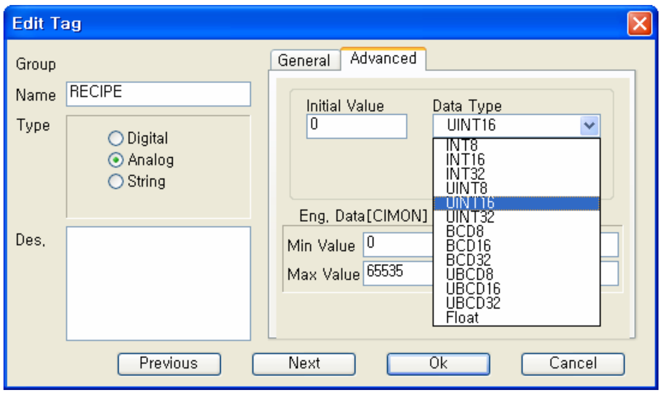
(1) Basic Properties

- **Model Name** : Every recipe model has its own name. This name will be used in the functions for recipe operation.
- **Number of Group** : One or more recipe group data can be predefined in a recipe configuration file. 'Number of Group' field gets the total number of predefined group data. These group data can be read or written by RcpFileRead() or RcpFileStore(). Each group is distinguished by the index number in those functions. According to the following example picture, index 0 is the Vanilla, and index 1 is Choco and so on.



	Vanilla	Choco	Cream
Sugar	100	50	80
Water	100	100	100
Milk	50	40	80
Yorgurt	60	40	60
Mix Time	120	150	110
Temp.	12	15	14
Pressure	200	250	180
P	80	88	78
I	60	64	55
D	0	0	0

- **Recipe Area Start Position** : One of analog TAG in database can be assigned to this field. The device address of the TAG must be designated with the first address of the recipe data memory area of a device. This memory area must be continuous. The analog TAG of this field specifies not only the first address but also specifies the type of data. XPanel supports WORD (16 bits) and DWORD (32 bits) type of data for recipe. Therefore, the data type of an analog TAG for this field should be one of INT16, UINT16, INT32 or UINT32. Following picture shows the property page of analog TAG.

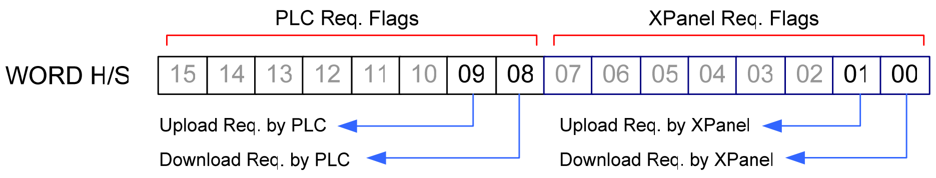


- **Number of Data** : This field gets the total number of recipe group data items. All groups in a model have the same number of data. The exact size of a recipe group data can be calculated from this value and the data type described in the previous field (Recipe Area Start Position).

(2) Handshake with the Device

XPanel does handshake with a device while up/download a recipe data via communication. The handshake is progressed on the basis of device memories. These handshake memories must be declared as TAGs in the database of XPanel.

- **Word Handshake**



Word handshake is a group of up/download request flags. It must be declared as an analog TAG with 16 bits property. This word is divided into two different areas. High nibble of the word is the flags for request from device (for example a PLC), and low nibble of word is the flags for request from XPanel.

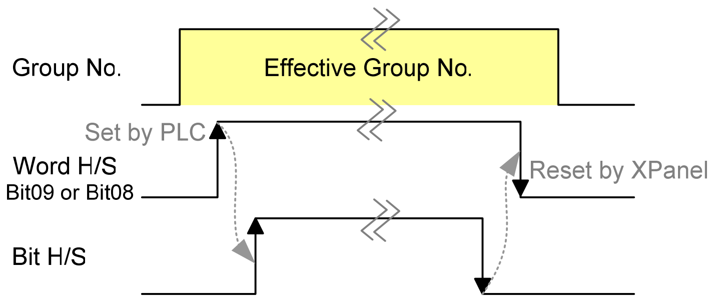
- **Bit Handshake**

The bit handshake must be declared as a digital TAG in database. It is set or reset by XPanel. This flag indicates that the recipe data transfer is under the progressing. Therefore, by this flag, the device can recognize the fact that its recipe data are ready for use or not.

- **Group Number**

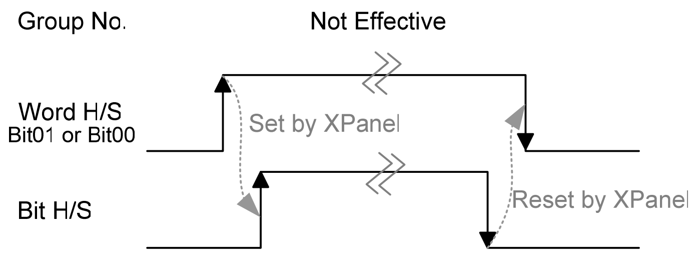
This field retains the group number for upload or download. And it is effective only when the recipe data transfer is issued by device (PLC) side. The device can choose a group in XPanel configuration file which is wanted to be up/downloaded. The group number can be a number between 0 to 1023.

Following time chart shows the status of handshake signals during recipe data transfer. It is an example of up/download issued by PLC side. For transfer a new recipe data from XPanel to device (PLC), the group number should be decided foremost and written to the memory of group number. And next, the device signals the bit08 (or bit09) flag of word handshake. Immediately after detection of that signal, XPanel sets the bit handshake and starts to transfer the recipe group data which is designated by the group number field. After the successful transfer, XPanel resets the bit handshake signal and clears the word handshake also. The device can recognize the completion of transfer through those handshake signals.



PLC Controlled Up/Down-Load

Following time chart shows an example of recipe data transfer issued by XPanel side. All the handshake signals are controlled by XPanel. The group number field has no meaning at XPanel issued transfer. Since the group number was designated in the parameter of functions already (**RcpUpload()**, **RcpDownload()**) or does not need the group number because data are not for file access (**RcpMemUp()**, **RcpMemDown()**). The device can recognize the progress of recipe data transfer with word and bit handshake signals.



XPanel Controlled Up/Down-Load

(3) Post Process for Uploaded Data

- **Automatically File Store On PLC Managed Upload**

  This option is effective only for the uploaded data by device side. As explained in previous section, the device (PLC) can control the uploading process by using handshake signal (bit9 of word handshake) and a group number.

  If this option is checked, XPanel will store the received recipe group data to the configuration file automatically at the end of upload process. At that moment, the group number from device will be used to determine which group in a file should be replaced with the newly received data. Stored group data will be retained permanently in flash memory even while XPanel is powered off. If this option is not checked, the uploaded data will be stored only in RAM.

3. Recipe Operator Interface

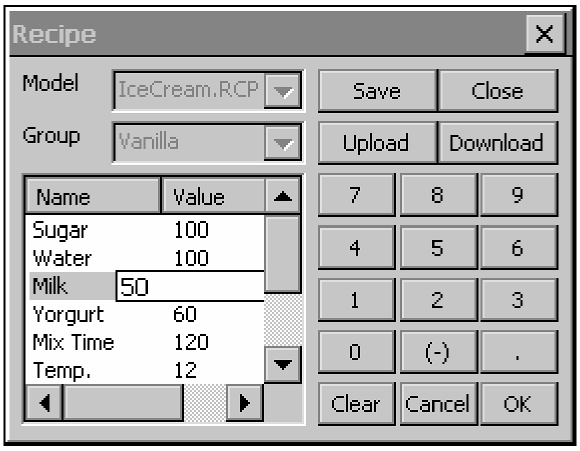
(1) Recipe Functions

XPanel provides 9 functions for recipe operation. Those functions can be used as command of touch operation as well as script. Following table shows a list of recipe functions. (Please refer to the '**Script'** section for more information)

Function	Description
RcpDownload(S1, R2)	Read a group of recipe data from file, and download it to device
RcpUpload(S1, R2)	Upload the specified recipe group data from device, and store it to file.
RcpStop(S1)	Stop and terminate the processing job of specified recipe model.
RcpFileStore(S1, R2)	Stores the recipe data to the file.
RcpFileRead(S1, R2)	Read the recipe data from file.
RcpMemDown(S1)	Download the current recipe data to device.
RcpMemUp(S1)	Upload the recipe data from device.
RcpCsvRd(S1, S2, R3)	Read the recipe data from CSV formatted file.
RcpConfig()	Pops up a dialog box for recipe data manipulation.

(2) Recipe Dialog Box

RcpConfig() function pops up a dialog box on screen. This dialog box provides some tools for manipulating recipe data in memory. Each data item in a group can be modified by touch operation. Following picture shows a example of recipe dialog box.



Name	Value
Sugar	100
Water	100
Milk	50
Yorgurt	60
Mix Time	120
Temp.	12

- **Save**

  Save button stores the recipe group data to the recipe configuration file. The storage media can be the internal flash memory of XPanel or other external none-volatile memory device such as SD/MMC. It is dependent on the location of project folder which is determined on start-up.

- **Upload**

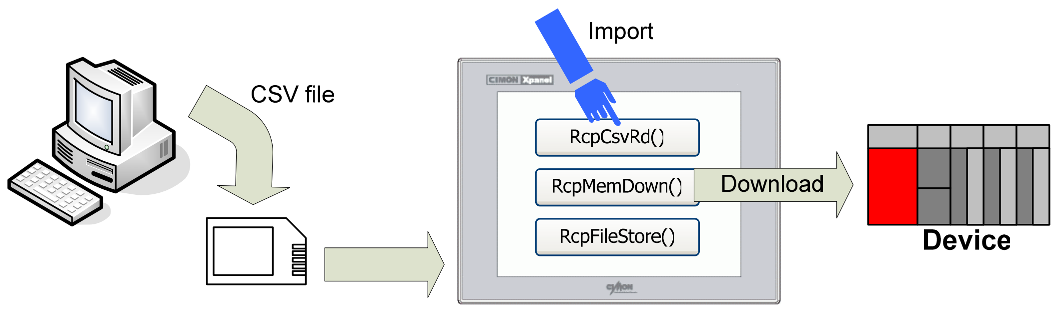
  Upload button reads the recipe data from device. Current recipe data in memory will be discarded and replaced with those of device. This button gives the same result with the function **RcpMemUp()**.

- **Download**

  Download button writes current recipe data in memory to device. This button gives the same result with the function **RcpMemDown()**.

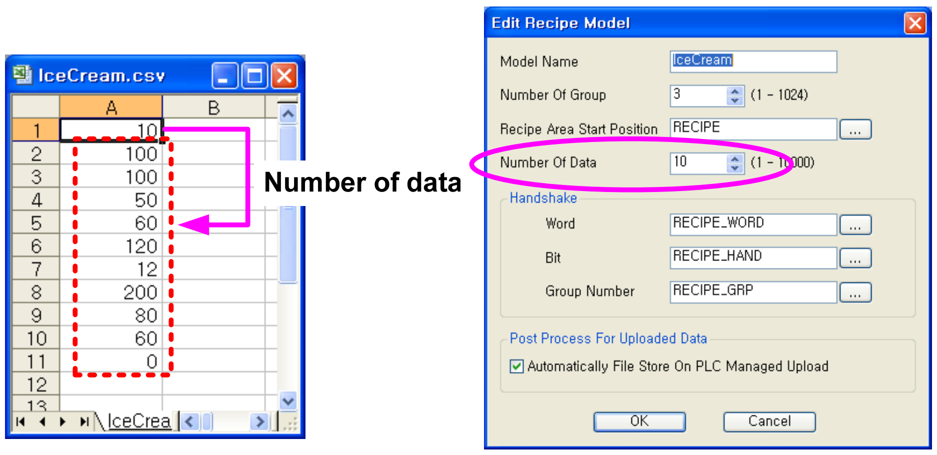
(3) CSV File Importing

XPanel provides a way to import the external recipe group data. External data must be a CSV formatted file and the SD/MMC or USB memory can be used for importing media. Following picture shows the workflow of recipe data importing.



