



MetaDevice User Manual

Version 1.0

User Manual for MetaDevice

Version 1.0

Last revised on April 23, 2002

Copyright

Copyright 2000, Sena Technologies, Inc. All rights reserved.

Sena Technologies reserves the right to make changes and improvements to its product without providing notice.

Trademark

MetaDevice™ is a trademark of Sena Technologies, Inc.

Windows® is a registered trademark of Microsoft Corporation.

Ethernet® is a registered trademark of XEROX Corporation.

Notice to Users

If system failure should cause serious consequences, protecting life and property against such consequences with a backup system or safety device is essential. The user agrees that protection against consequences resulting from system failure is the user's responsibility.

This software is not approved for life-support or medical systems.

Changes or modifications to this device not explicitly approved by Sena Technologies will void the user's authority to operate this software.

Company Address

Sena Technologies, Inc.

210 Yangjae-dong, Seocho-gu

Seoul, Korea 137-130

Telephone: +82-2-573-7772

Fax: +82-2-573-7710

Email: info@sena.com

Website: <http://www.sena.com>

Contents

Part I : General

- 1 General
 - 1.1 Overview
 - 1.1.1 MetaDevice Vision
 - 1.1.2 User Function
 - 1.1.3 MetaDevice Function
 - 1.2 MetaDevice Structure
 - 1.2.1 Deployment Structure
 - 1.2.2 Run-Time Structure
 - 1.2.3 Set Up of User Environment
 - 1.3 MetaDevice Functional Characteristics
 - 1.4 Implementation Methods using MetaDevice
- 2 Installing
- 3 MetaDevice Implementation Procedure
 - 3.1 Server System Implementation Procedure
 - 3.2 User Interface Implementation Procedure
 - 3.3 Summarize Implementation Procedure

Part II : Server Manager/Admin

- 1 Server Manager/Admin Overview and Basic Functions
 - 1.1 Set Server Environment Variables
 - 1.2 Database Admin
 - 1.3 Protocol Editing
 - 1.4 Semantics Editing
 - 1.5 Transfer Editing
 - 1.6 Server Program Deployment
- 2. ADMIN Module
 - 2.1 User Management

- 2.2 Device Management
- 2.3 Example of Device Management
- 2.4 Device Modifying/Deletion
- 2.5 User/Device Relation
- 2.6 Additional Functions

3. General Property

- 3.1 Project Name
- 3.2 Location
- 3.3 Daemon Type
- 3.4 Port Number
- 3.5 DB Used
- 3.6 DBMS
- 3.7 DB Connection Property

4. Protocol

- 4.1 Protocol List Panel
- 4.2 Protocol Edit Panel

5. Transfer

- 5.1 Transfer List Panel
- 5.2 Transfer Edit Panel

6. Semantics

- 6.1 Semantics List Panel
- 6.2 Semantics Edit Panel

Part III : UI Manager

1 UI Manager Overview

- 1.1 Role of UIManager
- 1.2 Notes and Method of Usage

2 Menu Bar

3 Tool Bar

4. Controls Component Group

- 4.1 PageLinkButton
- 4.2 DeviceDynamicSelector
- 4.3 DeviceSelector
- 4.4 DataReceiveButton
- 4.5 DataSendButton
- 4.6 DataMultiSendButton
- 4.7 InstantDataSendButton

5. Setters Component Group

- 5.1 ComboBoxValueSetter
- 5.2 SpinValueSetter
- 5.3 RotarySwitch
- 5.4 VerticalSliderSwitch
- 5.5 HorizontalSliderSwitch
- 5.6 ImageOnOffButton
- 5.7 InputFieldValueSetter

6. Getters Component Group

- 6.1 RotaryGauge1, RotaryGauge2
- 6.2 VerticalLinearGauge1, VerticalLinearGauge2, VerticalLinearGauge3
- 6.3 HorizontalLinearGauge1, HorizontalLinearGauge2, HorizontalLinearGauge3
- 6.4 SevenSegmentDisplay
- 6.5 LabelValueGetter
- 6.6 TextLabelValueGetter
- 6.7 Graph2D
- 6.8 DataMappingPanel
- 6.9 ImageDisplayValueGetter
- 6.10 ImageOnOffGauge

7. DBAccess Component Group

- 7.1 DBAccess Table
- 7.2 DBAccess Graph

8. Static Component Group

- 8.1 Static Image

8.2 Static Text

8.3 Static Rectangle

8.4 Static Ellipse

9 FileSetters Component Group

9.1 FileChooserListBox

Part IV : Examples

1 Sample Procedure

2 Arithmetic Operation Example

Part I General

1. Foreword

↓ MetaDevice is an integrated development tool for establishing Embedded Internet Application Systems.

The basic composition of the MetaDevice consists of User and Device Information Management, the Server Manager, and the UI Manager. The Server Manager is used for systematically managing Device Communication Protocols and the UI Manager is used for Graphic User Interface. Furthermore, the MetaDevice is an integrated development tool based on Java, supporting Multi-User, Multi-Device, Database and also Client/Server, Web Run-Time environments(Applets).

MetaDevice can reduce costs required for major developments or maintenance, thus enabling users to significantly increase their competitiveness.

↓ MetaDevice has the following characteristics.

- Integrates heterogeneous communication devices with TCP/IP Protocol
- facilitates distribution / integration of separate Applications
- Intuitive Rule-based Implementation of heterogeneous device Protocols
- Takes advantage of characteristic object-oriented-program benefits, enhancing user-friendliness
- Legacy interface by utilizing DB, API(Application Program Interface) which is called Transfer
- Supports all Ethernet TCP/IP products which use TCP/IP Protocol

- Version 1.3 Recommended Environment

DB : MS Access-2000, MS-SQL

MS Access: supports Single User, Single Tasking: 1 simultaneous user

OS : MS-Windows series- (Client)

Windows-2000 Server, Windows NT Server (Server)

VM : JRE1.3.1_02(International Version), JDK1.3.1_02

JDBC Driver: Microsoft JDBC Driver.

↓ Technological Support

Sena Technologies, Inc.

210 Yangjae-dong, Seocho-gu

Seoul, Korea 137-130

Telephone: +82-2-573-7772

Fax: +82-2-573-7710

e-mail : support@sena.com

Website: <http://www.sena.com>

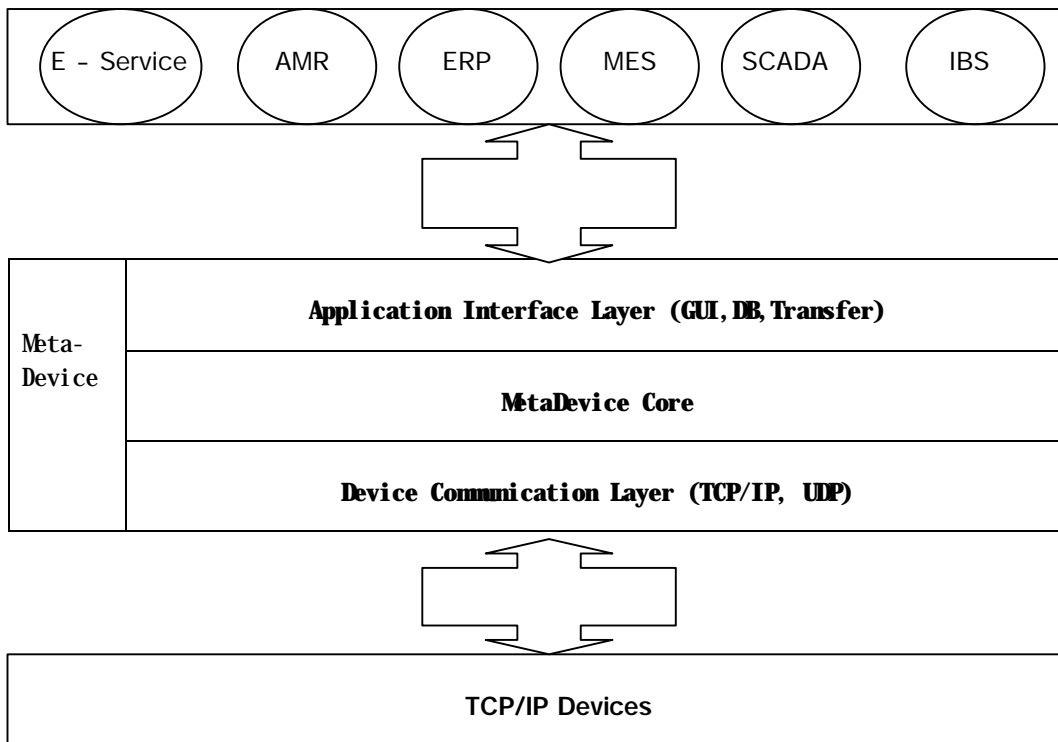
1.1 Overview

The MetaDevice is a tool facilitating the building of embedded Internet Application systems. It greatly improves productivity and enhances convenience from seamless implementation that does not need to depend on hard coding.

Therefore, MetaDevice users can build their application systems more easily by using the Server Manager and UI Manager provided by the MetaDevice.

Generally when planning a device communication program the program developer must ponder on how to establish the system using the given user device protocol. However, the Metadevice has eliminated such troubles through Built-In Class and Procedures. Furthermore, the MetaDevice can be established according to a business-oriented logic rather than the individual program developer's, making integration with various other business systems possible.

1.1.1 MetaDevice Vision



* **E-Service** : Controlling and monitoring services through the Internet; via Internet Sign Boards, Internet Card Readers, UPS, etc.

1.1.2 User Functions

- Graphic User Interface

- . Graphic editing
- . Graphic controlling/monitoring for each device location
- . Providing various Graphic functions and libraries
- . Alarm Propagation : supported in Version 1.5

- Real Time Monitoring

- . Real time trend monitoring
- . Real time status monitoring

- History Monitoring

- . Periodic data logging
- . Setting Data storage period
- . Setting Data storage condition

- Communication between heterogeneous devices

- . Exchanging protocol data between heterogeneous devices with differing protocols

- Communication with business divisions

- . Exchanging real time data with business systems through database

- Additional functions

- . Composing data reports
- . Analyzing data history
- . Integrating legacy
- . User and device management

- Alarm : supported in MetaDevice Ver 1.5

- . Setting Alarm for specific devices/locations
- . Alarm Management Functions
 - Alarm Conditions
 - Alarm definition/classification/registration/deletion
 - High/Low Alarm
 - Graphic Monitoring Interface
 - Acoustic/Color Alarm

1.1.3 MetaDevice Functions

Server Manager

- . General Panel
 - Setting for Project Parameter, DataBase Parameter, and Server Type
- . Protocol Editor
 - Setting for user device communication Protocol (Variables, Constant , CheckSum)
- . Semantics Editor
 - Setting Communication Process between Device and Server
- . Transfer Editor
 - Transferring and resetting Protocol between Device and UI, Device and Device

Admin Module

- Managing base information of MDPR File operated in server manager through Database
(Managing User information, Device information, integration between User and Device, Related Admin Tables)

UI Manager

- . Design Panel
 - Graphic Design Panel presented in Run-Time
- . Page Hierarchy & Device Information Panel
 - Page Hierarchy : Establishing interdisciplinary relations by setting Page for each screen
 - Device Information : Acquiring Server Information before UI Design and deployment
- . Component Panel
 - Controls, Setters, Getters, Statics, File Setter Component Selection Panel
- . Property Panel
 - Communication variable property and graphic property setting panel for each component
- . Message Panel
 - Providing Message and Protocol view of usage while using UI Manager

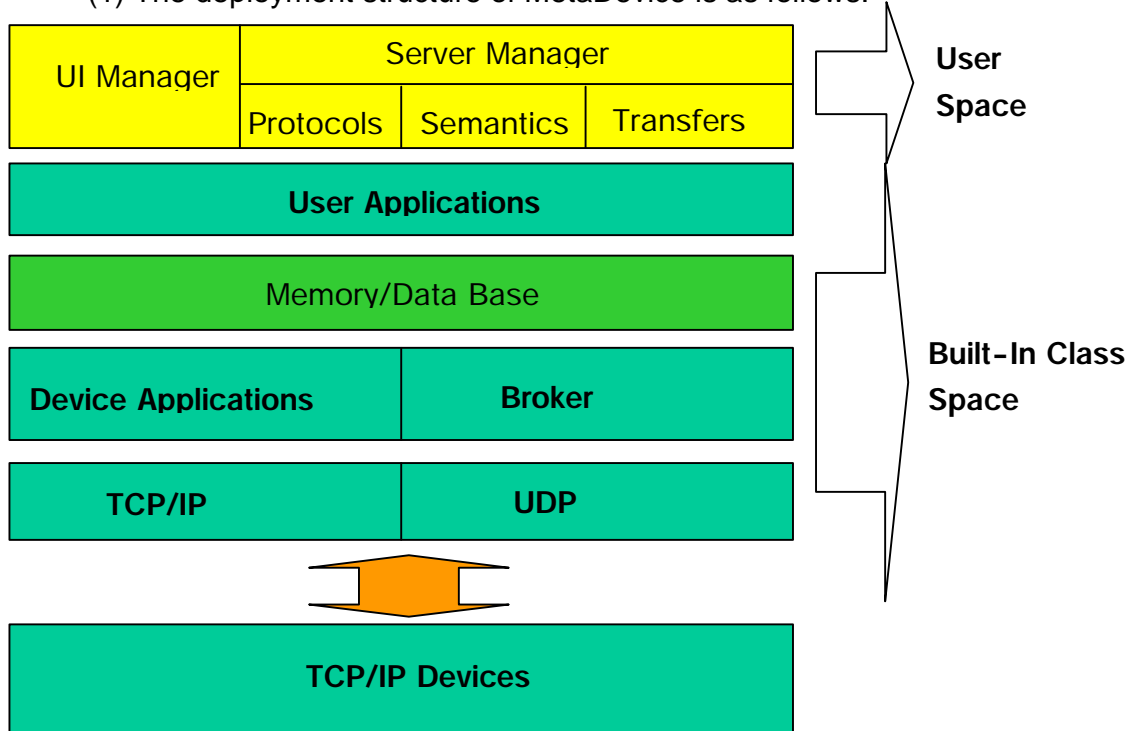
Item	Specification	Server Specification Recommended
HDD		Minimum 4GB
OS		Windows 2000 Server, Windows NT
Base Memory		256 MB
Base Memory for 50 devices		512 MB
Incremental Memory per 1 device		2~3 MB
DataBase		MS SQL
JDBC Driver		Microsoft JDBC Driver
Concurrent User		Depends on DataBase and JDBC Driver

[Table 1.1 MetaDevice V1.3 Recommended Server Specifications]

1.2 MetaDevice Structure

1.2.1 Deployment Structure

(1) The deployment structure of MetaDevice is as follows:



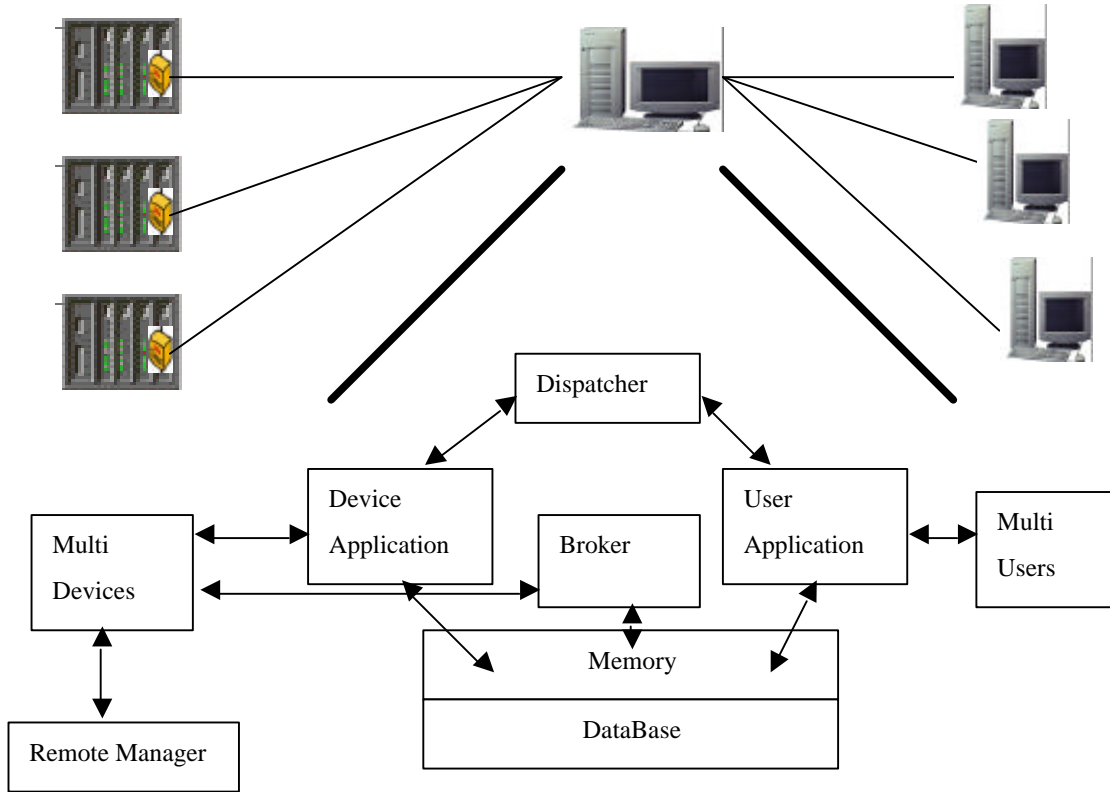
[Fig 1.1 MetaDevice Deployment Structure]

(2) Description of Deployment Structure

- **UI Manager** : Communication between host and user Client and formation of GUI Application/Applet
- **Protocols** : User Protocol Data Mapping
- **Semantics** : Defining communication process between Device and Server
- **Transfers** : Transferring Protocol data between device and device, device and GUI
- **User Applications:** Communication Program between GUI and Server
- **DataBase:** Data Logging during Device Communication
- **Device Applications:** Communication Program between Server and User Device.
- **Broker:** Acquiring information of Device IP, Device Tag, and ID

1.2.2 Run-Time Structure

(1) The MetaDevice Run-Time structure is as follows:



[Fig 1.2 MetaDevice Run-Time Structure]

(3) Description of Run-Time Structure

Device Appl. : Network Communication Program based on defined Protocol between server and host.

User Appl. : Communication Program between GUI Client and Server Host.

Broker : Program connecting User and User Device through IP

Dispatcher : Communication process handling program between User Process(User Application) and Device Process (Device Application)

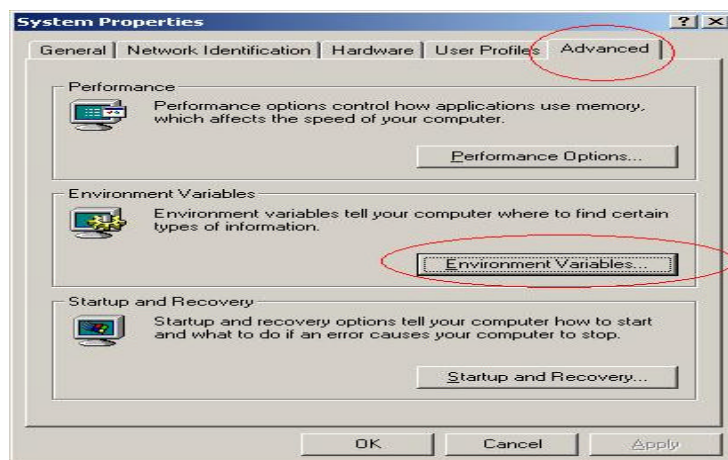
Remote Manager : Program connecting directly to Device from remote section in case of Device irregularities (HelloDevice IDE)

1.2.3 Set Up of User Environment

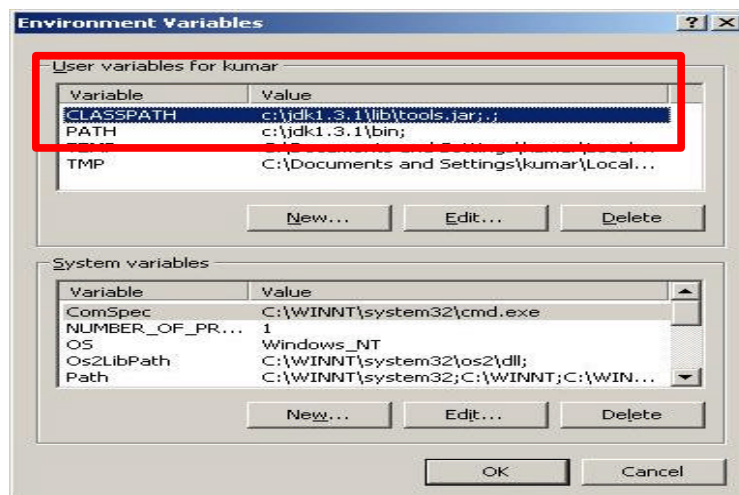
MetaDevice is purely implemented with Java Language and optimized with JDK1.3.1_02.

Server requires installment of JDK and Client requires JDK1.3.1_02 or JRE1.3.1_02. You can download the JDK Install Program by visiting java.sun.com. After installing JDK, you should set the Path of JDK or JRE.

If you use Windows NT, click the right mouse button on the desktop's **"MyComputer"** icon and set Path by selecting **"Properties"** and then **"Environment"**. If you use Windows 2000, click the right mouse button on the desktop's **"MyComputer"** icon. Set path by selecting **"Properties"**, **"Advanced"** and then **"Environment Variable"**. If you use Window 95/98, set Path by "Autoexec.bat" file and reboot your system.



[Fig 1.3 Window 2000 Environment Variables]



[Fig 1.4 Setting Path in Window 2000 environment]

1.3 MetaDevice Functional Characteristics

1.3.1 Administration

- User , Database , Fault , System and Device Management

1.3.2 Protocol Definition

- Managing various protocols down to Bit level units
- Customizing Data Storage Objects and Methods through usage of protocol editor
- Device Interface possible in ModBus, Standard and user defined serial interface. I/O, DPRAM I/F, etc. also workable

1.3.3 Semantics

- Increasing user convenience / immediacy by composing communication agreements into meaningful scripts.

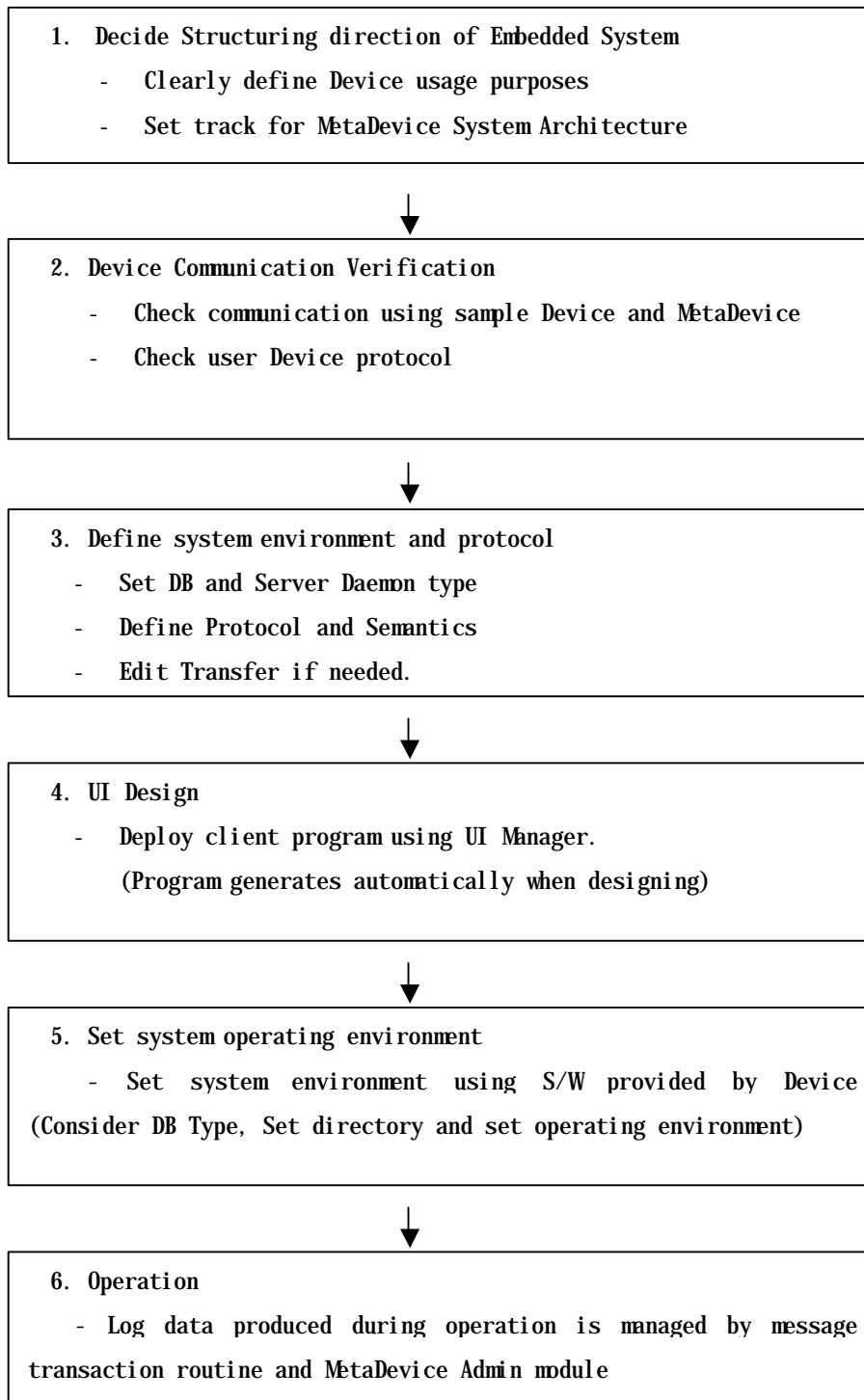
1.3.4 Transfers

- Integrative management of heterogeneous protocol from one Device possible
- Integrative management of heterogeneous Device protocol
- Protocol Exchange and communication possible between heterogeneous Devices

1.3.5 User Interface

- Composition by Java Beans Component
- Easy UI Design through Component's Drag & Drop function
- Automatic applet/application execution on deployment. Program Generation.
- Possible system formatting per user
- Extra programming unnecessary when formatting or converting system

1.4 Implementation Methods Using MetaDevice



2. Installing

2.1 MetaDevice Install Program Download

The MetaDevice Install Program can be downloaded from www.sena.com. Before downloading, your e-mail address must be registered. Once registered, you can download the program using your ID.

2.2 Installing

After downloading, doubleclick the Metadevice install Program. The install window will appear.



[Fig 1.5.1 Initial Install window]

Select "English" and press "OK".



[Fig 1.5.2 Second Install Window]

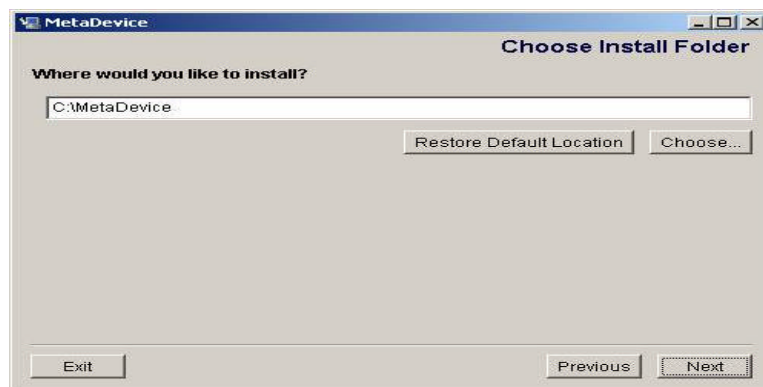
Press "Next" to continue installation.



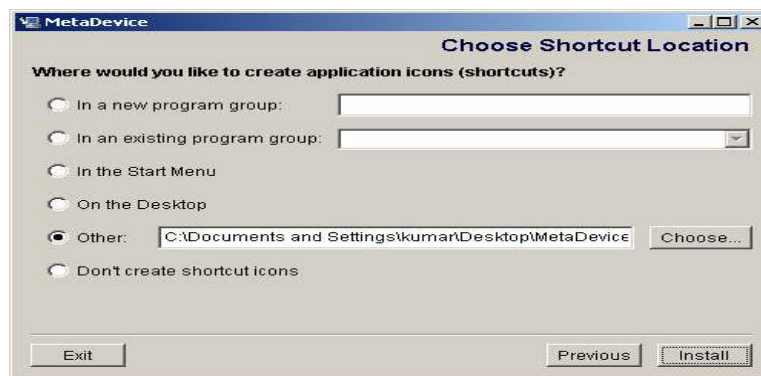
[Fig 1.5.3 License Agreement Window]

Read license agreement carefully before installing and press "Next".

If you do not agree with contents of the license agreement, press "No" or "Exit".



[Fig 1.5.4 MetaDevice Install Directory Selection Window]



[Fig 1.5.5 Short Cut Icon Location Selection Window]

Begin installation by pressing "Install".



[Fig 1.5.6 Install Completion Window]

After pressing "Done," click the "ServerManager" or "UIManager" icon on desktop. If the screen is displayed, MetaDevice has been installed normally.

3. MetaDevice Implementation Procedure

For step by step procedures, refer to Part IV Examples.

3.1 Server System Implementation Procedure

1. Creating Project Directory and DB Instance

- Compose Servers and UI Program Directories by using MetaDevice and create Instance by using DB related tools



2. ServerManager Operation and Admin Procedures

- Operate Server Manager and type/register General Properties
- Do Admin work after recalling MPR file from Admin Module
- **User Registration, Device Registration, User and Device Relation Registration** are most important in Admin work
- When you register a Device, be aware of **Device IP, Device Code, Semantics Name,** etc. Refer to Part II Section2. (Device depends on Run-Time License No.)