- Writing a CSV file

  CSV formatted file can be generated by spreadsheet program of PC such as Excel. A CSV file can include one group data and the first data of this file should be the number of following recipe data. This number must same with the 'Number of Data' field of recipe configuration dialog box.



	A	B
1	10	
2	100	
3	100	
4	50	
5	60	
6	120	
7	12	
8	200	
9	80	
10	60	
11	0	
12		

Edit Recipe Model

Model Name	IceCream
Number Of Group	3 (1 ~ 1024)
Recipe Area Start Position	RECIPE
Number Of Data	10 (1 ~ 10000)

Handshake

Word	RECIPE_WORD
Bit	RECIPE_HAND
Group Number	RECIPE_GRP

Post Process For Uploaded Data

☑ Automatically File Store On PLC Managed Upload

OK Cancel

- Importing the CSV file on external memory

RcpCsvRd() function is used for importing a CSV file from external memory media. External memory can be the SD/MMC or USB memory. If there is no fault to access the CSV file in external memory, the recipe group data will be loaded to RAM of XPanel. After successful loading of recipe group data, they can be manipulated in various ways. They can be stored in configuration file and also can be downloaded to the device, Please refer to the ‘Script’ section for more information about RcpCsvRd() function.

Script

1. Script Overview

The script of Xpanel is very similar with the 'C' programming language. Variables can be used in script. And script provides function call mechanism and supports almost keywords defined in standard 'C' language such as switch-case, for, while, if-else, goto.

All user programmed scripts are executed in multi-thread environment. Each script is assigned a priority among total 10 levels, such that the execution speed of a script is dependent on this level. (most high priority = 10, most low priority = 1)

<Features>

- Uses the 'C' language style statement
- All TAGs defined in Xpanel database can be used as variables in script. It does not need additional definition, symbol or configuration. That means, values of all TAGs can be referenced or changed in every script program freely.
- Two kinds of variable types are supported, real and string type. Actual type of a variable is decided automatically during program execution. But, only one type of value can be used in a script scope. All real type data are processed in 64 bits.
- Does not distinguish capital letters. That is, 'VAR' and 'var' are the same keyword.

2. Structure of Program

[Declaration Part] Declare internal variables and input parameters

[Program Part] All program statements can be located here. Except declarations.

Comments can be located on any place in a script. A comment must be started by '//' and all strings following this symbol are treated as a comment string until the end of a line.

2.1. Declaration Part

(1) Variable Declaration

VAR *variable_name [, variable_name]*;

Declare internal variables. The scope of these variables restricted in a declared script. The initial value of declared variable is zero. If there is need to declare several variables, comma(,) can be used between each variable name. This statement can be used in multiple, until the beginning of program part.

(2) Input Parameter Declaration

PARAM 1st parameter [, 2nd parameter …];

This declaration is needed when the program can be called by another program or command statements. The order of declaration must be matched with the order of input parameters. Parameters will be initialized with input parameters from caller. If there is need to declare multiple parameters, use comma (,) between parameter names. This statement can be used in multiple, until the beginning of program part.

(3) Example : MyPgm

```
VAR a, b;
VAR c;
PARAM p1, p2;
PARAM p3;
```

Assume that the name of above example program is 'MyPgm'. This program can be called by external program or command line as following statement.

MyPgm(1, 2, 3);

Parameters p1, p2, p3 will be initialized to 1, 2, 3 each. Notice that variables a, b, c will be initialized to zero.

2.2. Program Part

In program part, all the programming statements can be used except variable(VAR) and parameter(PARAM) declarations. Calculation, storing a data in some variable, function call and many other expressions can be used in a statement. Following is a simple example of statements.

```
Tag_a = Tag_a + 1;
Tag_b = MyPgm(Tag_a, 2, 3);
```

A statement must be ended by ';' character.

All functions return a value to caller. In the above program example, the returned value will be stored in Tag_b.

2.3. Constants

Following notation is used for each type of constant.

(1) Octal constant : Number characters between 0 and 7 can be used. It must be started by zero. (ex : 01277)

(2) Decimal constant : Normal notation is used. (ex : 15, 3.14, 2.45E-12)

(3) Hexadecimal constant : It must be started with '0x'. Alphabet and number character between '0' – 'F' can be used. (ex : 0xFFFF)

(4) String constant : A string must be described between two '"'. (ex : "string variable")

(5) Predefined constant : _PI_ (pi : 3.14…….) etc. (refer to the following table)

Constant Name	Value	Remark
_PI_	3.141592….. (pi)	

LOCAL	0	Can be used as parameters for functions or commands.
SDMEM	1	
USBMEM	2	

3. Operator

3.1. Operators for calculation

The result row of following table assumes that the value of variable ‘A’ is 3 and ‘B’ is 4.

Operator	Description	Example	Result
+	Addition	A + B	7
-	Subtraction	A – B	-1
*	Multiplication	A * B	12
/	Division	A / B	0.75
%	Remainder of division(1)	A % B	3
|	Bitwise OR(1)	A | B	7
&	Bitwise AND(1)	A & B	0
^	Bitwise XOR(1)	A ^ B	7
~	Bitwise Invert(1)	~A	FFFFFFFCh

(1) Remainder of division (%) and all bitwise operators are performed with 32 bits integer data.

3.2. Logical/Comparative Operators

The result row of following table assumes that the value of variable ‘A’ is 2 (true) and ‘B’ is 0 (false). XPanel treats all values except zero as logical TRUE. Only zero is treated as logical FALSE. When the result of logical operation is TRUE, the value 1 will be returned and stored.

Operator	Description	Example	Result
&&	Logical AND	A && B	0 (false)
||	Logical OR	A || B	1 (true)
!	Logical NOT	!A	0 (false)
==	EQUAL	A == B	0 (false)
!=	NOT EQUAL	A != B	1 (true)
>	Big	A > B	1 (true)
>= or =>	Big or Equal	A >= B	1 (true)
<	Small	A < B	0 (false)
<= or =<	Small or Equal	A <= B	0 (false)

3.3. Other Operators

Operator	Description	Example	Result Value
=	Store	A = B	Store the value of B to A, and uses it as a result value.

Store (=) operator can be used in a series. For example :

A = B = C;

In this case, The value of ‘C’ will be stored to ‘A’ and ‘B’. The value of right side of the operator is store in left side and this value is treated as a return value of the operator. The internal processing order of above example is :

B <- C

A <- B

(Warning)

This feature of ‘=’ operator places a bug in program sometimes. That is the confusion with a comparative operator (==). Following example is illustrating this kind of bug. In this example, a programmer confused comparative operator(==) with store operator (=). But, XPanelDesigner cannot detect such error. The reason is, the store operator has a return value and this value can be used as a logical value in ‘if’ statement. This bug makes unexpected flow of program, and furthermore, unwanted value will be stored in ‘A’.

```
//====================================
// An example of unsolvable program mistake
//====================================
If (A=B)      // Compare values of A and B. It is miss-spelling of A==B
{
       ….
}
```

4. Statements

4.1. IF – ELSE Statement

IF – ELSE statement is one of the most frequently using logical decision making statement. It supports nested IF statement like IF - ELSE IF - ELSE IF…. There is no limit on level of nested IF. Else statement can be omitted.

```
If (A == 1)
{
        // performed when the value of A is 1
}
Else
{
        // performed when the value of A is not 1
}
```

4.2. WHILE / DO-WHILE Statements

These statements are loop statements. WHILE keyword is followed by a logical expression. During the value of this expression is TRUE, all statements in WHILE statement are performed repeatedly.

```
A = 0;
While (A < 10)
{
      // Describe the job should be performed repeatedly
      // during the value of logical expression is TRUE
      // (This example performs following statement 10 times)
      A = A + 1;
}
```

WHILE statement makes decision before performing the loop. Such that, statements in loop can never be performed.

But, DO-WHILE statement performs the loop at least one time. And then makes decision by evaluating logical expression. This is a difference between WHILE and DO-WHILE statement.

```
Do {
          // Describe the job should be performed repeatedly
          // during the value of logical expression is TRUE
          // (This loop is performed at least 1 time)
} While (A < 10);  // Semi-colon (;) must be used for marking end of statement.
```

4.3. FOR Statement

It is a little bit complex statement. But, after familiar with this statement, it will be very useful loop statement. The specialty of this statement is that all expressions for initializing, decision making and post loop processing are described in one statement line.

Following example shows the same job with the example of WHILE statement presented in former section. Compare it with WHILE statement.

```
For (A=0; A<10; A=A+1)
{
              // describe the job to repeat here
}
```

FOR keyword is followed by a pair of parenthesis. Expression for initializing, decision making and post loop processing are described in this parenthesis. Each expression must be separated by semicolons (‘;’).

FOR (*initializing exp.* **;** *decision making exp.* **;** *post processing exp.* **)**

4.4. SWITCH – CASE Statement

This statement is very useful when there are many different processing way depend on a variable or expression. Each different processing way is described by CASE keyword followed by a constant value. There is no limit on the number of CASE.

```
switch (A)
{
Case 1:
            // describe the job to perform when the value of A is 1
      Break;
Case 5:
Case 7:
            // describe the job to perform when the value of A is 5 or 7
      Break;
Default:
            // describe the job to perform when A is any other value
            // DEFAULT statements can be omitted
      Break;
}
```

CASE keyword must be followed by a constant value and terminated by a colon (:). A TAG, variable or expression cannot be used in CASE statement.

DEFAULT statement can be used only once in a SWITCH statement. It does not need a following constant, and this statement will be performed when there is no matching case. DEFAULT statement can be omitted if needless.

BREAK keyword in CASE/DEFAULT statement plays role of terminating SWITCH statement. If there is no BREAK keyword in the progress of program execution until reaches the other CASE/DEFAULT keyword, the program will continue to execute the statements in the other CASE/DEFAULT scope. This feature of SWITCH statement is helpful for short and simple programming, only if it is used intentionally.

4.5. GOTO Statement

This statement controls the flow of program execution directly. This statement needs a matching index in a program scope. Index is a unique name of a jump destination in a program scope, and it is started with '@' character.

```
VAR A;

    A = 0;
@JumpHere                       // destination of jump
    If (A < 10)
    {
        A = A + 1;
        Goto JumpHere;          // change the flow of program
    }
```

This example program processes the same job with the previous examples of FOR and WHILE statement section. The name of index ('JumpHere' in the above example) can not be duplicated with variables, tags and program names.

4.6. CONTINUE Keyword

Continue keyword is very useful in loop. When a CONTINUE keyword is met in the progress of a program, the flow of the program will be moved to the first statement in the loop. This keyword is effective in the loop of WHILE, DO-WHILE and FOR statement. It is especially useful when there are many decision making in the loop such as IF-ELSE.