3. Protocol Editing and Registration

- Edit user Device protocol after Admin work. When you edit protocol, use Protocol explorer in the left hand screen of Server Manager.
- After registering “ **Protocol Name**” and “ Edit” registered protocol.
- When the protocol is registered, all “ **Protocol Symbols**” in the system are automatically created and managed in the MetaDevice system
- Protocol consists of “ **SEND Protocol**” (Server->Device) and “ **RECEIVE Protocol**” (Device->Server).



4. Semantics Editing and Registration

- After working on Protocol, define how data should be communicated while editing Semantics.
- When editing Semantics, use Semantics explorer on the left window of the Server Manager.
- After registering “ **Semantics Name,**” “ Edit” the registered Semantics.
- “ **Semantics Symbol**” is managed automatically. The “ **Semantics Name**” must be recorded in the Semantics Name Field of the Admin Device Registration Table.
- Semantics Method basically utilizes SEND() and RECEIVE(). SEND() uses one Protocol Name , while RECEIVE() can use multiple Protocol Names.



5. Transfer Editing and Registration

- It is not necessary to use TRANSFER() Method in Semantics if it is not needed.
- Transfer is edited as to enable the user to edit TRANSFER Method when it can not be implemented by Semantics Logic.
- Transfer Name is used as a factor of Semantics TRANSFER() Method.
- Transfer supports various arithmetic operations, casting operations, process conditions, Protocol exchange and transfer.
- Transfer has the following function

- . **Self Protocol exchange** : when managing data with multiple Protocols in one Device transfer/integration to a common protocol is possible.
(Transfer_Self)

- . **Protocol transfer** : When using heterogeneous protocol devices, data transfer through identical content is possible

The user can change protocol data and transfer to other devices. In this case, data is transferred based on the Server's polling time, but in semantics frame polling, the most recent data is transferred to the heterogeneous device.

protocol data within Semantics Frame polling time can be overlapped.
(Transfer_MAC)

- . **Instant protocol transfer** : Data loss is possible in case of " Transfer_MAC." To avoid this, user can use instant protocol transfer for Persistence Transfer.

- For more detailed information, see Part II Section 5 and 6.2->3)->(3.3)



6. Server Program Deployment

- Save completed work by selecting " Save" form the Server Manager's " File" menu. The file will be saved " Project_Name.MDPR" .
- The MDPR File is important data managed together with the MDU File composed in UI Manager. Therefore it must be managed with care.
- If the " **Make Server**" item from the " Deploy" menu is selected, the Server Program is deployed automatically.
- Server Program is created in the " Project_Directory\Server" Directory created in procedure 1 and the execution file name is " **Project_Name.bat**"
- To run the Server Program, doubleclick the .bat file.

3.2 User Interface Implementation Procedure

1. Executing UI Manager

- Click the UI Manager ICON on the desktop or the MetaDevice folder.



2. Reading Device Information

- Click “ Device Information” Tab in the upper left Panel.
- Select “ Get Device Info.” among the four Buttons displayed.
- Select MPR file created in the Server Manager from the MPR selection window displayed.
- Because each user needs to edit his own system, input user ID registered in the database as followed in Server Implementation Procedure 2.
- By inputting your user ID in the user ID Dialog window, you can build the user system by referring to the UI Manager about Device information(Mac Number, Related Protocols), and construct Applet or Application System after editing UI.



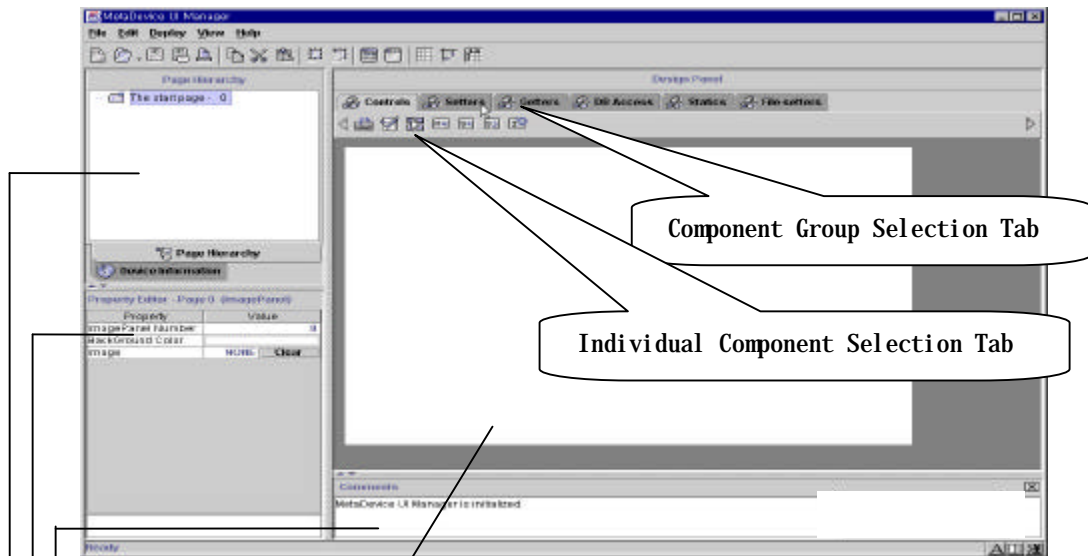
3. Component Building and Deployment

- After selecting “ Getters” from Component Group Tab and selecting one of the desired components, click the left mouse button on the design panel to designate location. Component will be formed. Refer to [Fig 1.6]
- If the Component in the Design Panel is double clicked related Property List will be displayed in the lower left Property Panel.
(For more information on Property, see Part III UI Manager)
- If it is difficult to input Property after inputting the corresponding Property, do not set up Property but click “ Deploy” in the UI Manager menu and select “ Make Application” .
- Select Directory where Application program will be created and deploy UI program Maintain IP “ 127.0.0.1” .



4. Program Execution

- Execute Server program (.bat file created in the Server Manager).
- Execute Client program created from UI Manager (MyApplication.bat file) and input User_ID / PassWord to see UI created.



[Fig 1.6 UI Manager Initial Window]

Initial Window Description

? Design Panel

Panel enabling user GUI using selected Component. Appears when running applet or application program

Page & Device Information Panel

Panel that establishes multiple pages to allow movement from page to page and allows user to view Server Data information (Protocol, Mac Number, etc.)

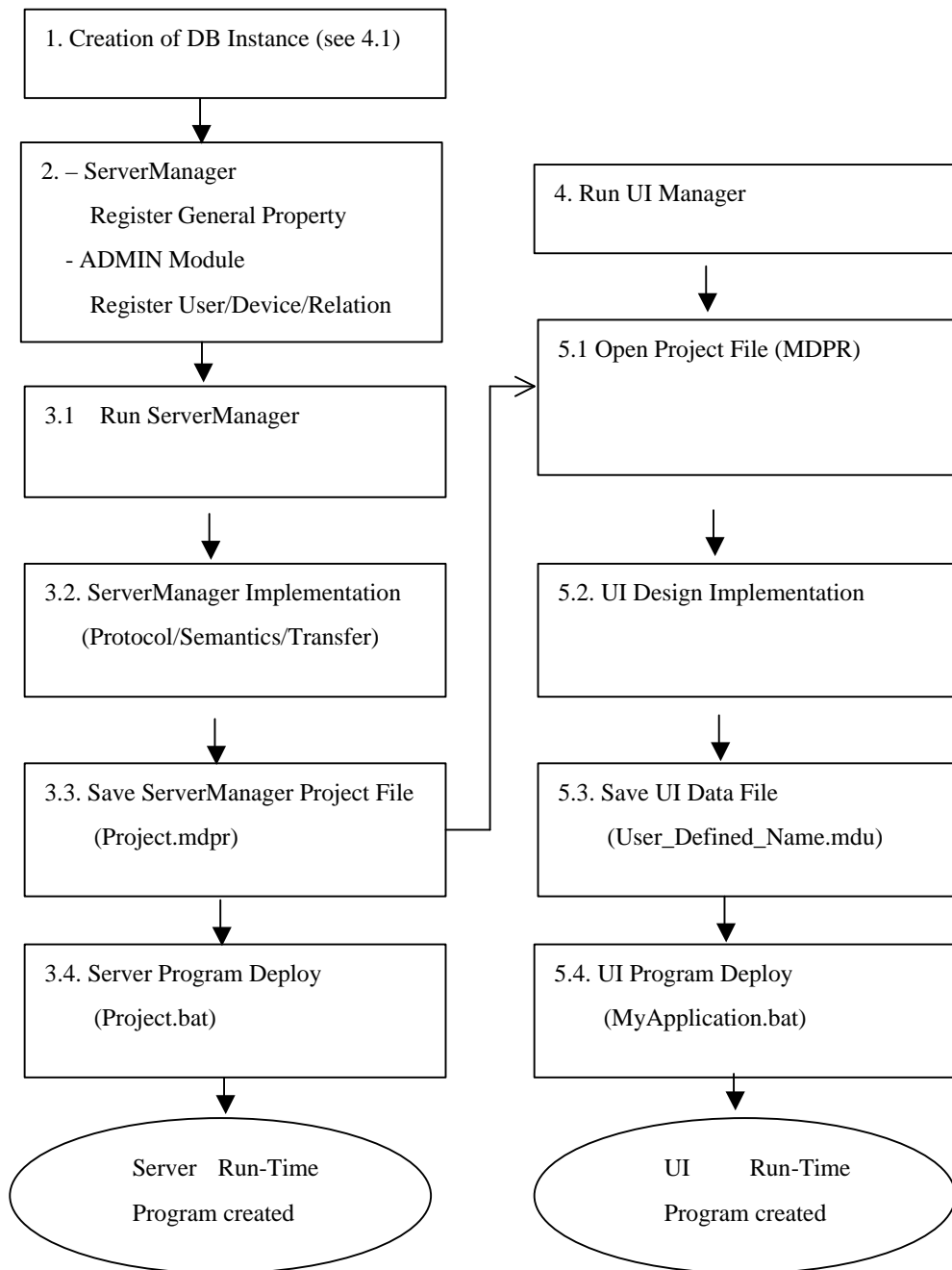
Property Editor Panel

Panel that allows input of communication Property(Protocol, Mac Number, etc.) and static Property(Color, Size, Font, etc.) when writing each Component. Various Property content can be established depending on component type and characteristic.

Message Panel

Messages and Server Protocol data can be viewed while using UI Manager.

3.3 Implementation Procedure Summary

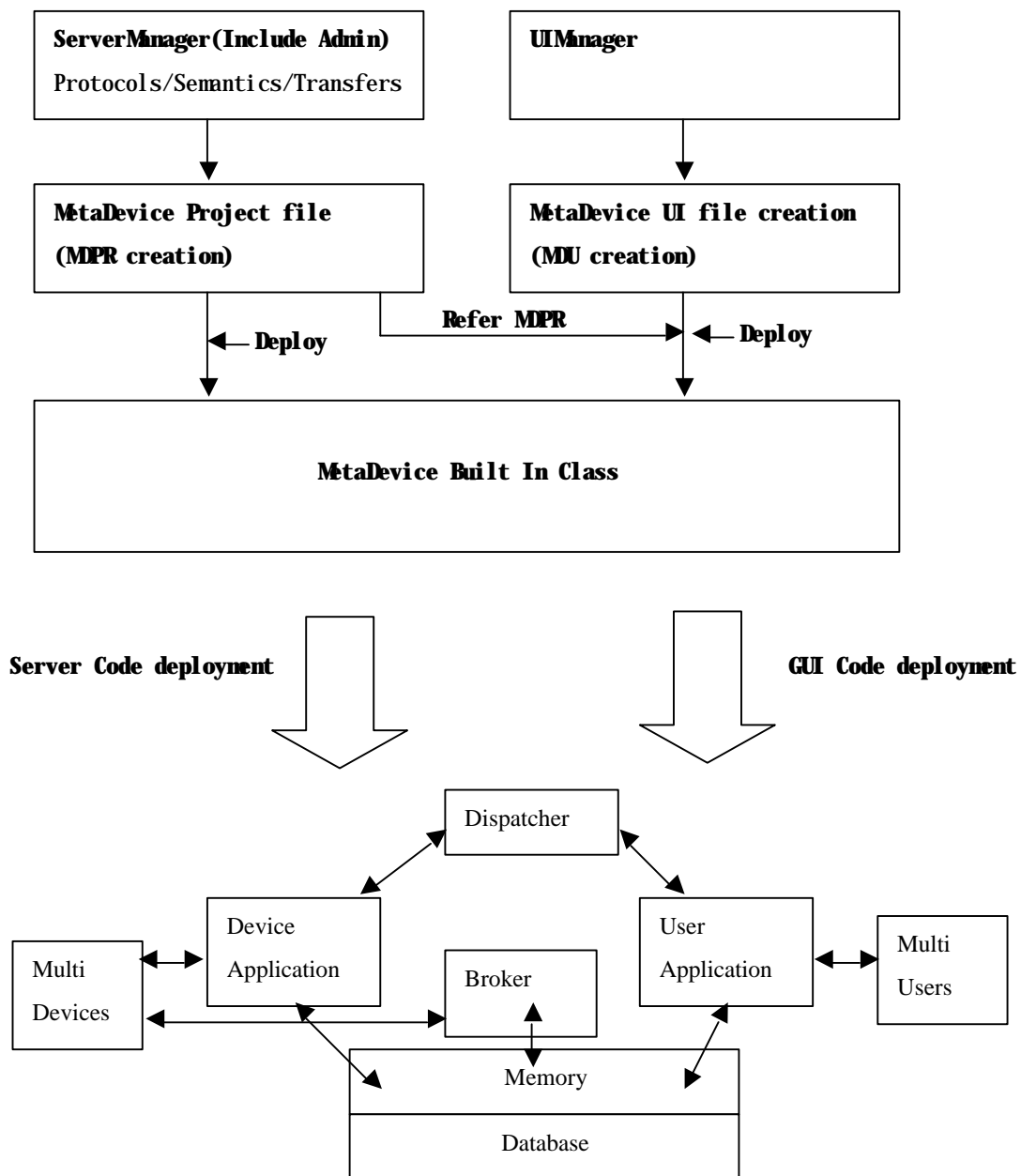


[Fig 1.7 Summary of Implementation Procedure]

Note1) In Procedure No.2, the Admin task can be done before deploying UI program or running Server Run-Time Program.

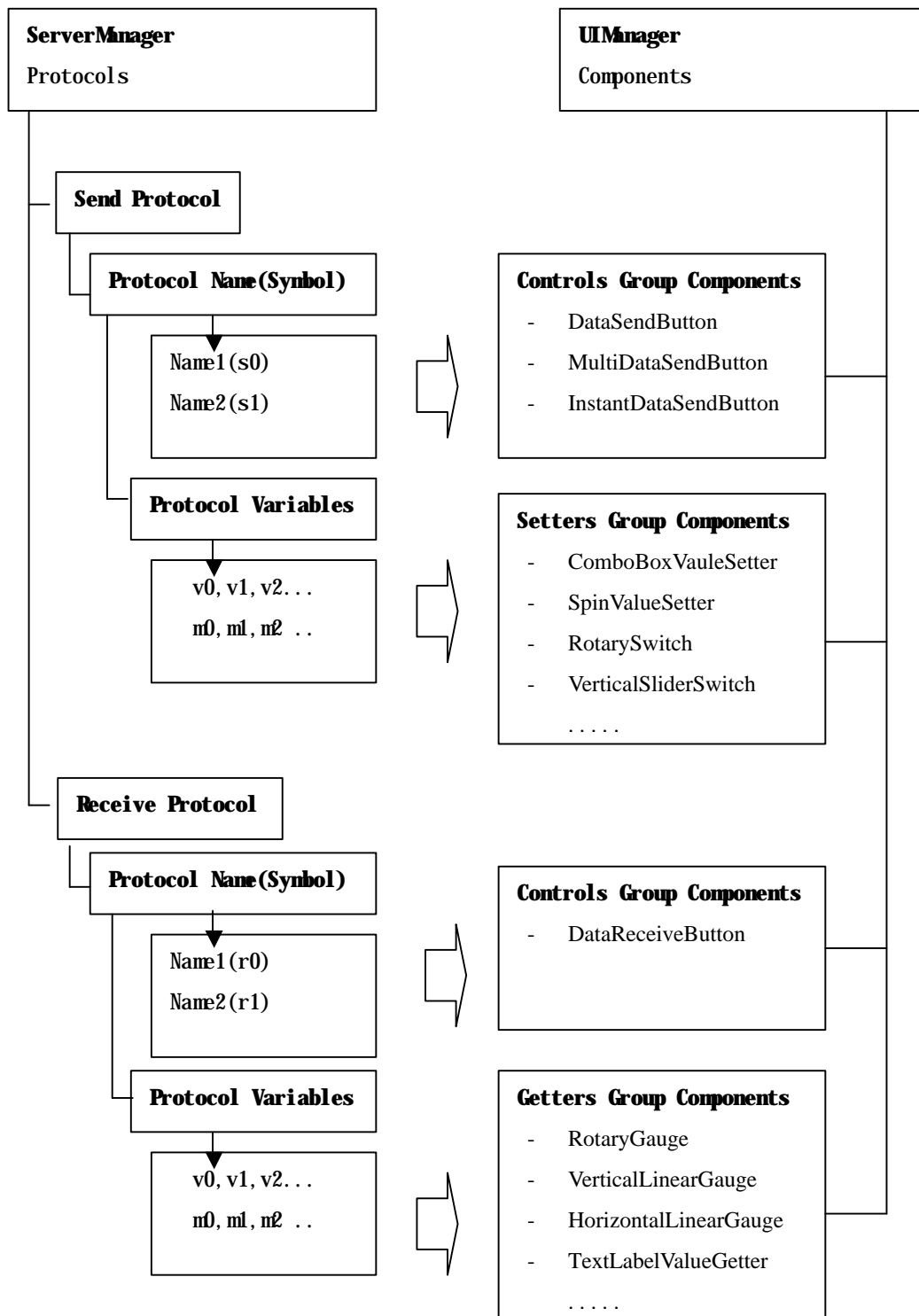
Note2) For the user's Device Segmentation in Procedure No.5.1, read MDPR file and user information before deploying UI Component. The opening job in Procedure No.5.1 can be done before deploying UI program.

3.4 Code Generation Procedure



[Fig 1.8 Code Generation Procedure]

3.5 Data Interdependency



[Fig 1.9 Data Interdependency]

Part II Server Mnager/Admin

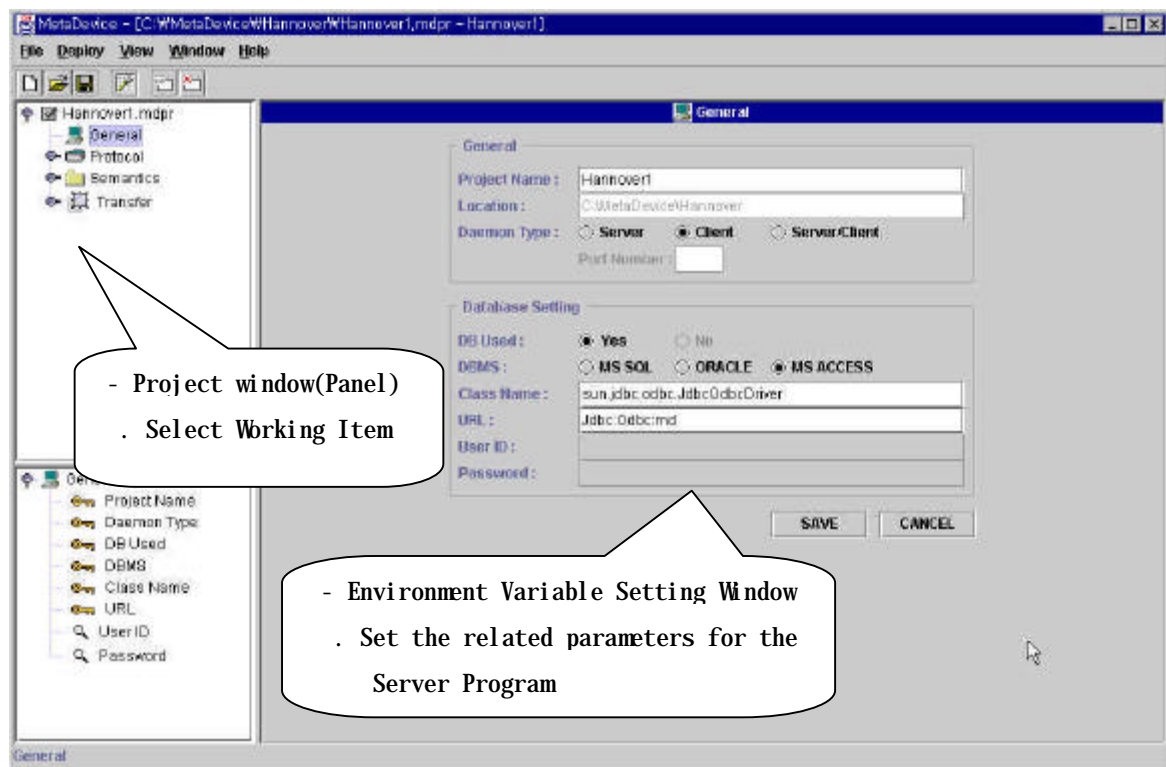
1. Overview and Basic Functions of Server Manager

The Server Manager is used for future deployment of Server Programs and sets various condition parameters of the Server Program including Program Type, Polling Condition, Data Process Condition, Device / Server Communication Method and User / Server Communication Method. The basic functions of the Server Manager is summarized as follows.

1.1 Setting Server Environment Variables

Editing or inputting relevant parameters to the Server Manager's General Panel sets server environment variables.

[Fig 2.1] Relevant parameters are as follows.



[Fig 2.1 Setting Environment Variable]

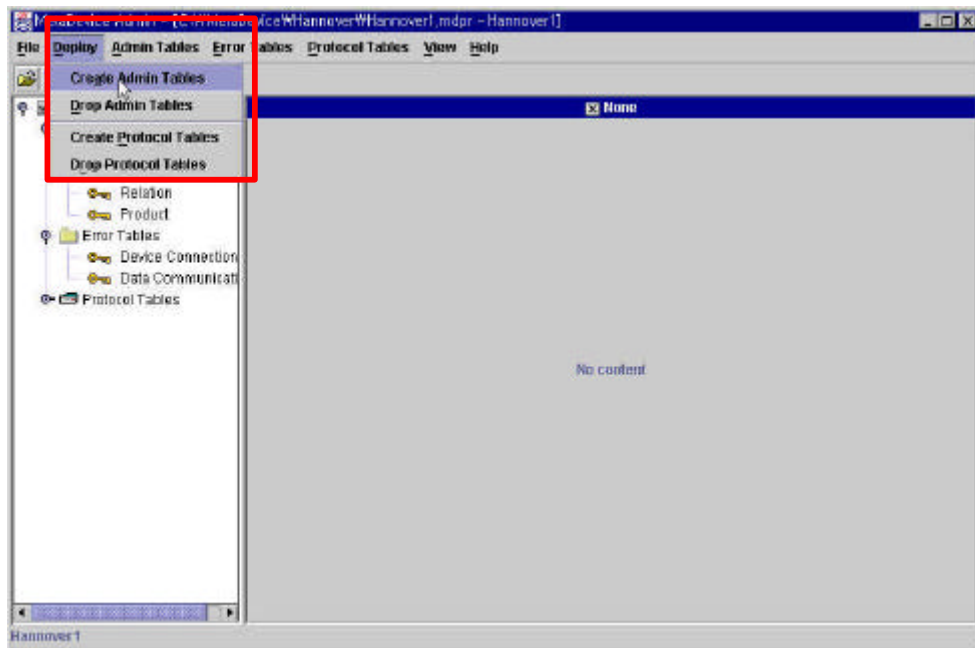
- **Project Name** : All data edited in the Server Manager is saved as Project File(MDPR File). In other words, it is saved as Project Name.mdp.
- **Location** : Select directory where Project File and Server Program are to be saved.
Server Program is created as Project_Name.bat file in "Location_Dir\Server"

- **Daemon Type** : Daemon Type sets the Server and Device communication method.
 - . Server : Device is operated by Client while Server awaits Device connection to begin communication.
 - . Client : Device is operated by Server while the Server periodically connects to Device for communication.
 - . Server/Client : Set type when Device is operated by Server/Client or when separate Devices are operated by Server/Client.
- **Port Number** : Port Number needed to connect to Server when Device is operated by Client or Server/Client. Must also be set in Device when installing Device.
- **DB Used** : Asks whether Database will be used. Always select "Yes" because User and Device management is needed.
- **DBMS** : Select type of Database. Since Version 1.3 does not support "Oracle" select "MS-SQL" or "MS Access".
- **Class Name** : Select Database Connection Driver. In case of using MS-SQL Database , select Microsoft JDBC Driver.
- **URL** : Select Database Instance. Please refer to Database references for creating Database URL. In case of using MS Access, use "DSN" Name.
- **UserID/Password** : If you use MS-SQL Database or Oracle Database, set UserID/Password of Database.
When Parameter Setting is completed, click "Create" or "Save" button to create or save Project File.

1.2 Database Admin Functions

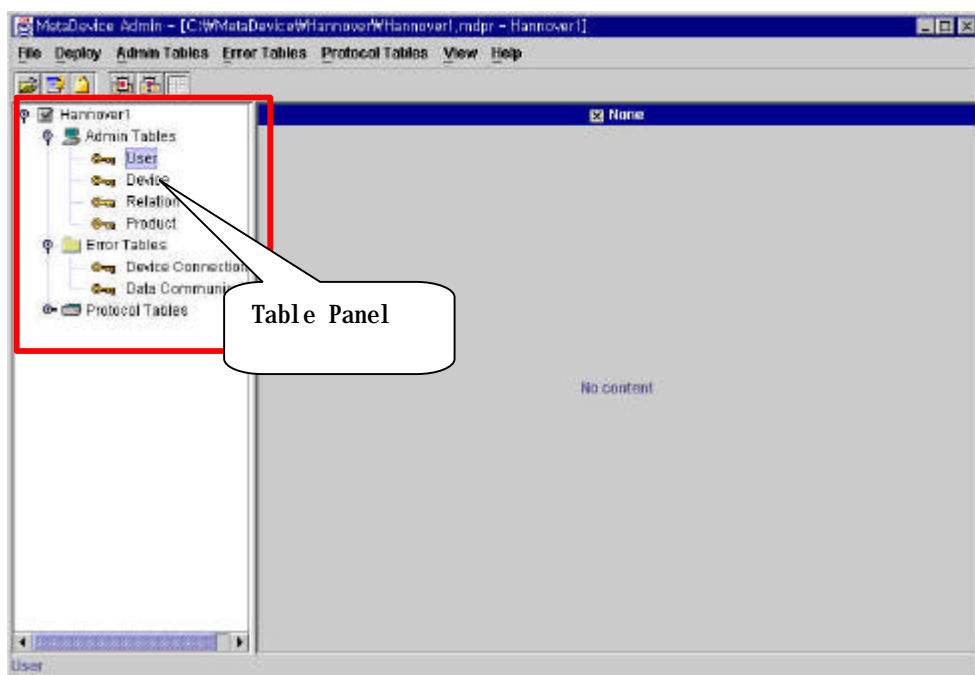
Admin functions include "User management", "Device management", "Relation management(sets User and Device relation)", "Error management", "Protocol management", "Device Product management", plus Excel File creation functions.

The most important job in initial system operation is to edit the basic Admin Table in the selected Database. Simply select "CreateAdmin Tables" as in [Figure 2.2.1] in the Admin Module. Be sure to bring relevant MDPR Files.



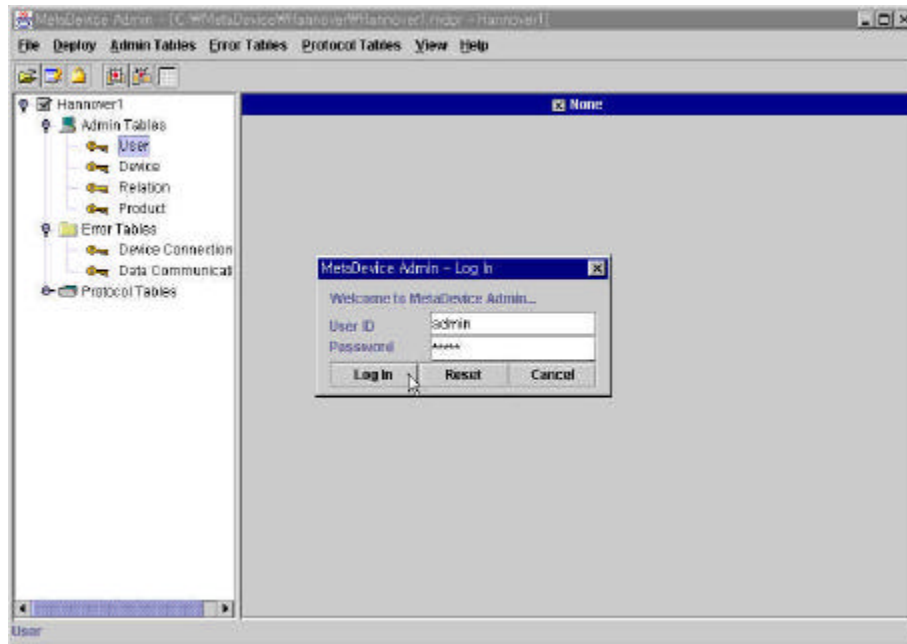
[Fig 2.2.1 Creating Admin Table]

Once the Admin Table is created, select Admin Table in DB Table Panel and begin Admin registration.