Following two program examples process the same job.

```
While (A<10)
{
    A = A + 1;
    If (A < 5)
    {
        // Job to do when A is smaller than 5
    }
    Else
    {
        // Job to do when A is larger than or equal to 5
    }
}
```

```
While (A<10)
{
        A = A + 1;
        If (A < 5)
        {
                // Job to do when A is smaller than 5
                Continue;
        }
                // Job to do when A is larger than or equal to 5
}
```

4.7. RETURN Keyword

Every program returns a result value to caller. RETURN keyword is used for designating a value to return and terminating the program. Following example program processes three input parameters and returns the sum of them to caller.

```
PARAM p1, p2, p3;

Return p1 + p2 + p3;
```

Assuming that the name of above program is 'MyPgm', it can be called from another program as following program line.

```
RtnValue = MyPgm(1, 2, 3);
```

After 'MyPgm' call, a value of 6 will be stored in 'RtnValue'.

4.8. RUNSCRIPT Keyword

When a program calls another program, the caller program will be stalled until when the called program terminates.

RUNSCRIPT keyword provides another way of invoking other program. It does not make stall the caller program. After invoke another program, the caller program continues it's own job. As a result, both programs will run in parallel and this is the feature of multi-threading.

Following example shows the multi-threading feature by using RUNSCRIPT keyword. It is assuming that the name of another program is ‘NewThread’.

```
If(A == 0)
      RunScript NewThread();          // Invoke another program (NewThread)
A = A + 1;                            // Continues next job without stalling.
```

5. Internal Functions

5.1. Function List

- R1, R2, …. : Real type input parameters
- S1, S2, …. : String type input parameters

(1) Trigonometric Functions

Function	Description	Return
Sin(R1)	Sine	Real
Cos(R1)	Cosine	Real
Tan(R1)	Tangent	Real
Asin(R1)	Arc Sine	Real
Acos(R1)	Arc Cosine	Real
Atan(R1)	Arc Tangent	Real
Sinh(R1)	Hyperbolic Sine	Real
Cosh(R1)	Hyperbolic Cosine	Real
Tanh(R1)	Hyperbolic Tangent	Real
Atan2(R1, R2)	Arc Tangent, returns the same result with Atan(R1/R2)	Real

(2) Other Mathematic Functions

Function	Description	Return
Log(R1)	Calculates logarithm	Real
Log10(R1)	Calculates logarithm	Real
Ceil(R1)	Calculates the ceiling of a value	Real
Floor(R1)	Calculates the floor of a value	Real
Abs(R1)	Calculates the absolute value	Real
Sqrt(R1)	Calculates the square value	Real
Rand()	Generates a pseudorandom number	Real
Fmod(R1, R2)	Calculates the floating-point remainder (R1/R2)	Real

(3) Special Functions

Function	Description	Return
PageOpen(S1)	Opens a new page named as S1	
PageClose(S1)	Closes an opened page	
FrameOpen(S1)	Opens a new frame named as S1	
StringTable(R1, R2)	Get a string from string table (R1=Group No., R2=String No.)	String
GetTime(R1)	Get a clock data (R1 : type of data)	Real

TimeStr(R1, S2)	Assemble a string of time R1 : clock value from GetTime(0) S2 : Format string	String
RunApp(S1, S2)	Executes an external program S1 : Program name S2 : Program input parameter	
MakeCsv(S1, R2)	Makes a CSV formatted log file in SD memory S1 : Name of log model R2 : Number of block (0=current logging block)	
MakeCsvUsb(S1, R2)	Makes a CSV formatted log file in USB memory S1 : Name of log model R2 : Number of block (0=current logging block)	
DataLog(S1, R2)	Start / Stop the data logging S1 : Name of log model R2 : 0=Stop, 1=Start	
RcpDownload(S1, R2)	Read a group of recipe data from file, and download it to device S1 : Name of recipe model R2 : Group number	
RcpUpload(S1, R2)	Upload the specified recipe group data from device, and store it to file. S1 : Name of recipe model R2 : Group number	
RcpStop(S1)	Stop and terminate the processing job of specified recipe model. S1 : Name of recipe model	
RcpFileStore(S1, R2)	Stores the recipe data to the file. S1 : Name of recipe model R2 : Group number	
RcpFileRead(S1, R2)	Read the recipe data from file. S1 : Name of recipe model R2 : Group number	
RcpMemDown(S1)	Download the current recipe data to device. S1 : Name of recipe model	
RcpMemUp(S1)	Upload the recipe data from device. S1 : Name of recipe model	
RcpCsvRd(S1, S2, R3)	Read the recipe data from CSV formatted file.	

	S1 : Name of recipe model S2 : File name R3 : File location (0=local, 1=SD/MMC, 2=USB)	
RcpConfig()	Pops up a dialog box for recipe data manipulation.	
TrendCsvWr(S1, R2)	Make a CSV formatted file from trend data. S1 : Trend name R2 : File location (0=local, 1=SD/MMC, 2=USB)	
ScrCapture(S1, R2)	Make a BMP formatted file of current screen S1 : BMP file name. R2 : File location (0=local, 1=SD/MMC, 2=USB)	
Sleep(R1)	Put delay on the script program R1 : delay time in milli-second	

5.2. Function Details

PageOpen(S1)	**Open a new page**

[S1] new page name (without extension)

Opens a new screen page. If the new page has no special style, then previous page or frame will be closed automatically. Otherwise, according to the style of the new page, following result can be shown.

(1) A pop-up style

If a new page has a style of pop-up, current page does not affected by this new page. The new page pops up over current page.

(2) A page for frame

This kind of page has a position number in a frame. For open this page, the frame has to be opened in advance. The old page which has the same position number in the frame will be replaced by this page automatically.

PageClose(S1)	**Closes an opened page**

[S1] page name (without extension)

Closes an opened screen page. If the page of S1 parameter was not opened, the current screen will not be affected by this function.

FrameOpen(S1)	**Open a new frame**

[S1] new frame name (without extension)

Opens a new screen frame. Previous page or frame will be closed automatically. The default pages for frame will be opened automatically.

StringTable(R1, R2)	Get a string data from string table

[R1] String group number

[R2] String number in a group

Gets a string data from the string table. The string table must have been configured by 'String Editor'. Assign the group number of a string table to the R1, and the string number of a string in that group to the R2 parameter. Returned data is a string type value, and it can be used for string tag, variable or other function's parameter.

[Example Program]

```
StrTag = StringTable(1, 14);
```

[Result]

StrTag : "A string that is defined as 14 in group #1"

GetTime(R1)	Get the current clock data

[R1] type of clock data

0 : The number of seconds after 1st. January 1970 UTC

1 : the number of year (1970..)

2 : the number of month (1..12)

3 : the number of day (1..31)

4 : the number of hour (0..23)

5 : the number of minute (0..59)

6 : the number of second (0..59)

7 : the number of day of week (1..7 : 1=Sunday, 2=Monday…, 7=Saturday)

8 : the number of today's minute count (0..1439 : 0=midnight)

9 : the number of today's second count (0..86399, 0=midnight)

TimeStr(R1, S2)	Make a string of clock

[R1] Clock data (It can be acquired by GetTime(0) function call)

The number of seconds after 1st. January 1970 UTC

[S2] Format string (can include following symbols)

%A : Full name of weekday (ex : Sunday)

%a : Abbreviation of weekday (ex : Sun)

%B : Full name of month (ex : January)

%b : Abbreviation of month (ex : Jan)

%d : date (1..31)

%H : Hour in 24-hour format (0..23)

%I : Hour in 12-hour format (1..12)

%m : Month (1..12)

%M : Minute (0..59)

%p : A.M. / P.M. indicator for 12-hour format (A.M., P.M.)

%S : Second (0..59)

%y : Number of year without century (00..99)

%Y : Number of year with century (ex : 2006)

[Example Program]

CurTime = GetTime(0);

StrTag = TimeStr(CurTime, “%Y/%m/%d %H:%M:%S”) ;

[Result]

StrTag : “2006/10/12 23:59:59”

RunApp(S1, S2)	**Executes an external application program.**

[S1] Application program name with path and extension

[S2] Input parameter string for the application program.

Executes an external application program. This program can be not only system included program, also user programmed program.

[Example Program]

RunApp(“Ping.Exe”, “-t 100.100.100.1”);

[Result]

Ping test screen will be popped up when this command or program is issued.

MakeCsv(S1, R2)	**Makes a CSV formatted log file in SD memory**

[S1] Log model name

[R2] Block number of designated log model.

Converts the designated block file to a CSV formatted file and stores it on the SD/MMC memory. A SD/MMC memory has to be plugged before the issuing of this function. A CSV file will be written in the folder ‘XPanel’ of the SD/MMC memory card. The CSV file name is assembled with the date and time log started, and it’s from original block file. (YYMMDD HHmmSS.CSV)

When the R2 parameter is zero, this function will try to convert the current log proceeding (uncompleted) block file. If the logging task of designated model is not running at the moment of the function call, no CSV file will be created.

The parameter R2 can be designated up to the ‘Maximum Block No.’ which should have been set in the model configuration dialog box. When the R2 parameter is 1, this function

will try to convert the most recently log completed block file.
[Example Program] MakeCsv(“ModelName”, 1); [Result] A CSV file will be created from the most recently log completed block file.

MakeCsvUsb(S1, R2)	**Makes a CSV formatted log file in USB memory**
[S1] Log model name [R2] Block number of designated log model. Converts the designated block file to a CSV formatted file and stores it on the USB memory. An USB memory has to be plugged before the issuing of this function. A CSV file will be written in the folder ‘XPanel’ of the USB memory card. The CSV file name is assembled with the date and time log started, and it’s from original block file. (YYMMDD HHmmSS.CSV) When the R2 parameter is zero, this function will try to convert the current log proceeding (uncompleted) block file. If the logging task of designated model is not running at the moment of the function call, no CSV file will be created. The parameter R2 can be designated up to the ‘Maximum Block No.’ which should have been set in the model configuration dialog box. When the R2 parameter is 1, this function will try to convert the most recently log completed block file.	
[Example Program] MakeCsvUsb(“ModelName”, 1); [Result] A CSV file will be created from the most recently log completed block file.	

DataLog(S1, R2)	**Data logging start / stop.**
[S1] Log model name [R2] 0 or 1 (0 : STOP, 1 : START) This function is effective only on the log model of external call start type. Models with other start type such as periodic, trigger tag, enable tag or on time are not affected by this function. If the R2 parameter is 0, the designated logging task will be terminated. Otherwise, the logging task will be invoked by this function. This function treats the non-zero value of R2 as 1.	
[Example Program] DataLog(“ModelName”, 1);	

[Result]

Data logging task will be invoked.

RcpDownload(S1, R2)	Reads a recipe data group from file, and downloads to device
[S1] Recipe model name [R2] Group number Reads a group data from the recipe configuration file, and download them to the device. After this function, the group data in XPanel memory will be changed to the read group data. The group data must have been configured in XPanel Designer before this function call.	
[Example Program] RcpDownload(“ModelName”, 1); [Result] Reads the #1 group data from “ModelName” recipe file, and download them to the device.	

RcpUpload(S1, R2)	Uploads a recipe group data from device, and stores them to the recipe configuration file.
[S1] Recipe model name [R2] Group number Reads a group data from the device through communication, and stores them to the recipe configuration file. After this function, the group data in XPanel memory will be changed to the read group data. The designated recipe group must have been declared in XPanel Designer before this function call.	
[Example Program] RcpUpload(“ModelName”, 2); [Result] Uploads the recipe data from the device, and stores them to the group #2 of “ModelName” recipe file.	