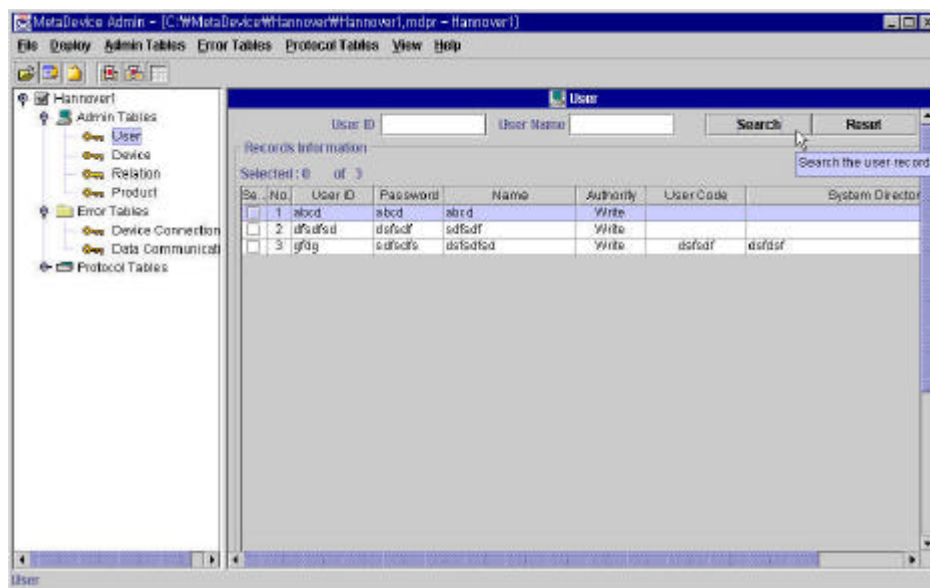


[Fig 2.2.2 Admin Tables]

To register Admin data, user should login with Admin User ID and Password.
The Default User ID and Password is admin/admin.



[Fig 2.2.3 ID/Password Input]



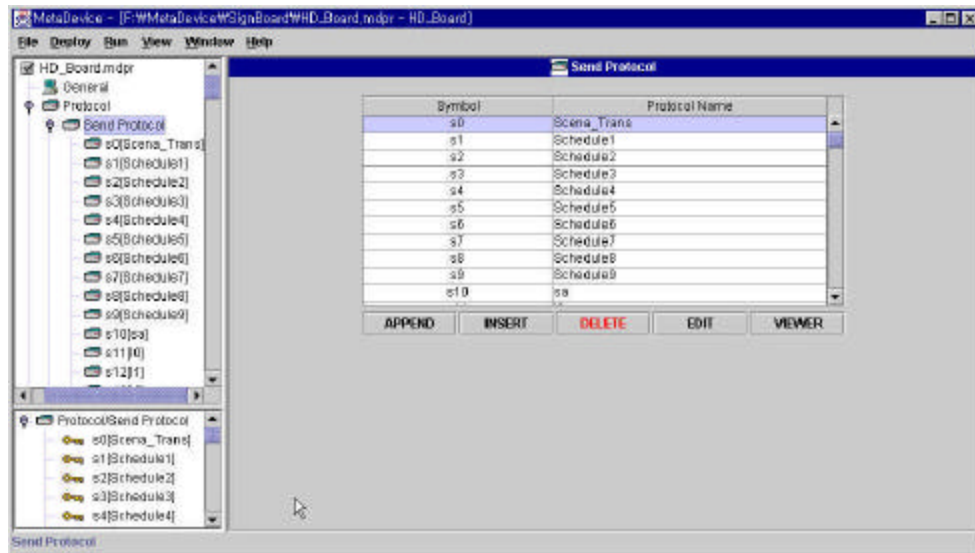
[Fig 2.2.4 Admin Table Window]

After Admin Login, [Fig 2.2.4] window is displayed.

[Fig 2.2.4] is the window displayed after logging in and selecting "User" in the Table panel. For more detailed information about Admin, refer to section 2 "Admin Program."

1.3 Protocol Editing

After concluding Admin operation, select “Protocol” in the upper left Project window[**Fig 2.1**]. Protocol registration and List window will be displayed as in [**Fig 2.3**]. Protocol consists of two types: SEND Protocol and RECEIVE Protocol. The SEND Protocol is the command protocol from Server to Device while the RECEIVE Protocol is the reception Protocol from Device to Server.

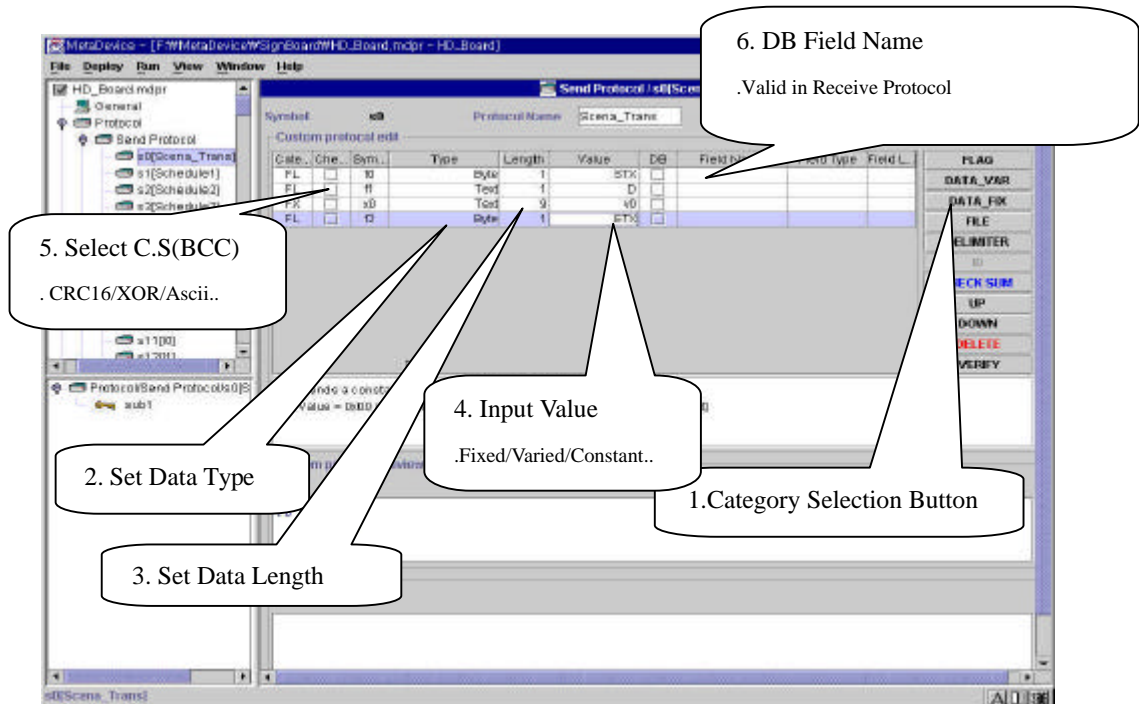


[**Fig 2.3** Protocol Name Registration Window]

When registering the Protocol Name, click “Append” or “Insert” in the registration window. The Protocol Symbol is automatically created for management in the System, but the user defines the Protocol Name.

When editing Protocol data, double click the relevant Protocol item or click “EDIT”. Protocol Editing/Creation window is as [**Fig 2.4**].

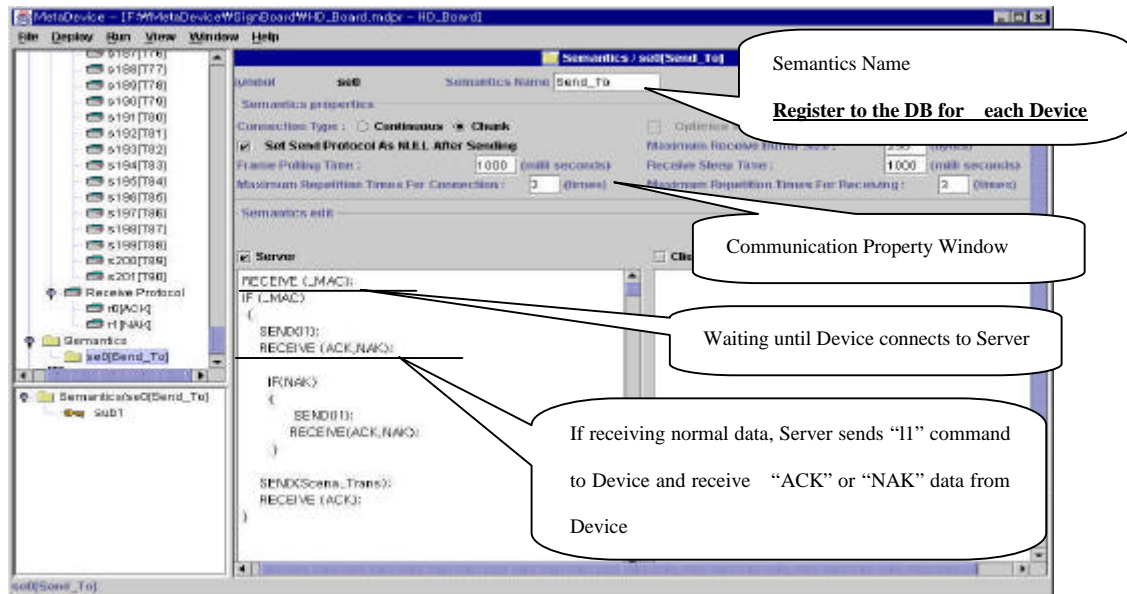
For detailed information about editing/creating protocol, refer to Section 4.



[Fig 2.4 Protocol Edit Window]

1.4 Semantics Editing

Semantics meaningfully defines how and what kind of Protocol Data the Server and Device will exchange. User can define Semantics using SEND() Method and RECEIVE() Method. SEND Method is how the Server sends Data to the Device while RECEIVE Method is how the Device sends to Server command with receive protocol data. [Fig 2.5] shows one example when the Server's Daemon Type is the Server.



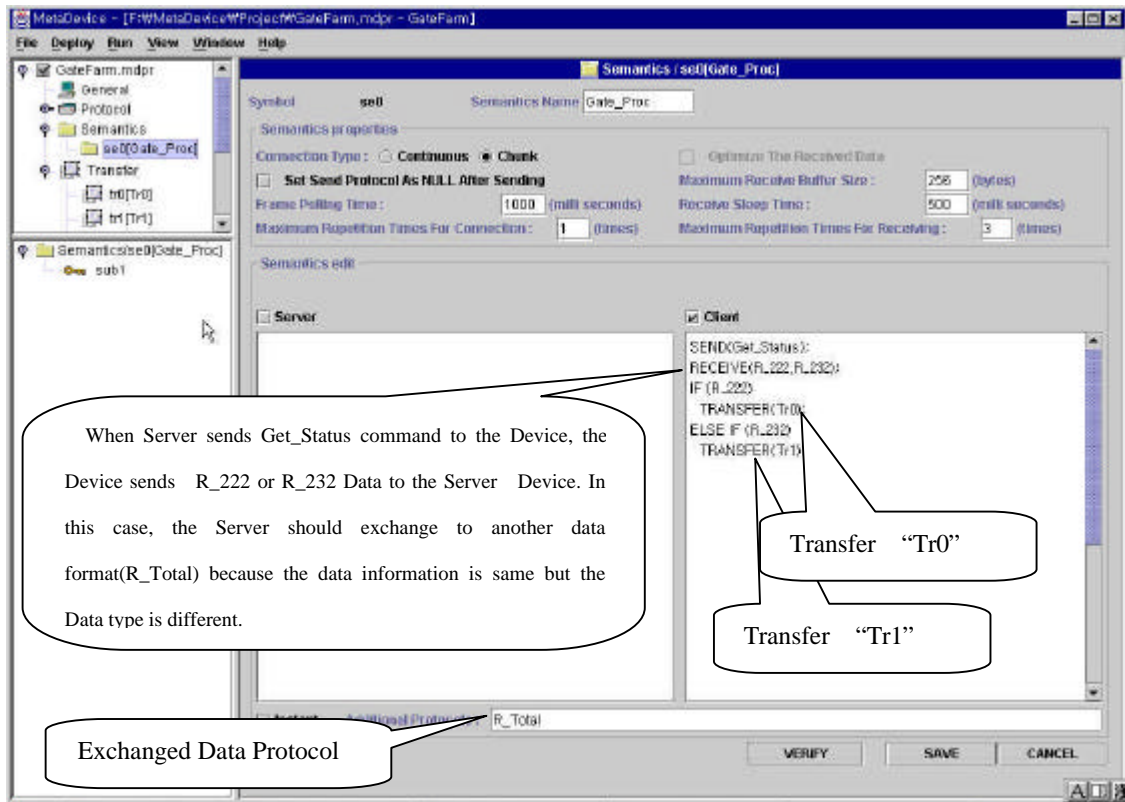
[Fig 2.5 Semantics Editing Window]

1.5 Transfer Editing

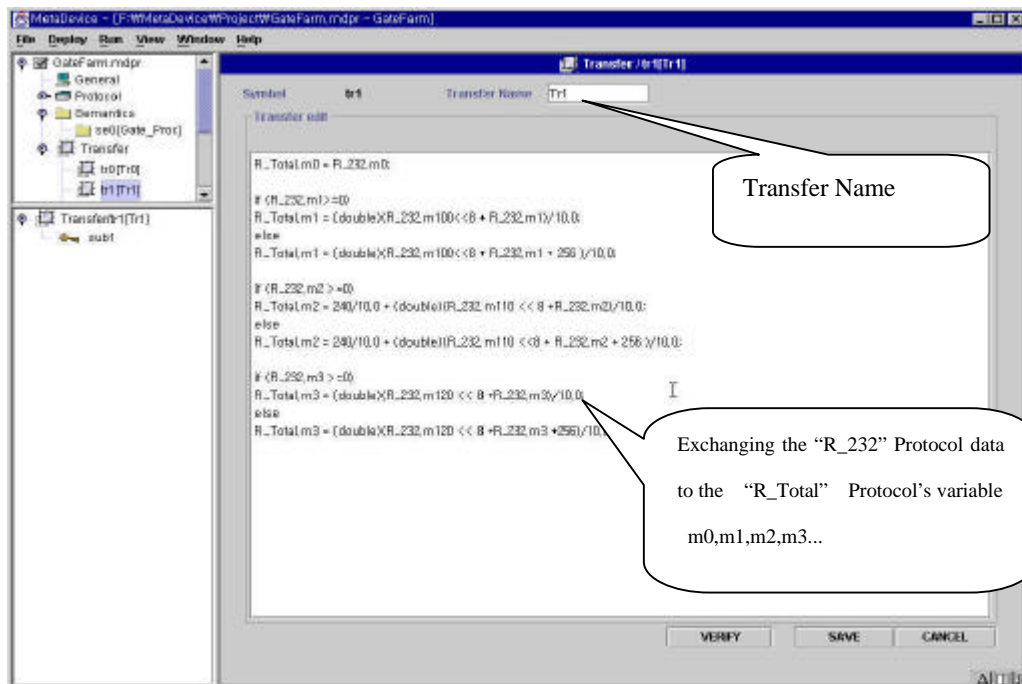
When Semantics Logic cannot do Transfer implementation, MetaDevice allows the user to make implementations. The Transfer Name is used in the Semantics TRANSFER() Method argument.

Transfer supports the arithmetic operations, casting operations, process conditions, Protocol transfer or exchange and contains the following applications.

- . **Self Protocol Exchange** : When the user is managing Data with multiple protocols within one Device, Transfer/integration using one Protocol is possible. (Transfer_Self)
- . **Protocol transfer** : When user is using heterogeneous protocol devices, the user can change protocol data and transfer to other devices. In this case, the Server transfers data according to polling time. The protocol data within Semantics Frame polling time can be overlapped. (Transfer_MAC)
- . **Instant protocol transfer** : Because Protocol transfer only sends the most recent data sent through polling time, prior data sent has the possibility of becoming lost. To avoid this, utilize Persistence Transfer for instant protocol transfer.



[Fig 2.6 Semantics Window including Transfer Method]

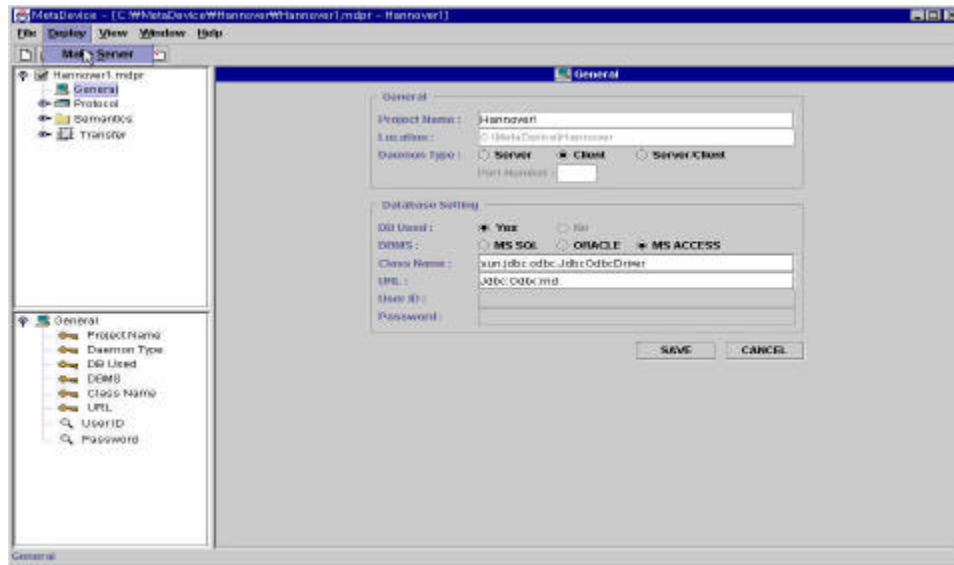


[Fig 2.7 Transfer Edit Window]

1.6 Server Program Deployment

The Server Program can be deployed by selecting “Make Server” from the Server Manager Menu. When doing so, the user must consider Protocol Table generation.

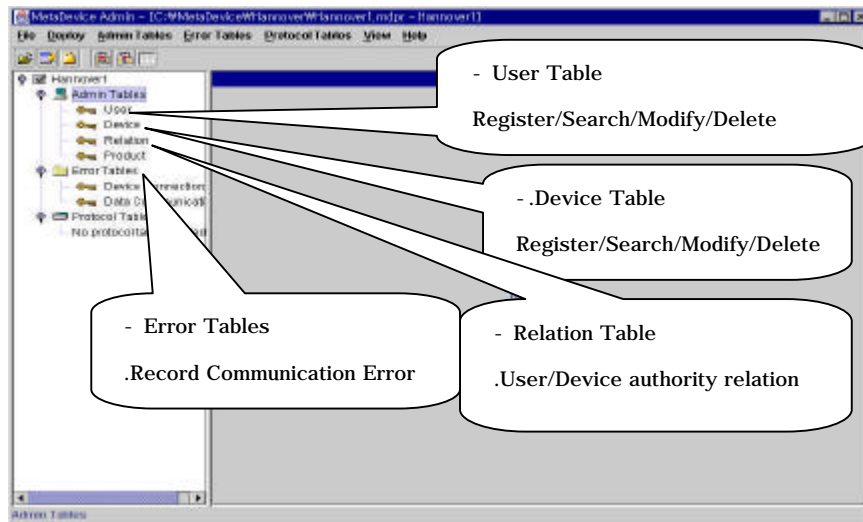
You can create a Protocol Table using “Create Protocol Tables” from the Admin Module after editing Protocol and Semantics or just before running the Server Run-Time Program. But you do not need to create Protocol Tables if there are no DB relations with the Protocol Data.



[Fig 2.8 Server Program Deployment Window]

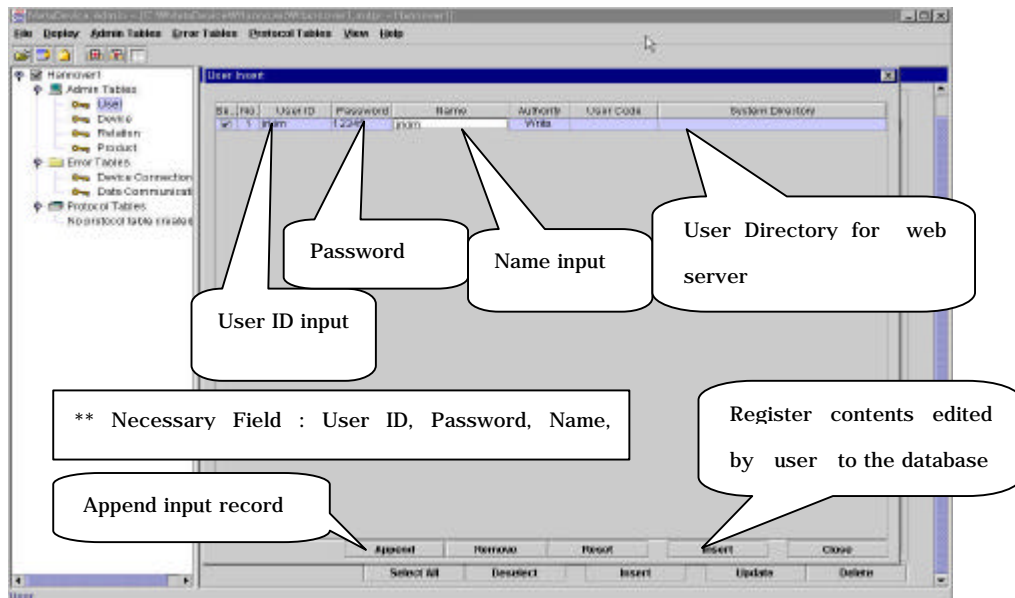
2. Admin Module

Admin Module registers/manages base information for smooth operation of the System. Mandatory items that need to input before deployment of UI Program are **“User Registration”**, **“Device Registration”**, and **“User/Device Relation Registration”**. Product Table”, “Error Data Table ”, “Protocol Data Table”, “Excel File Creation” options are also available.

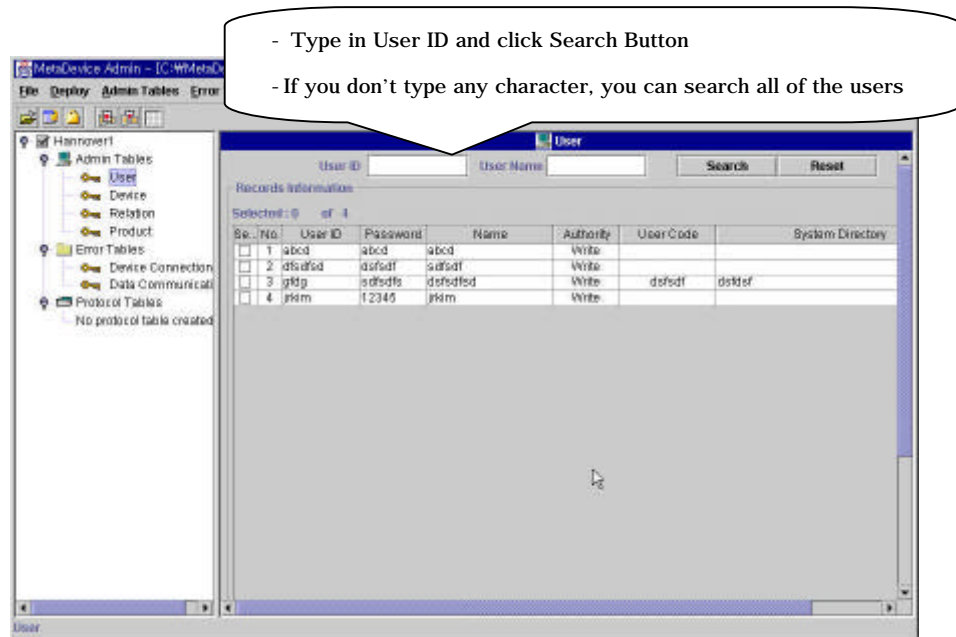


[Fig 2.9 Admin Initial Window]

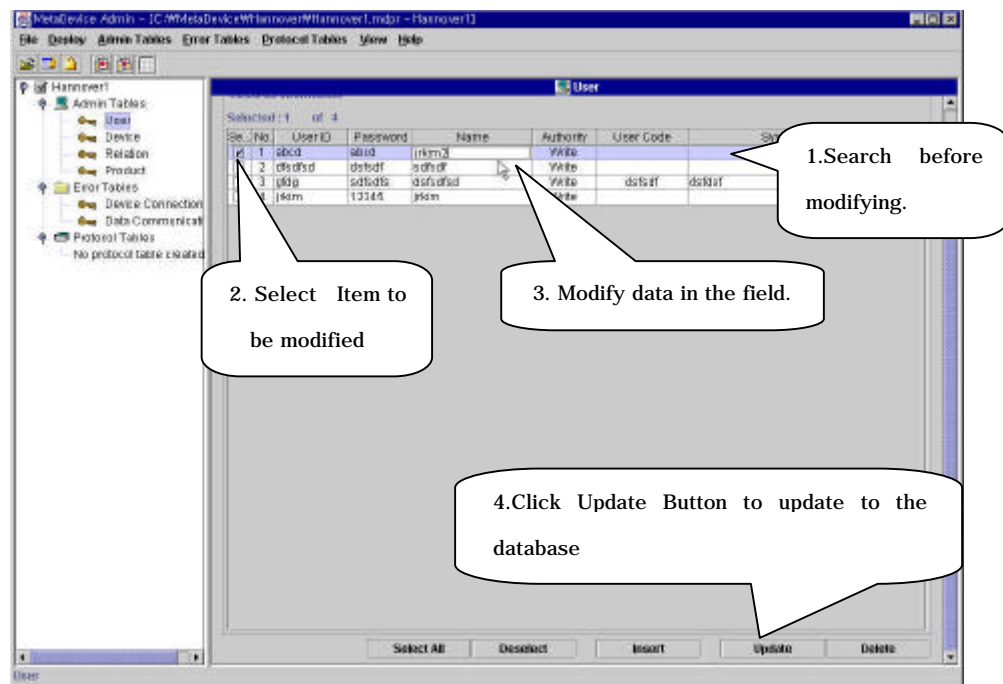
2.1 User Management



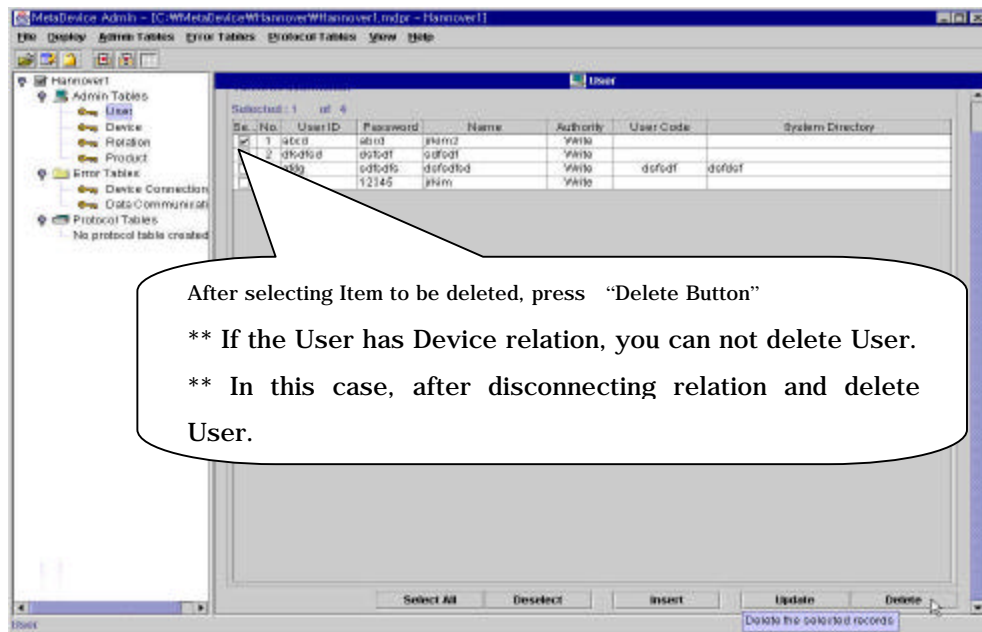
[Fig 2.10.1 User Registration Window]



[Fig 2.10.2 User Search Window]



[Fig 2.10.3 User Information Modifying Window]