RcpStop(S1)	Stop and terminate the recipe operations.
[S1] Recipe model name This function terminates any job related with designated name of recipe model. For example, if download process is proceeding through communication when this function is issued, then the communication will be stopped and terminated. Be careful to use this function, because there is a possibility of storing unwanted data group in file or device. Use this function only when there is a unrecoverable network or device breakdown.	

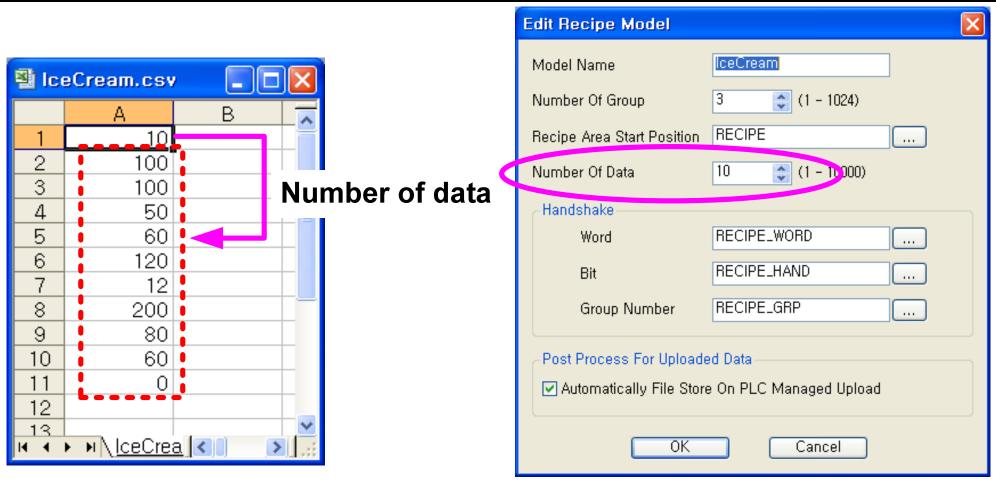
RcpFileStore(S1, R2)	Stores the current recipe data in memory to the file.
[S1] Recipe model name [R2] Group number This function stores a group data in memory of XPanel to the configuration file. The designated recipe group must have been declared in XPanel Designer before this function call.	

RcpFileRead(S1, R2)	Reads a group data from the recipe configuration file.
[S1] Recipe model name [R2] Group number This function reads a group data in the configuration file and stores them to the memory of XPanel. The designated recipe group must have been configured by XPanel Designer, or stored by other way before this function call.	

RcpMemDown(S1)	Downloads a group of recipe data in memory to the device.
[S1] Recipe model name This function downloads a group data in memory of XPanel to the device. The group data must have been loaded in memory by other way before this function issue.	

RcpMemUp(S1)	Uploads a group of recipe data to memory from the device.
[S1] Recipe model name This function uploads a group data in device to the memory of XPanel. The uploaded group data can be processed by other functions or recipe tool window.	

RcpCsvRd(S1, S2, R3)	Reads a group of recipe data from CSV formatted file.
[S1] Recipe model name [S2] CSV file name (without extension and path) [R3] CSV file location code (0 : local, 1 : SD/MMC, 2 : USB) This function reads a group of data from CSV formatted file and stores them to the memory of XPanel. The CSV formatted file must have information about the number of data it has. This number must be located in the top most of data list, and the number must be matched with the one that assigned in the recipe model setup dialog box.	



	A	B
1	10	
2	100	
3	100	
4	50	
5	60	
6	120	
7	12	
8	200	
9	80	
10	60	
11	0	
12		
13		

Model Name: IceCream
Number Of Group: 3 (1 - 1024)
Recipe Area Start Position: RECIPE
Number Of Data: 10 (1 - 10000)
Handshake
Word: RECIPE_WORD
Bit: RECIPE_HAND
Group Number: RECIPE_GRP
Post Process For Uploaded Data
[✓] Automatically File Store On PLC Managed Upload

The second parameter S2 is a name of CSV file to read. There is no need to specify the extension. But, the path of the CSV file location must be root folder of each storage media. The third parameter R3 indicates the storage media where the CSV file was stored. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R3 instead of number.

Media	Number	Symbol	CSV file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

RcpCsvRd(“ModelName”, “RcpFileName”, _SDMEM_);

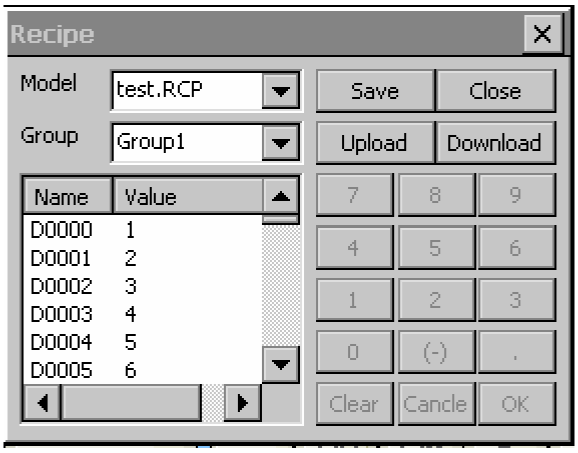
[Result]

Reads a CSV formatted file named as “RcpFileName.CSV” in SD/MMC memory card, and stores the data in memory of XPanel.

RcpConfig()	Pops up the recipe data manipulation tool window.

No Parameter

This function pops up a tool window for recipe. This tool window shows the current recipe data in memory. And it provides some tools for editing individual datum in a group, changing current group, saving the current data to file and up/downloading the recipe data from/to the device. (Please refer to the ‘**Recipe**’ section for more information)



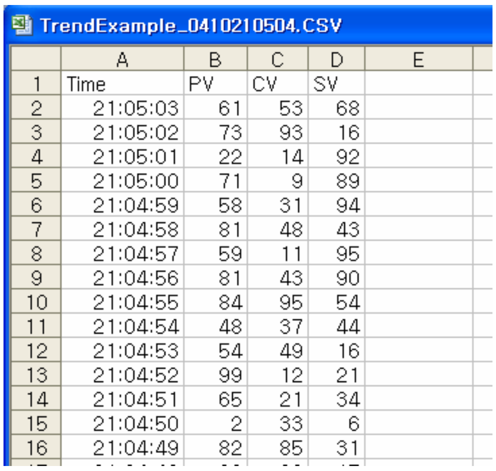
Name	Value
D0000	1
D0001	2
D0002	3
D0003	4
D0004	5
D0005	6

TrendCsvWr(S1, R2)	Makes and stores a CSV formatted file from trend.

[S1] Trend name

[R2] CSV file location code (0 : local, 1 : SD/MMC, 2 : USB)

This function makes a CSV formatted file from a trend object. Output CSV file can be stored in not only local flash disk, but external memory card. Next screen capture shows an example of output CSV file.


TrendExample_0410210504.CSV

	A	B	C	D	E
1	Time	PV	CV	SV	
2	21:05:03	61	53	68	
3	21:05:02	73	93	16	
4	21:05:01	22	14	92	
5	21:05:00	71	9	89	
6	21:04:59	58	31	94	
7	21:04:58	81	48	43	
8	21:04:57	59	11	95	
9	21:04:56	81	43	90	
10	21:04:55	84	95	54	
11	21:04:54	48	37	44	
12	21:04:53	54	49	16	
13	21:04:52	99	12	21	
14	21:04:51	65	21	34	
15	21:04:50	2	33	6	
16	21:04:49	82	85	31	

The name of generated CSV file has the format of "***trendname*_MMDDHHmmSS.CSV**". This name is generated from trend name, date and time.

The second parameter R2 indicates the storage media where the CSV file will be stored. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R2 instead of number.

Media	Number	Symbol	CSV file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

TrendCsvWr("TrendExample", _SDMEM_);

[Result]

Makes a CSV formatted file on SD/MMC cards. The data will be taken from the "TrendExample" trend data.

ScrCapture(S1, R2)	Captures current screen and makes a BMP format image file.

[S1] Seed for making BMP file name

[R2] BMP file location code (0 : local, 1 : SD/MMC, 2 : USB)

This function captures current screen display and makes a BMP formatted image file. The name of BMP file will be made with the seed string of S1 parameter and time. File name format is "***seedname*_HHmmSS.BMP**".

The second parameter R2 indicates the storage media for generated BMP file. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R2 instead of number.

Media	Number	Symbol	BMP file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

ScrCapture("Mybmp", _SDMEM_);

[Result]

Captures current screen and makes a BMP formatted file on SD/MMC cards.

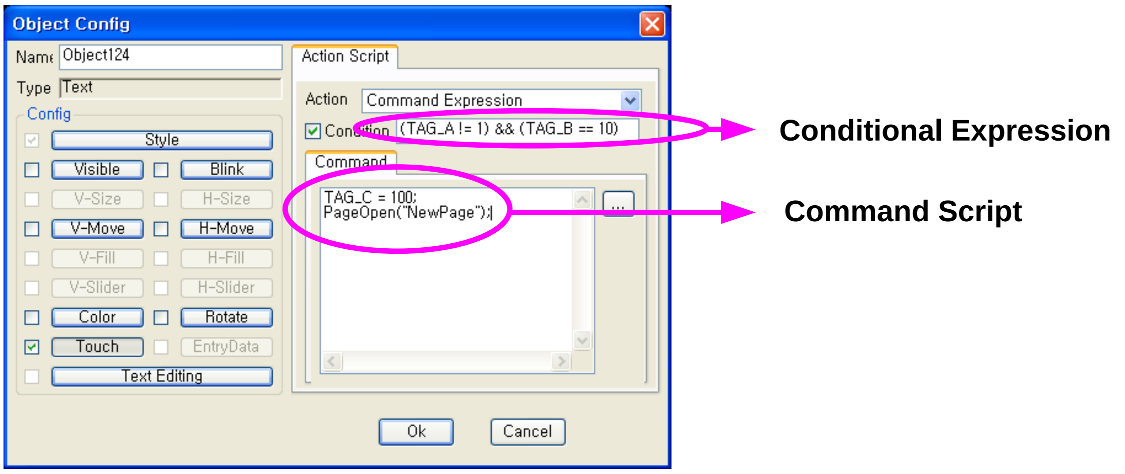
Sleep(R1)	Put some delay to the current script program

[R1] Delay time in milli-seconds

This function suspends the execution of the current script program for a specified interval (R1). The interval must be specified in milli-seconds unit.

6. Scripts for Command and Condition

In some application, there is a need to use a simple script on graphic object configuration. For example, to assign a touch operation on an object, one or several command lines must be written in case of “Command Expression” action type (see an example configuration dialog box below). And further more, for making it conditional touch operation, a conditional expression must be described. All these conditional expression and command script have the same rules with script programming.


Condition: (TAG_A != 1) && (TAG_B == 10)

```
TAG_C = 100;
PageOpen("NewPage");
```

6.1. Conditional Expression

The conditional expression always returns only a logical value. A logical value can be one of two values, TRUE or FALSE. XPanel uses the number ‘0’ as the value of logical FALSE, and ‘1’ as the value of logical TRUE. A logical expression needs some special operators. The logical operators in XPanel are shown in as following table. For more precise information, refer to the section 3.2.