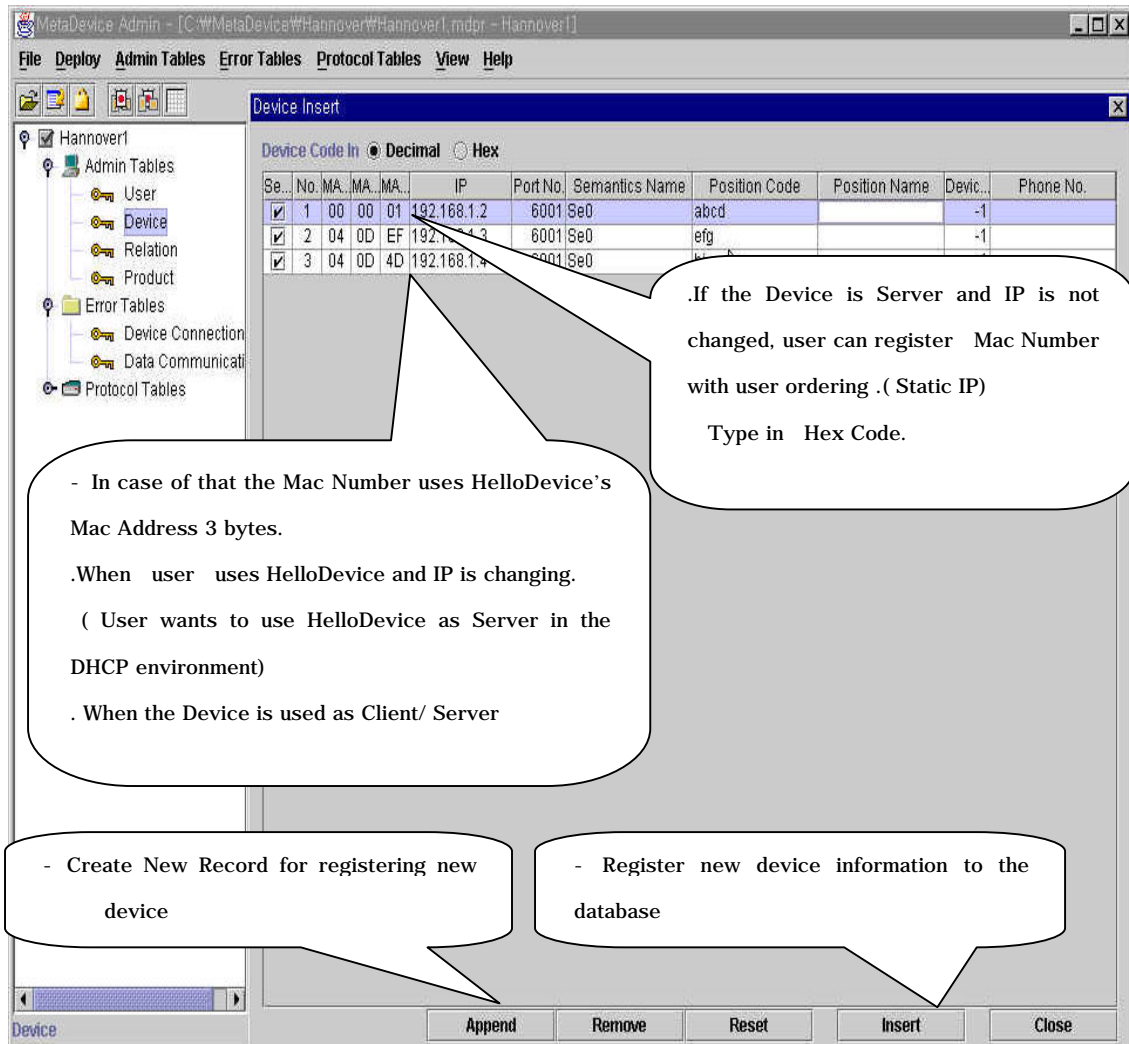
[Fig 2.10.4 User Delete Window]

2.2 Device Management

Because the Device communicates directly with the user's system it is necessary to precisely register relevant information in order to manage data efficiently when operating the system. It is also important for the user to understand a few important concepts, such as Uniqueness and Consistency. Since the usage and location of the Device or related IP information will eventually be changed at some time, a fixed set of base information is needed.

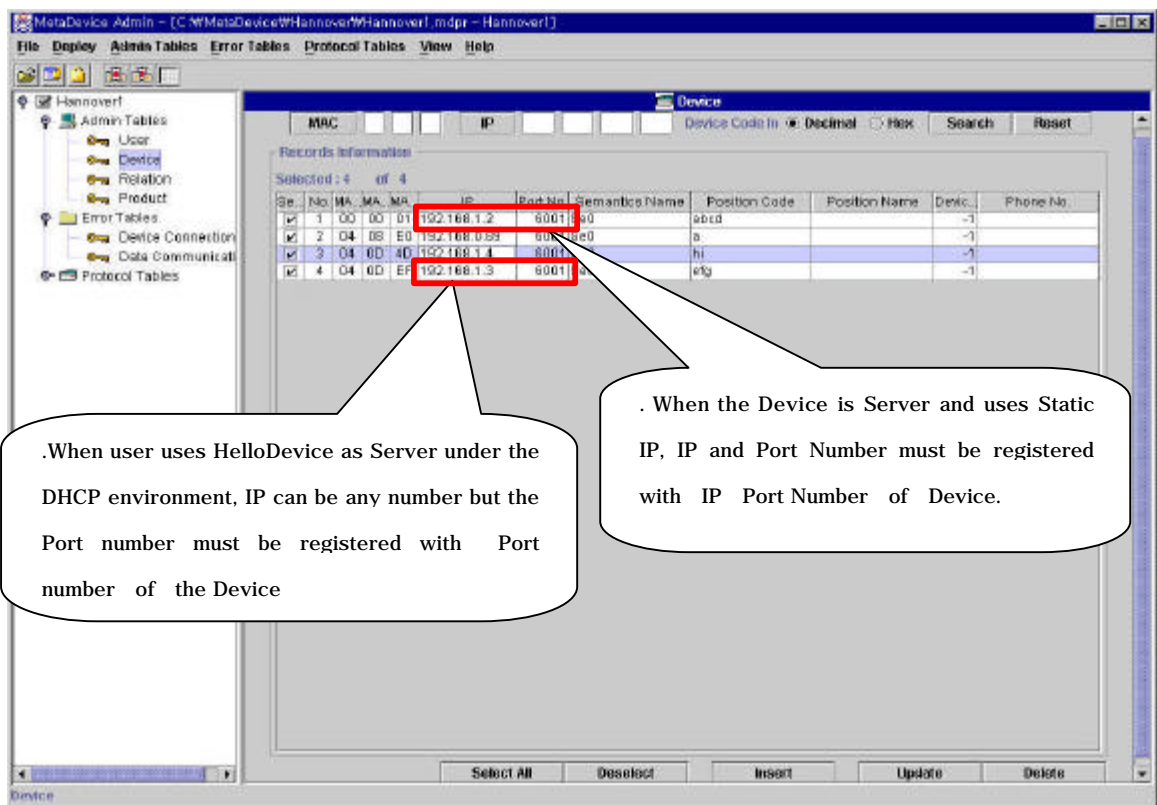
- **System Uniqueness** : The sole base information that is confirmed within the System, the MetaDevice has introduced the MAC Number concept, because most TCP/IP devices possess unique Mac Addresses. However, establishing all Mac Addresses for System Uniqueness may cause memory shortages and lack of system performance. So the solution has been to introduce Mac Numbers of 3 bytes with capacity of $256 \times 256 \times 256$, allowing management as well as ensuring uniqueness of approximately 17million items.
- **User Device Uniqueness** : The user will have difficulty in distinguishing the Mac Number managed by the system. The solution has been to introduce Position Codes and Device Codes to establish User Device Uniqueness. The Position Code and the Device Code establishes User identification by mapping the actual user device number. The Device

can manage up to 4byte integers, while the Position Code can manage up to a maximum of 50 characters, therefore enabling mapping of any type of device to establish user device uniqueness.

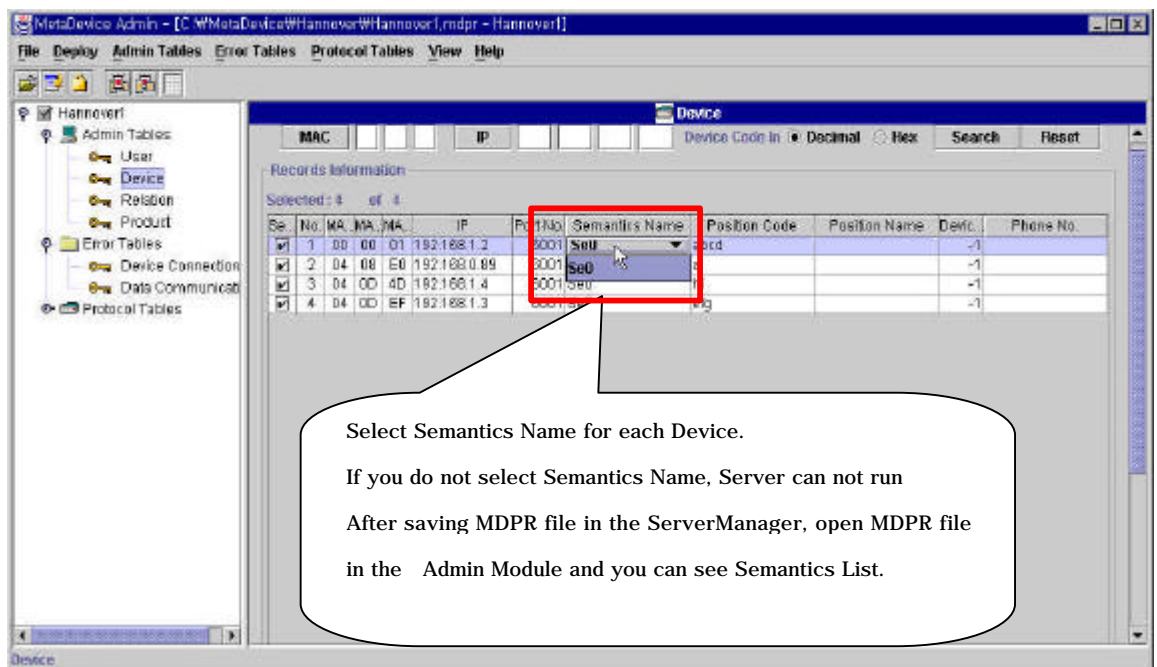


[Fig 2.11.1 Device Register Window 1]

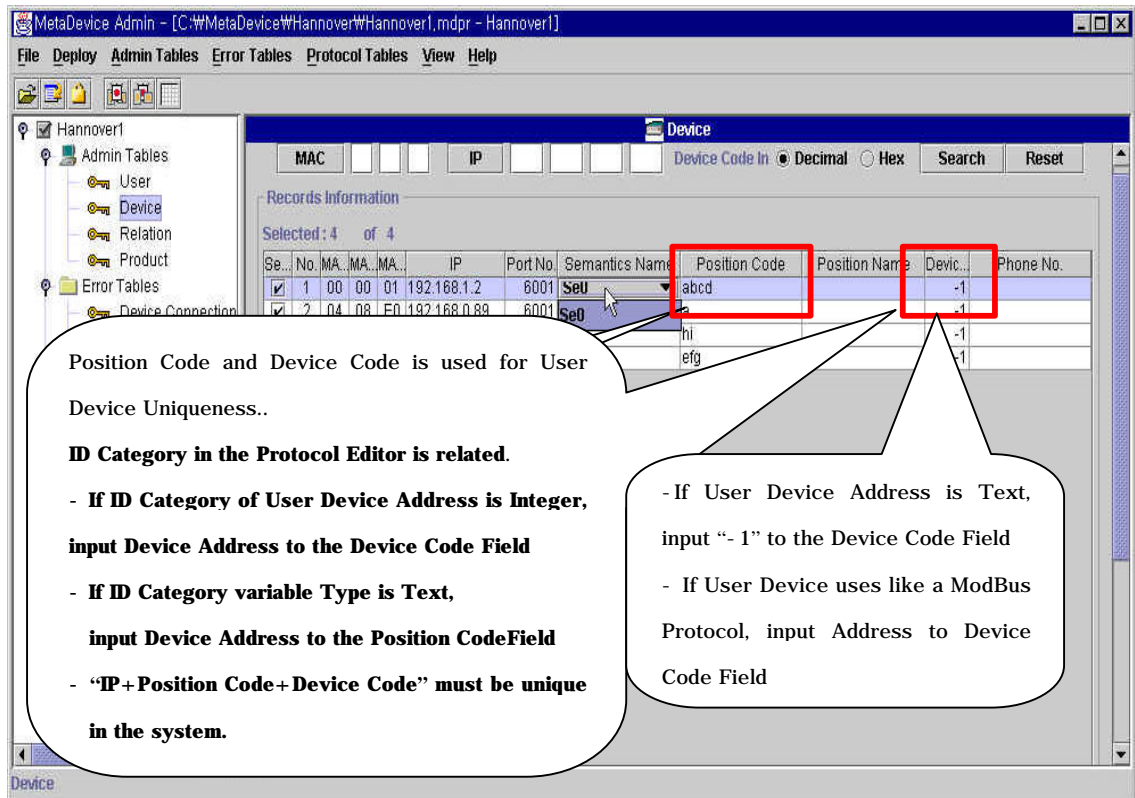
According to the figure above, if you use HelloDevice, administer the HelloDevice Mac Address by using the latter 3 Bytes as the Mac Number. If the Mac Address is "00:01:95:04:0d:ef", record "04,0d,ef" as the Mac Number. If you use other TCP/IP products, record their consecutive unique numbers. In any case, the Mac Number must be unique, and should not be overlapped. In other words, System Uniqueness can be guaranteed by the Mac Number.



[Fig 2.11.2 Device Registration Window 2]

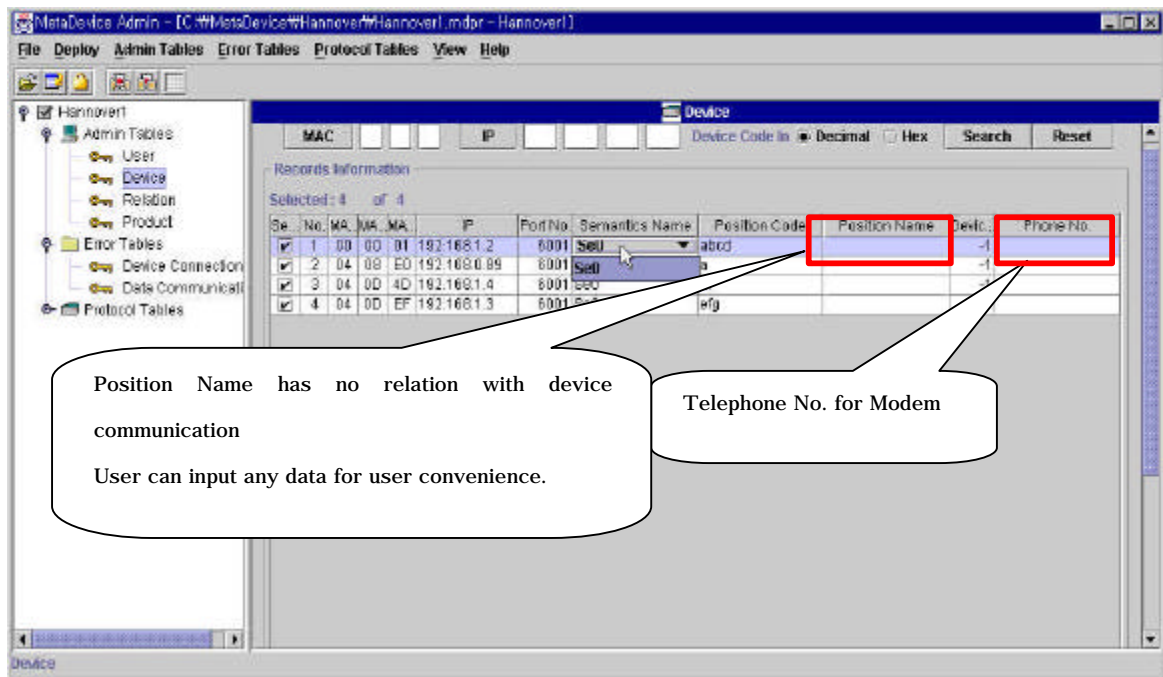


[Fig 2.11.3 Device Registration Window 3]



[Fig 2.11.4 Device Registration Window 4]

Note : If using the Position Code for User Device Address, always input "-1" as the Device Code Field Value. When the Device Address is an Integer like a ModBus, a new Device Address must be input in the Device Code.