Logical Operator	Description	Example
&&	Logical AND	A && B
||	Logical OR	A || B
!	Logical NOT	!A
Comparative Operator	**Description**	**Example**
==	EQUAL	A == B
!=	NOT EQUAL	A != B
>	Big	A > B
>= or =>	Big or Equal	A >= B
<	Small	A < B
<= or =<	Small or Equal	A <= B

All logical operators get one or two values for input (operands) and return a logical output value. According to the above examples, ‘A’ and ‘B’ are those inputs. The value for logical arithmetic operand is not restricted to only two logical values (TRUE or FALSE). The input value can be any number. But, only zero is treated as logical FALSE. All the other values are treated as TRUE.

Comparative operators get two real values for operands and return a logical output value. Those logical output values can be used in other logical expression.

Here is some examples of conditional expression that can be used in object configuration dialog box.

Conditional Expression Example	Remark
(Tag_a != 1) && (Tag_b == 10)	If the value of Tag_a is not equal to 1 and the value of Tag_b is equal to 10, then the expression returns TRUE. Otherwise FALSE.
Sin(Tag_a) == 1	If the sine of Tag_a is 1, then the expression will return TRUE, otherwise FALSE.
Tag_a < 100	If the value of Tag_a is smaller than 100, then the expression will return TRUE, otherwise FALSE.
(Tag_a + Tag_b) < (Tag_a + Tag_c)	If the sum of Tag_a and Tag_b is smaller than the sum of Tag_a and Tag_c, then the expression will return TRUE, otherwise FALSE.

6.2. Command Script

The command in XPanel is not restricted to use only one instruction. A long list of instructions can construct a command. That is, the command is a special kind of script program. Within a command script, all kinds of keywords and functions can be used.

But only one thing, the priority of command script is different with the normal script. When a command script is issued by touch operation or any other method, it will be executed with top-most priority. For that reason, if a command script has a time consuming loop inside, all the other process will be halted until the end of command script. So, especially for calling other function within a command script, the programmer has to keep in mind that the priority. If the function has a loop, ‘**RunScript**’ keyword must be used. This keyword is used for parallel execution. The calling command script does not wait the termination of called function.

Assume that a programmer wants to call a function named as ‘MyLoop’, and this function has a time consuming loop. Then, following instruction must be used, instead of calling directly.

RunScript MyLoop();

Following table shows some typical command script examples.

Command Script Example	Remark

Tag_a = 100;
PageOpen("NewPage");
	After changing the value of Tag_a to 100, open a page named as "NewPage".

Tag_b = Tag_b + 1;	Increase the value of Tag_b by 1.

Tag_a = 100;
Tag_b = 1;
RunScript MyLoop();
MakeCsv("LogModel", 1);
	After changing the values of Tag_a and Tag_b, invoke MyLoop() script function. After that, do not wait the termination of MyLoop(), and create a CSV formatted file of the 'LogModel' on SD memory card.

Script

1. Script Overview

The script of Xpanel is very similar with the ‘C’ programming language. Variables can be used in script. And script provides function call mechanism and supports almost keywords defined in standard ‘C’ language such as switch-case, for, while, if-else, goto.

All user programmed scripts are executed in multi-thread environment. Each script is assigned a priority among total 10 levels, such that the execution speed of a script is dependent on this level. (most high priority = 10, most low priority = 1)

<Features>

- Uses the ‘C’ language style statement
- All TAGs defined in Xpanel database can be used as variables in script. It does not need additional definition, symbol or configuration. That means, values of all TAGs can be referenced or changed in every script program freely.
- Two kinds of variable types are supported, real and string type. Actual type of a variable is decided automatically during program execution. But, only one type of value can be used in a script scope. All real type data are processed in 64 bits.
- Does not distinguish capital letters. That is, ‘VAR’ and ‘var’ are the same keyword.

2. Structure of Program

[Declaration Part] Declare internal variables and input parameters

[Program Part] All program statements can be located here. Except declarations.

Comments can be located on any place in a script. A comment must be started by '//' and all strings following this symbol are treated as a comment string until the end of a line.

2.1. Declaration Part

(1) Variable Declaration

VAR *variable_name [, variable_name]*;

Declare internal variables. The scope of these variables restricted in a declared script. The initial value of declared variable is zero. If there is need to declare several variables, comma(,) can be used between each variable name. This statement can be used in multiple, until the beginning of program part.

(2) Input Parameter Declaration

PARAM 1stparameter [, 2ndparameter …];

This declaration is needed when the program can be called by another program or command statements. The order of declaration must be matched with the order of input parameters. Parameters will be initialized with input parameters from caller. If there is need to declare multiple parameters, use comma (,) between parameter names. This statement can be used in multiple, until the beginning of program part.

(3) Example : MyPgm

```
VAR a, b;
VAR c;
PARAM p1, p2;
PARAM p3;
```

Assume that the name of above example program is ‘MyPgm’. This program can be called by external program or command line as following statement.

MyPgm(1, 2, 3);

Parameters p1, p2, p3 will be initialized to 1, 2, 3 each. Notice that variables a, b, c will be initialized to zero.

2.2. Program Part

In program part, all the programming statements can be used except variable(VAR) and parameter(PARAM) declarations. Calculation, storing a data in some variable, function call and many other expressions can be used in a statement. Following is a simple example of statements.

```
Tag_a = Tag_a + 1;
Tag_b = MyPgm(Tag_a, 2, 3);
```

A statement must be ended by ‘;’ character.

All functions return a value to caller. In the above program example, the returned value will be stored in Tag_b.

2.3. Constants

Following notation is used for each type of constant.

(1) Octal constant : Number characters between 0 and 7 can be used. It must be started by zero. (ex : 01277)

(2) Decimal constant : Normal notation is used. (ex : 15, 3.14, 2.45E-12)

(3) Hexadecimal constant : It must be started with ‘0x’. Alphabet and number character between ‘0’ – ‘F’ can be used. (ex : 0xFFFF)

(4) String constant : A string must be described between two ““’. (ex : “string variable”)

(5) Predefined constant : _PI_ (pi : 3.14…….) etc. (refer to the following table)

Constant Name	Value/Usage	Remark
_PI_	3.141592….. (pi)	

LOCAL	0, Local flash memory	Can be used as parameters for functions or commands.
SDMEM	1, SD/MMC memory	
USBMEM	2, USB memory	
COM232	0, COM1, RS232C mode	COM port number. See the serial communication functions.
COM422	1, COM1, RS422 mode	
COM485	2, COM1, RS485 mode	
COMAUX	3, COM2 (RS232C only)	
BPS300	300 bps	Baudrate code. See the OpenPort() function.
BPS600	600 bps	
BPS1200	1200 bps	
BPS2400	2400 bps	
BPS4800	4800 bps	
BPS9600	9600 bps	
BPS19200	19200 bps	
BPS38400	38400 bps	
BPS56000	56000 bps	
BPS57600	57600 bps	
BPS115200	115200 bps	
BPS128000	128000 bps	
BPS256000	256000 bps	
_PARITY_NONE_	No parity	Parity code. See the OpenPort() function.
_PARITY_EVEN_	Even parity	
_PARITY_ODD_	Odd parity	
_PARITY_MARK_	Mark parity	
_PARITY_SPACE_	Space parity	
_STOPBIT_ONE_	1 stop bit	Stopbit code. See the OpenPort() function.
_STOPBIT_TWO_	2 stop bits	
_STOPBIT_ONE5_	1.5 stop bit	

3. Operator

3.1. Operators for calculation

The result row of following table assumes that the value of variable ‘A’ is 3 and ‘B’ is 4.

Operator	Description	Example	Result
+	Addition	A + B	7
-	Subtraction	A – B	-1
*	Multiplication	A * B	12
/	Division	A / B	0.75
%	Remainder of division(1)	A % B	3
|	Bitwise OR(1)	A | B	7
&	Bitwise AND(1)	A & B	0
^	Bitwise XOR(1)	A ^ B	7
~	Bitwise Invert(1)	~A	0xFFFFFFFC

(1) Remainder of division (%) and all bitwise operators are performed with 32 bits integer data.

3.2. Logical/Comparative Operators

The result row of following table assumes that the value of variable ‘A’ is 2 (true) and ‘B’ is 0 (false). XPanel treats all values except zero as logical TRUE. Only zero is treated as logical FALSE. When the result of logical operation is TRUE, the value 1 will be returned and stored.

Operator	Description	Example	Result
&&	Logical AND	A && B	0 (false)
||	Logical OR	A || B	1 (true)
!	Logical NOT	!A	0 (false)
==	EQUAL	A == B	0 (false)
!=	NOT EQUAL	A != B	1 (true)
>	Big	A > B	1 (true)
>= or =>	Big or Equal	A >= B	1 (true)
<	Small	A < B	0 (false)
<= or =<	Small or Equal	A <= B	0 (false)

3.3. Other Operators

Operator	Description	Example	Result Value
=	Store	A = B	Store the value of B to A, and uses it as a result value.

Store (=) operator can be used in a series. For example :

A = B = C;

In this case, The value of 'C' will be stored to 'A' and 'B'. The value of right side of the operator is store in left side and this value is treated as a return value of the operator. The internal processing order of above example is :

B <- C

A <- B

(Warning)

This feature of '=' operator places a bug in program sometimes. That is the confusion with a comparative operator (==). Following example is illustrating this kind of bug. In this example, a programmer confused comparative operator(==) with store operator (=). But, XPanelDesigner cannot detect such error. The reason is, the store operator has a return value and this value can be used as a logical value in 'if' statement. This bug makes unexpected flow of program, and furthermore, unwanted value will be stored in 'A'.

```
//===================================
// An example of unsolvable program mistake
//===================================
If (A=B)      // Compare values of A and B. It is miss-spelling of A==B
{
      ….
}
```

4. Statements

4.1. IF – ELSE Statement

IF – ELSE statement is one of the most frequently using logical decision making statement. It supports nested IF statement like IF - ELSE IF - ELSE IF…. There is no limit on level of nested IF. Else statement can be omitted.

```
If (A == 1)
{
        // performed when the value of A is 1
}
Else
{
        // performed when the value of A is not 1
}
```

4.2. WHILE / DO-WHILE Statements

These statements are loop statements. WHILE keyword is followed by a logical expression. During the value of this expression is TRUE, all statements in WHILE statement are performed repeatedly.

```
A = 0;
While (A < 10)
{
      // Describe the job should be performed repeatedly
      // during the value of logical expression is TRUE
      // (This example performs following statement 10 times)
      A = A + 1;
}
```

WHILE statement makes decision before performing the loop. Such that, statements in loop can never be performed.

But, DO-WHILE statement performs the loop at least one time. And then makes decision by evaluating logical expression. This is a difference between WHILE and DO-WHILE statement.

```
Do {
          // Describe the job should be performed repeatedly
          // during the value of logical expression is TRUE
          // (This loop is performed at least 1 time)
} While (A < 10);  // Semi-colon (;) must be used for marking end of statement.
```

4.3. FOR Statement

It is a little bit complex statement. But, after familiar with this statement, it will be very useful loop statement. The specialty of this statement is that all expressions for initializing, decision making and post loop processing are described in one statement line.

Following example shows the same job with the example of WHILE statement presented in former section. Compare it with WHILE statement.

```
For (A=0; A<10; A=A+1)
{
              // describe the job to repeat here
}
```

FOR keyword is followed by a pair of parenthesis. Expression for initializing, decision making and post loop processing are described in this parenthesis. Each expression must be separated by semicolons (‘;’).

FOR (*initializing exp.* **;** *decision making exp.* **;** *post processing exp.* **)**

4.4. SWITCH – CASE Statement

This statement is very useful when there are many different processing way depend on a variable or expression. Each different processing way is described by CASE keyword followed by a constant value. There is no limit on the number of CASE.

```
switch (A)
{
Case 1:
            // describe the job to perform when the value of A is 1
      Break;
Case 5:
Case 7:
            // describe the job to perform when the value of A is 5 or 7
      Break;
Default:
            // describe the job to perform when A is any other value
            // DEFAULT statements can be omitted
      Break;
}
```

CASE keyword must be followed by a constant value and terminated by a colon (:). A TAG, variable or expression cannot be used in CASE statement.
DEFAULT statement can be used only once in a SWITCH statement. It does not need a following constant, and this statement will be performed when there is no matching case. DEFAULT statement can be omitted if needless.
BREAK keyword in CASE/DEFAULT statement plays role of terminating SWITCH statement. If there is no BREAK keyword in the progress of program execution until reaches the other CASE/DEFAULT keyword, the program will continue to execute the statements in the other CASE/DEFAULT scope. This feature of SWITCH statement is helpful for short and simple programming, only if it is used intentionally.

4.5. GOTO Statement

This statement controls the flow of program execution directly. This statement needs a matching index in a program scope. Index is a unique name of a jump destination in a program scope, and it is started with ‘@’ character.

```
VAR A;

      A = 0;
@JumpHere                                    // destination of jump
      If (A < 10)
      {
            A = A + 1;
            Goto JumpHere;                   // change the flow of program
      }
```

This example program processes the same job with the previous examples of FOR and WHILE statement section. The name of index ('JumpHere' in the above example) can not be duplicated with variables, tags and program names.

4.6. CONTINUE Keyword

Continue keyword is very useful in loop. When a CONTINUE keyword is met in the progress of a program, the flow of the program will be moved to the first statement in the loop. This keyword is effective in the loop of WHILE, DO-WHILE and FOR statement. It is especially useful when there are many decision making in the loop such as IF-ELSE.
Following two program examples process the same job.

```
While (A<10)
{
      A = A + 1;
      If (A < 5)
      {
            // Job to do when A is smaller than 5
      }
      Else
      {
            // Job to do when A is larger than or equal to 5
      }
}
```

```
While (A<10)
{
        A = A + 1;
        If (A < 5)
        {
                // Job to do when A is smaller than 5
                Continue;
        }
                // Job to do when A is larger than or equal to 5
}
```

4.7. RETURN Keyword

Every program returns a result value to caller. RETURN keyword is used for designating a value to return and terminating the program. Following example program processes three input parameters and returns the sum of them to caller.

```
PARAM p1, p2, p3;

Return p1 + p2 + p3;
```

Assuming that the name of above program is ‘MyPgm’, it can be called from another program as following program line.

```
RtnValue = MyPgm(1, 2, 3);
```

After ‘MyPgm’ call, a value of 6 will be stored in ‘RtnValue’.

4.8. RUNSCRIPT Keyword

When a program calls another program, the caller program will be stalled until when the called program terminates.

RUNSCRIPT keyword provides another way of invoking other program. It does not make stall the caller program. After invoke another program, the caller program continues it’s own job. As a result, both programs will run in parallel and this is the feature of multi-threading.

Following example shows the multi-threading feature by using RUNSCRIPT keyword. It is assuming that the name of another program is ‘NewThread’.

```
If(A == 0)
      RunScript NewThread();          // Invoke another program (NewThread)
A = A + 1;                            // Continues next job without stalling.
```

5. Internal Functions

5.1. Function List

- R1, R2, …. : Real type input parameters
- S1, S2, …. : String type input parameters

(1) Trigonometric Functions

Function	Description	Return
Sin(R1)	Sine	Real
Cos(R1)	Cosine	Real
Tan(R1)	Tangent	Real
Asin(R1)	Arc Sine	Real
Acos(R1)	Arc Cosine	Real
Atan(R1)	Arc Tangent	Real
Sinh(R1)	Hyperbolic Sine	Real
Cosh(R1)	Hyperbolic Cosine	Real
Tanh(R1)	Hyperbolic Tangent	Real
Atan2(R1, R2)	Arc Tangent, returns the same result with Atan(R1/R2)	Real

(2) Other Mathematic Functions

Function	Description	Return
Log(R1)	Calculates logarithm	Real
Log10(R1)	Calculates logarithm	Real
Ceil(R1)	Calculates the ceiling of a value	Real
Floor(R1)	Calculates the floor of a value	Real
Abs(R1)	Calculates the absolute value	Real
Sqrt(R1)	Calculates the square value	Real
Rand()	Generates a pseudorandom number	Real
Fmod(R1, R2)	Calculates the floating-point remainder (R1/R2)	Real

(3) Special Functions

Function	Description	Return
PageOpen(S1)	Opens a new page named as S1	
PageClose(S1)	Closes an opened page	
FrameOpen(S1)	Opens a new frame named as S1	
StringTable(R1, R2)	Get a string from string table (R1=Group No., R2=String No.)	String
GetTime(R1)	Get a clock data (R1 : type of data)	Real

TimeStr(R1, S2)	Assemble a string of time R1 : clock value from GetTime(0) S2 : Format string	String
RunApp(S1, S2)	Executes an external program S1 : Program name S2 : Program input parameter	
MakeCsv(S1, R2)	Makes a CSV formatted log file in SD memory S1 : Name of log model R2 : Number of block (0=current logging block)	
MakeCsvUsb(S1, R2)	Makes a CSV formatted log file in USB memory S1 : Name of log model R2 : Number of block (0=current logging block)	
DataLog(S1, R2)	Start / Stop the data logging S1 : Name of log model R2 : 0=Stop, 1=Start	
RcpDownload(S1, R2)	Read a group of recipe data from file, and download it to device S1 : Name of recipe model R2 : Group number	
RcpUpload(S1, R2)	Upload the specified recipe group data from device, and store it to file. S1 : Name of recipe model R2 : Group number	
RcpStop(S1)	Stop and terminate the processing job of specified recipe model. S1 : Name of recipe model	
RcpFileStore(S1, R2)	Stores the recipe data to the file. S1 : Name of recipe model R2 : Group number	
RcpFileRead(S1, R2)	Read the recipe data from file. S1 : Name of recipe model R2 : Group number	
RcpMemDown(S1)	Download the current recipe data to device. S1 : Name of recipe model	
RcpMemUp(S1)	Upload the recipe data from device. S1 : Name of recipe model	
RcpCsvRd(S1, S2, R3)	Read the recipe data from CSV formatted file.	

	S1 : Name of recipe model S2 : File name R3 : File location (0=local, 1=SD/MMC, 2=USB)	
RcpConfig()	Pops up a dialog box for recipe data manipulation.	
TrendCsvWr(S1, R2)	Make a CSV formatted file from trend data. S1 : Trend name R2 : File location (0=local, 1=SD/MMC, 2=USB)	
ScrCapture(S1, R2)	Make a BMP formatted file of current screen S1 : BMP file name. R2 : File location (0=local, 1=SD/MMC, 2=USB)	
Sleep(R1)	Put delay on the script program R1 : delay time in milli-second	
HardCopy()	Print out the current screen image to USB printer.	
OpenPort(R1, R2, R3, R4, R5)	Open a serial port for communication. R1 : Port number (predefined as _**COMxxx**_) R2 : Baudrate code (predefined as _**BPSxxxx**_) R3 : Parity code (predefined as _**PARITY_xxx**_) R4 : Data bits (7 or 8) R5 : Stop bits code (predefined as _**STOPBIT_xxx**_)	1 : OK. 0 : Error
ClosePort(R1)	Close a serial port. R1 : Port number (predefined as _**COMxxx**_)	1 : OK. 0 : Error
SendByte(R1, R2)	Transmit a byte datum though the R1 port. R1 : Port number (predefined as _**COMxxx**_) R2 : A datum to transmit. (0..255)	1 : OK. 0 : Error
SendString(R1, S2)	Transmit a string data though the R1 port. R1 : Port number (predefined as _**COMxxx**_) S2 : A string to transmit.	1 : OK. 0 : Error
ReceiveByte(R1, R2)	Receive a byte datum though the R1 port. R1 : Port number (predefined as _**COMxxx**_) R2 : Time-out in milli-second.	Received data or error.

5.2. Function Details

PageOpen(S1)	**Opens a new page**

[S1] new page name (without extension)

Opens a new screen page. If the new page has no special style, then previous page or frame will be closed automatically. Otherwise, according to the style of the new page, following result can be shown.

(1) A pop-up style If a new page has a style of pop-up, current page does not affected by this new page. The new page pops up over current page. (2) A page for frame This kind of page has a position number in a frame. For open this page, the frame has to be opened in advance. The old page which has the same position number in the frame will be replaced by this page automatically.

PageClose(S1)	Closes an opened page
[S1] page name (without extension) Closes an opened screen page. If the page of S1 parameter was not opened, the current screen will not be affected by this function.	

FrameOpen(S1)	Opens a new frame
[S1] new frame name (without extension) Opens a new screen frame. Previous page or frame will be closed automatically. The default pages for frame will be opened automatically.	

StringTable(R1, R2)	Gets a string data from string table
[R1] String group number [R2] String number in a group Gets a string data from the string table. The string table must have been configured by ‘String Editor’. Assign the group number of a string table to the R1, and the string number of a string in that group to the R2 parameter. Returned data is a string type value, and it can be used for string tag, variable or other function’s parameter.	
[Example Program] StrTag = StringTable(1, 14); [Result] StrTag : “A string that is defined as 14 in group #1”	

GetTime(R1)	Gets the current clock data
[R1] type of clock data 0 : The number of seconds after 1st. January 1970 UTC 1 : the number of year (1970..) 2 : the number of month (1..12) 3 : the number of day (1..31)	

4 : the number of hour (0..23) 5 : the number of minute (0..59) 6 : the number of second (0..59) 7 : the number of day of week (1..7 : 1=Sunday, 2=Monday…, 7=Saturday) 8 : the number of today's minute count (0..1439 : 0=midnight) 9 : the number of today's second count (0..86399, 0=midnight)

TimeStr(R1, S2)	Makes a string of clock
[R1] Clock data (It can be acquired by GetTime(0) function call) The number of seconds after 1st. January 1970 UTC [S2] Format string (can include following symbols) %A : Full name of weekday (ex : Sunday) %a : Abbreviation of weekday (ex : Sun) %B : Full name of month (ex : January) %b : Abbreviation of month (ex : Jan) %d : date (1..31) %H : Hour in 24-hour format (0..23) %I : Hour in 12-hour format (1..12) %m : Month (1..12) %M : Minute (0..59) %p : A.M. / P.M. indicator for 12-hour format (A.M., P.M.) %S : Second (0..59) %y : Number of year without century (00..99) %Y : Number of year with century (ex : 2006)	
[Example Program] CurTime = GetTime(0); StrTag = TimeStr(CurTime, "%Y/%m/%d %H:%M:%S") ; [Result] StrTag : "2006/10/12 23:59:59"	

RunApp(S1, S2)	Executes an external application program.
[S1] Application program name with path and extension [S2] Input parameter string for the application program. Executes an external application program. This program can be not only system included program, also user programmed program.	
[Example Program]	

RunApp(“Ping.Exe”, “-t 100.100.100.1”);

[Result]

Ping test screen will be popped up when this command or program is issued.

MakeCsv(S1, R2)	Makes a CSV formatted log file in SD memory

[S1] Log model name

[R2] Block number of designated log model.

Converts the designated block file to a CSV formatted file and stores it on the SD/MMC memory. A SD/MMC memory has to be plugged before the issuing of this function. A CSV file will be written in the folder ‘XPanel’ of the SD/MMC memory card. The CSV file name is assembled with the date and time log started, and it’s from original block file. (YYMMDD HHmmSS.CSV)

When the R2 parameter is zero, this function will try to convert the current log proceeding (uncompleted) block file. If the logging task of designated model is not running at the moment of the function call, no CSV file will be created.

The parameter R2 can be designated up to the ‘Maximum Block No.’ which should have been set in the model configuration dialog box. When the R2 parameter is 1, this function will try to convert the most recently log completed block file.

[Example Program]

MakeCsv(“ModelName”, 1);

[Result]

A CSV file will be created from the most recently log completed block file.

MakeCsvUsb(S1, R2)	Makes a CSV formatted log file in USB memory

[S1] Log model name

[R2] Block number of designated log model.

Converts the designated block file to a CSV formatted file and stores it on the USB memory. An USB memory has to be plugged before the issuing of this function. A CSV file will be written in the folder ‘XPanel’ of the USB memory card. The CSV file name is assembled with the date and time log started, and it’s from original block file. (YYMMDD HHmmSS.CSV)

When the R2 parameter is zero, this function will try to convert the current log proceeding (uncompleted) block file. If the logging task of designated model is not running at the moment of the function call, no CSV file will be created.

The parameter R2 can be designated up to the ‘Maximum Block No.’ which should have been set in the model configuration dialog box. When the R2 parameter is 1, this function will try to convert the most recently log completed block file.

[Example Program]

MakeCsvUsb(“ModelName”, 1);

[Result]

A CSV file will be created from the most recently log completed block file.

DataLog(S1, R2)	Data logging start / stop.

[S1] Log model name

[R2] 0 or 1 (0 : STOP, 1 : START)

This function is effective only on the log model of external call start type. Models with other start type such as periodic, trigger tag, enable tag or on time are not affected by this function.

If the R2 parameter is 0, the designated logging task will be terminated. Otherwise, the logging task will be invoked by this function. This function treats the non-zero value of R2 as 1.

[Example Program]

DataLog(“ModelName”, 1);

[Result]

Data logging task will be invoked.

RcpDownload(S1, R2)	Reads a recipe data group from file, and downloads to device

[S1] Recipe model name

[R2] Group number

Reads a group data from the recipe configuration file, and download them to the device. After this function, the group data in XPanel memory will be changed to the read group data. The group data must have been configured in XPanel Designer before this function call.

[Example Program]

RcpDownload(“ModelName”, 1);

[Result]

Reads the #1 group data from “ModelName” recipe file, and download them to the device.

RcpUpload(S1, R2)	Uploads a recipe group data from device, and stores them to the recipe configuration file.

[S1] Recipe model name

[R2] Group number

Reads a group data from the device through communication, and stores them to the recipe configuration file. After this function, the group data in XPanel memory will be changed to the read group data. The designated recipe group must have been declared in XPanel Designer before this function call.

[Example Program]

RcpUpload(“ModelName”, 2);

[Result]

Uploads the recipe data from the device, and stores them to the group #2 of “ModelName” recipe file.

RcpStop(S1)	Stops and terminates the recipe operations.

[S1] Recipe model name

This function terminates any job related with designated name of recipe model. For example, if download process is proceeding through communication when this function is issued, then the communication will be stopped and terminated. Be careful to use this function, because there is a possibility of storing unwanted data group in file or device. Use this function only when there is a unrecoverable network or device breakdown.

RcpFileStore(S1, R2)	Stores the current recipe data in memory to the file.

[S1] Recipe model name

[R2] Group number

This function stores a group data in memory of XPanel to the configuration file. The designated recipe group must have been declared in XPanel Designer before this function call.

RcpFileRead(S1, R2)	Reads a group data from the recipe configuration file.

[S1] Recipe model name

[R2] Group number

This function reads a group data in the configuration file and stores them to the memory of XPanel. The designated recipe group must have been configured by XPanel Designer, or stored by other way before this function call.

RcpMemDown(S1)	Downloads a group of recipe data in memory to the device.

[S1] Recipe model name

This function downloads a group data in memory of XPanel to the device. The group data must have been loaded in memory by other way before this function issue.

RcpMemUp(S1)	Uploads a group of recipe data to memory from the device.

[S1] Recipe model name

This function uploads a group data in device to the memory of XPanel. The uploaded group data can be processed by other functions or recipe tool window.

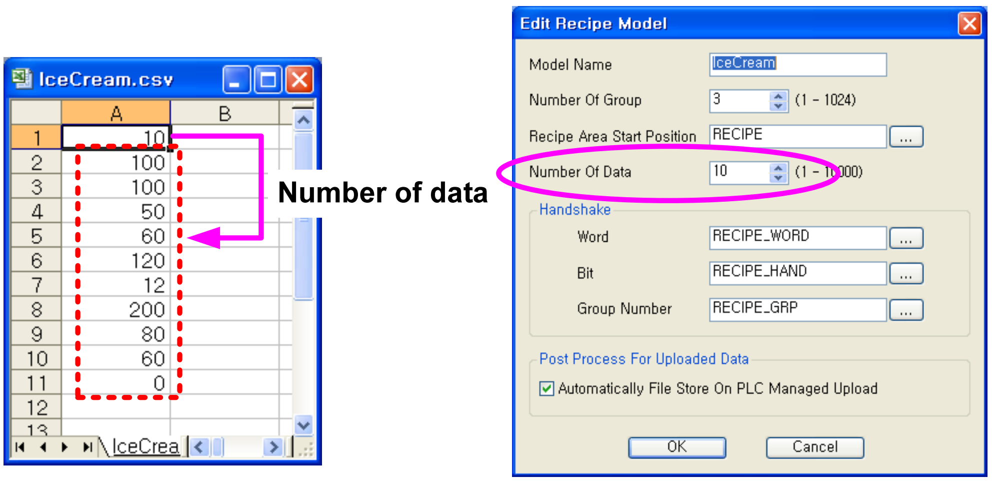
RcpCsvRd(S1, S2, R3)	Reads a group of recipe data from CSV formatted file.

[S1] Recipe model name

[S2] CSV file name (without extension and path)

[R3] CSV file location code (0 : local, 1 : SD/MMC, 2 : USB)

This function reads a group of data from CSV formatted file and stores them to the memory of XPanel. The CSV formatted file must have information about the number of data it has. This number must be located in the top most of data list, and the number must be matched with the one that assigned in the recipe model setup dialog box.



	A
1	10
2	100
3	100
4	50
5	60
6	120
7	12
8	200
9	80
10	60
11	0
12	

Model Name: IceCream
Number Of Group: 3 (1 ~ 1024)
Recipe Area Start Position: RECIPE
Number Of Data: 10 (1 ~ 10000)
Word: RECIPE_WORD
Bit: RECIPE_HAND
Group Number: RECIPE_GRP
☑ Automatically File Store On PLC Managed Upload

The second parameter S2 is a name of CSV file to read. There is no need to specify the extension. But, the path of the CSV file location must be root folder of each storage media. The third parameter R3 indicates the storage media where the CSV file was stored. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R3 instead of number.

Media	Number	Symbol	CSV file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

RcpCsvRd(“ModelName”, “RcpFileName”, _SDMEM_);

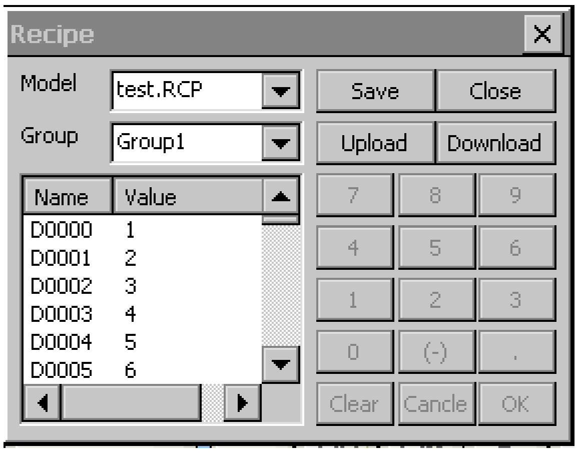
[Result]

Reads a CSV formatted file named as “RcpFileName.CSV” in SD/MMC memory card, and stores the data in memory of XPanel.

RcpConfig()	Pops up the recipe data manipulation tool window.

No Parameter

This function pops up a tool window for recipe. This tool window shows the current recipe data in memory. And it provides some tools for editing individual datum in a group, changing current group, saving the current data to file and up/downloading the recipe data from/to the device. (Please refer to the ‘**Recipe**’ section for more information)



Model: test.RCP

Group: Group1

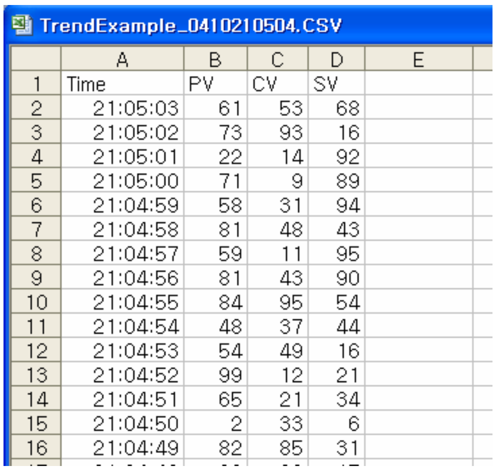
Name	Value
D0000	1
D0001	2
D0002	3
D0003	4
D0004	5
D0005	6

TrendCsvWr(S1, R2)	Makes and stores a CSV formatted file from trend.

[S1] Trend name

[R2] CSV file location code (0 : local, 1 : SD/MMC, 2 : USB)

This function makes a CSV formatted file from a trend object. Output CSV file can be stored in not only local flash disk, but external memory card. Next screen capture shows an example of output CSV file.



TrendExample_0410210504.CSV

	A	B	C	D	E
1	Time	PV	CV	SV	
2	21:05:03	61	53	68	
3	21:05:02	73	93	16	
4	21:05:01	22	14	92	
5	21:05:00	71	9	89	
6	21:04:59	58	31	94	
7	21:04:58	81	48	43	
8	21:04:57	59	11	95	
9	21:04:56	81	43	90	
10	21:04:55	84	95	54	
11	21:04:54	48	37	44	
12	21:04:53	54	49	16	
13	21:04:52	99	12	21	
14	21:04:51	65	21	34	
15	21:04:50	2	33	6	
16	21:04:49	82	85	31	

The name of generated CSV file has the format of “***trendname*_MMDDHHmmSS.CSV**”. This name is generated from trend name, date and time.

The second parameter R2 indicates the storage media where the CSV file will be stored. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R2 instead of number.

Media	Number	Symbol	CSV file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

TrendCsvWr(“TrendExample”, _SDMEM_);

[Result]

Makes a CSV formatted file on SD/MMC cards. The data will be taken from the “TrendExample” trend data.

ScrCapture(S1, R2)	Captures current screen and makes a BMP format image file.

[S1] Seed for making BMP file name

[R2] BMP file location code (0 : local, 1 : SD/MMC, 2 : USB)

This function captures current screen display and makes a BMP formatted image file. The name of BMP file will be made with the seed string of S1 parameter and time. File name format is “***seedname*_HHmmSS.BMP**”.

The second parameter R2 indicates the storage media for generated BMP file. It can be one of three numbers, 0 to 2. XPanel provides predefined constants for them. That is, following constant symbols can be used as parameter R2 instead of number.

Media	Number	Symbol	BMP file location
Internal Flash Disk	0	_LOCAL_	\\XPanel\
SD/MMC	1	_SDMEM_	Root
USB	2	_USBMEM_	Root

[Example Program]

ScrCapture(“Mybmp”, _SDMEM_);

[Result]

Captures current screen and makes a BMP formatted file on SD/MMC cards.

Sleep(R1)	Puts some delay to the current script program

[R1] Delay time in milli-seconds

This function suspends the execution of the current script program for a specified interval (R1). The interval must be specified in milli-seconds unit.

HardCopy()	Prints out the current screen image to USB printer
This function prints out the screen image to USB printer. The printer must support PCL (Printer Control Language).	

OpenPort(R1, R2, R3, R4, R5)	Opens the R1 serial port for communication.
[R1] COM port number (use the predefined symbol. _COMxxxx_) [R2] Baudrate (use the predefined symbol, _BPSxxxx_) [R3] Parity (use the predefined symbol, _PARITY_xxx) [R4] Data bits (7 or 8) [R5] Stop bits (use the predefined symbol, _STOPBIT_xxx) This function opens a serial port for communication. XPanel has two serial communication ports (COM1 and COM2). The COM1 port can be configured as one of three modes (RS232C, RS422 and RS485). Predefined symbols for R1, R2, R3 and R5 parameters are provided, please refer to the 'Constants' section of this manual. The port which is being used for device communication or already opened, cannot be opened by this function. Before calling this function, ensure that the designated port is not being used in device driver. And also, if the port has been opened and used previously before this function call, ensure that it was closed by ClosePort() function. If there is an error for opening a port, this function returns zero (0). When the port was opened successfully, this function returns non-zero value.	
[Example Program] OpenPort(_COM485_, _BPS19200_, _PARITY_NONE_, 8, _STOPBIT_ONE_); SendByte(_COM485_, 2); …… ClosePort(_COM485); [Result] Opens the COM1 port for communication if it is available for using. The port will be set up as RS485 communication mode, 19200 bps, no parity, 8 data bits and 1 stop bit.	

ClosePort(R1)	Closes the R1 serial port which has been opened.
[R1] COM port number (use the predefined symbol. _COMxxxx_) This function closes the designated serial port which has been opened by OpenPort() function. Predefined symbols for R1 parameter are provided, please refer to the 'Constants' section of this manual.	
[Example Program] OpenPort(_COM485_, _BPS19200_, _PARITY_NONE_, 8, _STOPBIT_ONE_);	

```
SendByte(_COM485_, 2);
……
ClosePort(_COM485);
```
[Result]

Closes the COM1 port.

SendByte(R1, R2)	Transmits a byte datum (R2) through the R1 port.

[R1] COM port number (use the predefined symbol. _COMxxxx_)

[R2] A datum to transmit. The value can be between 0 and 255.

This function transmits a datum through the designated serial port (R1) which has been opened by OpenPort() function. Predefined symbols for R1 parameter are provided, please refer to the 'Constants' section of this manual.

The R2 parameter must be a byte datum, between 0 (0x00) and 255 (0xFF). If a value larger than 255 (0xFF) was designated to the R2, the upper bytes will be discarded and only the lower 1 byte datum will be transmitted. For example, if the value 0x1234 was assigned to the R2, the 0x34 will be transmitted.

Ensure that the R1 port must have been opened by OpenPort() function, before calling this function.

[Example Program]
```
OpenPort(_COM485_, _BPS19200_, _PARITY_NONE_, 8, _STOPBIT_ONE_);
SendByte(_COM485_, 2);
……
```
[Result]

Transmits a byte datum 2 (0x02) through the COM1 port in RS485 mode.

SendString(R1, S2)	Transmits a string (S2) through the R1 port.

[R1] COM port number (use the predefined symbol. _COMxxxx_)

[S2] A string to transmit. The maximum length of the string is 255 characters.

This function transmits a string through the designated serial port (R1) which has been opened by OpenPort() function. Predefined symbols for R1 parameter are provided, please refer to the 'Constants' section of this manual. The S2 parameter must be a string or string tag.

Ensure that the R1 port must have been opened by OpenPort() function, before calling this function.

[Example Program]
```
OpenPort(_COM485_, _BPS19200_, _PARITY_NONE_, 8, _STOPBIT_ONE_);
SendString(_COM485_, "This string will be transmitted.");
```

```
……
```

[Result]

Transmits a string through the COM1 port in RS485 mode.

ReceiveByte(R1, R2)	Receives a byte datum from R1 port.

[R1] COM port number (use the predefined symbol. _COMxxxx_)

[R2] Timeout in milli-second.

This function waits and receives a byte datum from designated port (R1) which has been opened by OpenPort() function. If there is no datum until when the timeout (R2) expires, this function will return 256 (0x100). Otherwise, a received byte datum will be returned. (0 to 255)

R1 parameter specifies the COM port for receiving data, and predefined symbols can be used for this parameter. Please refer to the 'Constants' section of this manual.

Ensure that the R1 port must have been opened by OpenPort() function, before calling this function.

[Example Program]

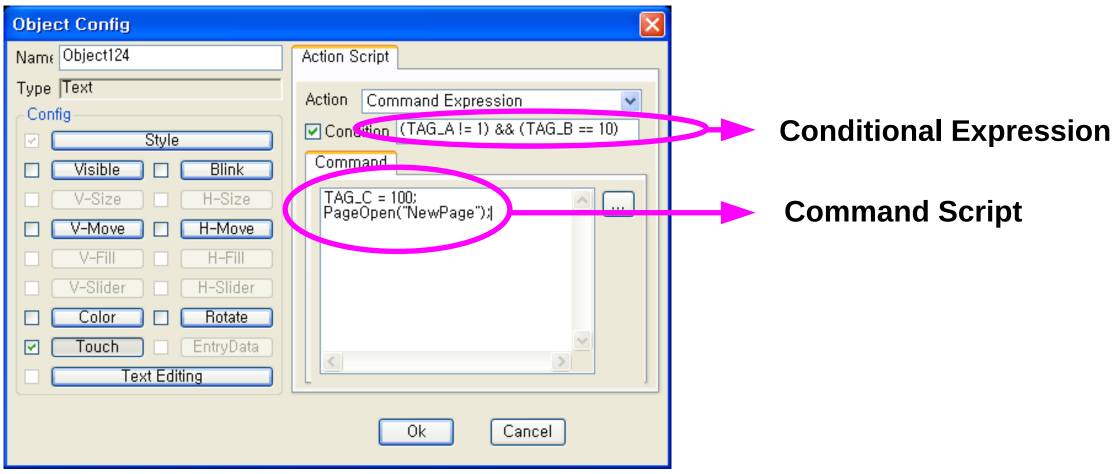
```
VAR RxData;
OpenPort(_COM485_, _BPS19200_, _PARITY_NONE_, 8, _STOPBIT_ONE_);
RxData = ReceiveByte(_COM485_, 1000);
If(RxData < 256)
{
}
……
```

[Result]

Receives a datum from the COM1 port in RS485 mode, and stores the datum in 'RxData' variable.

6. Scripts for Command and Condition

In some application, there is a need to use a simple script on graphic object configuration. For example, to assign a touch operation on an object, one or several command lines must be written in case of "Command Expression" action type (see an example configuration dialog box below). And further more, for making it conditional touch operation, a conditional expression must be described. All these conditional expression and command script have the same rules with script programming.



Name Object124
Type Text
Config: [✓ Touch]
Action Command Expression
[✓] Condition (TAG_A != 1) && (TAG_B == 10)
Command

```
TAG_C = 100;
PageOpen("NewPage");
```

6.1. Conditional Expression

The conditional expression always returns only a logical value. A logical value can be one of two values, TRUE or FALSE. XPanel uses the number '0' as the value of logical FALSE, and '1' as the value of logical TRUE. A logical expression needs some special operators. The logical operators in XPanel are shown in as following table. For more precise information, refer to the section 3.2.

Logical Operator	Description	Example
&&	Logical AND	A && B
||	Logical OR	A || B
!	Logical NOT	!A
Comparative Operator	**Description**	**Example**
==	EQUAL	A == B
!=	NOT EQUAL	A != B
>	Big	A > B
>= or =>	Big or Equal	A >= B
<	Small	A < B
<= or =<	Small or Equal	A <= B

All logical operators get one or two values for input (operands) and return a logical output value. According to the above examples, 'A' and 'B' are those inputs. The value for logical arithmetic operand is not restricted to only two logical values (TRUE or FALSE). The input value can be any number. But, only zero is treated as logical FALSE. All the other values are treated as TRUE.

Comparative operators get two real values for operands and return a logical output value. Those logical output values can be used in other logical expression.

Here is some examples of conditional expression that can be used in object configuration dialog box.

Conditional Expression Example	Remark
(Tag_a != 1) && (Tag_b == 10)	If the value of Tag_a is not equal to 1 and the value of Tag_b is equal to 10, then the expression returns TRUE. Otherwise FALSE.
Sin(Tag_a) == 1	If the sine of Tag_a is 1, then the expression will return TRUE, otherwise FALSE.
Tag_a < 100	If the value of Tag_a is smaller than 100, then the expression will return TRUE, otherwise FALSE.
(Tag_a + Tag_b) < (Tag_a + Tag_c)	If the sum of Tag_a and Tag_b is smaller than the sum of Tag_a and Tag_c, then the expression will return TRUE, otherwise FALSE.

6.2. Command Script

The command in XPanel is not restricted to use only one instruction. A long list of instructions can construct a command. That is, the command is a special kind of script program. Within a command script, all kinds of keywords and functions can be used.

But only one thing, the priority of command script is different with the normal script. When a command script is issued by touch operation or any other method, it will be executed with top-most priority. For that reason, if a command script has a time consuming loop inside, all the other process will be halted until the end of command script. So, especially for calling other function within a command script, the programmer has to keep in mind that the priority. If the function has a loop, '**RunScript**' keyword must be used. This keyword is used for parallel execution. The calling command script does not wait the termination of called function.

Assume that a programmer wants to call a function named as 'MyLoop', and this function has a time consuming loop. Then, following instruction must be used, instead of calling directly.

RunScript MyLoop();

Following table shows some typical command script examples.

Command Script Example	Remark

Tag_a = 100;
PageOpen(“NewPage”);
	After changing the value of Tag_a to 100, open a page named as “NewPage”.

Tag_b = Tag_b + 1;	Increase the value of Tag_b by 1.

Tag_a = 100;
Tag_b = 1;
RunScript MyLoop();
MakeCsv(“LogModel”, 1);
	After changing the values of Tag_a and Tag_b, invoke MyLoop() script function. After that, do not wait the termination of MyLoop(), and create a CSV formatted file of the ‘LogModel’ on SD memory card.