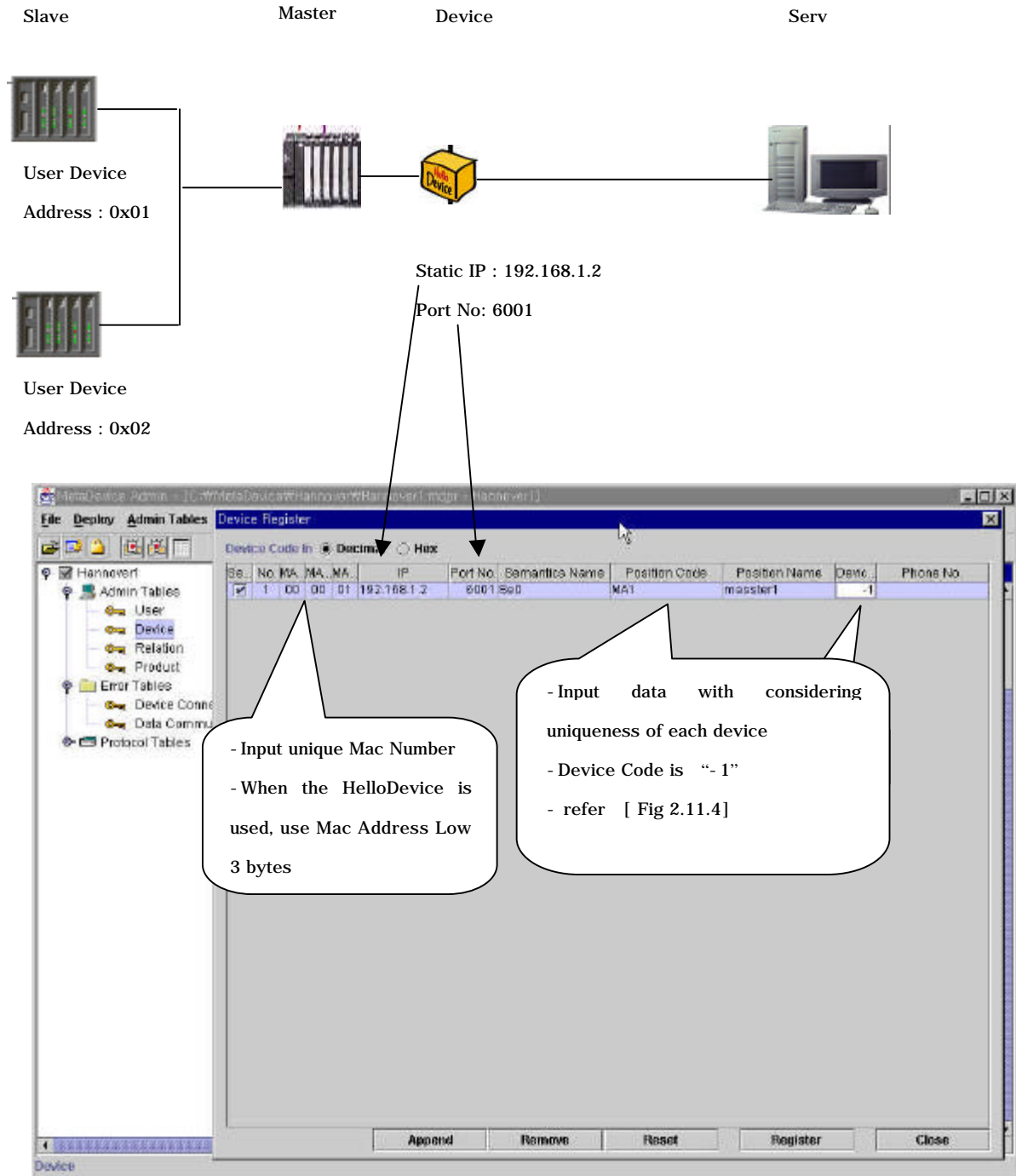


[Fig 2.11.5 Device Registration Window 5]

2.3 Example of Device Management

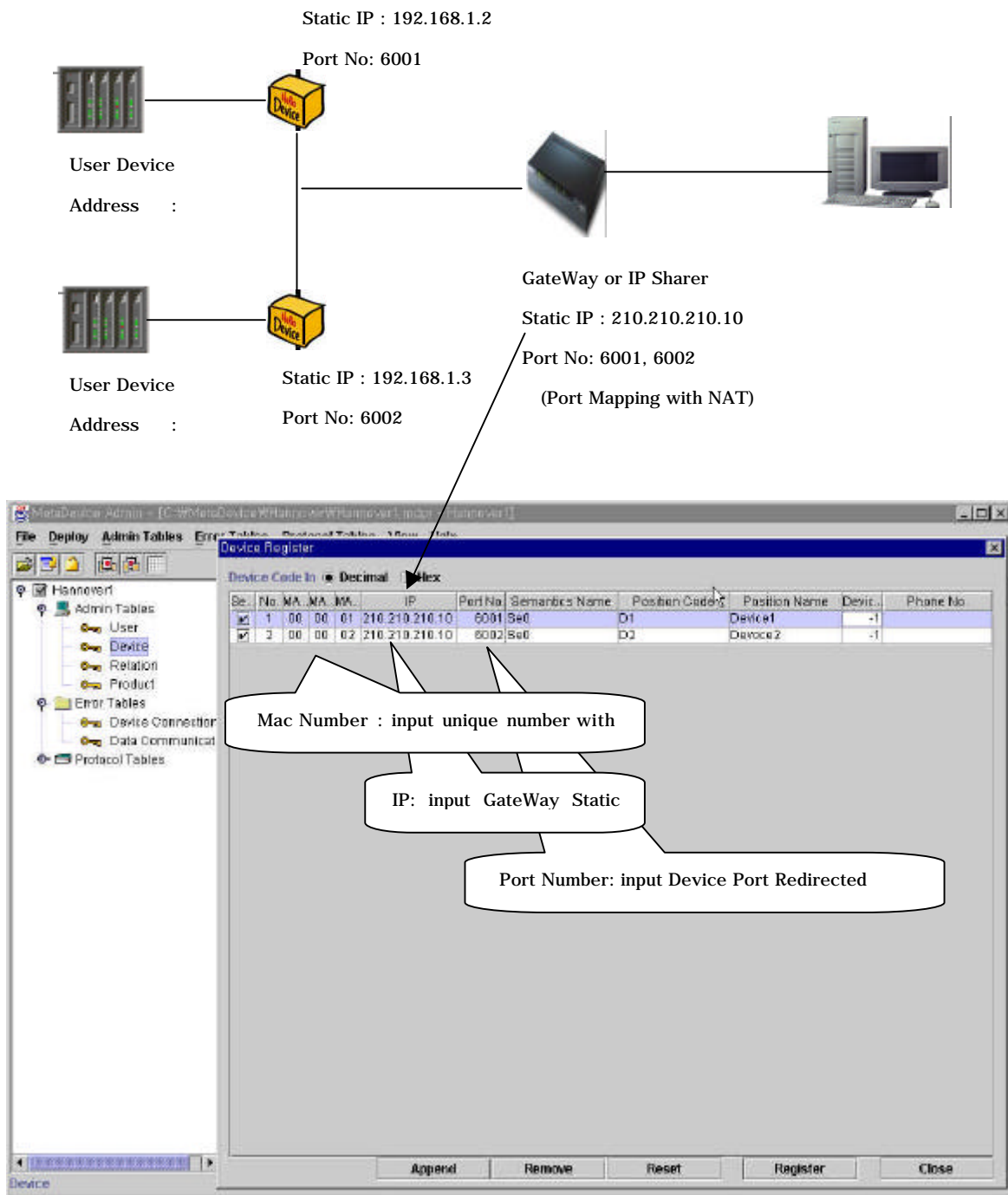
(1) Device is used as Server

- Case 1



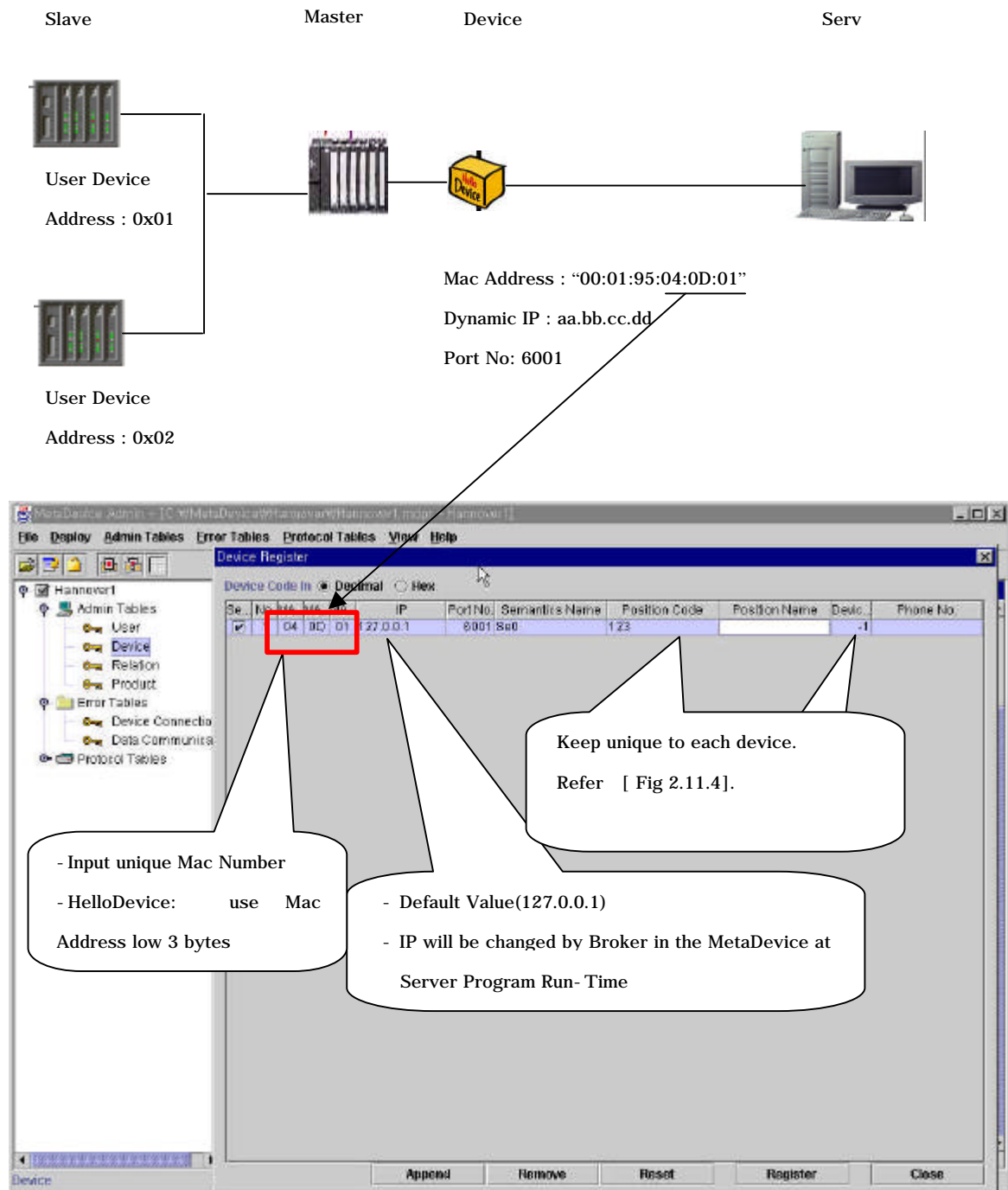
[Fig 2.11.6 Device Registration Window Ex1]

- Case 2



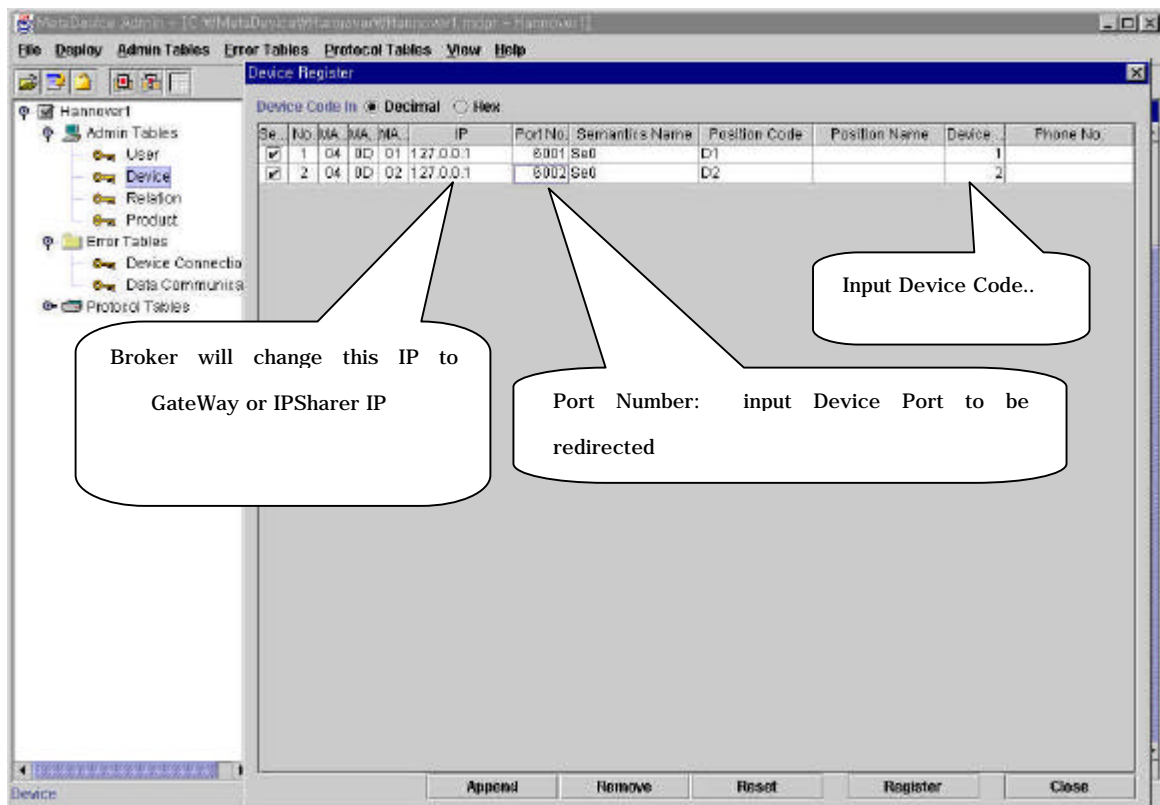
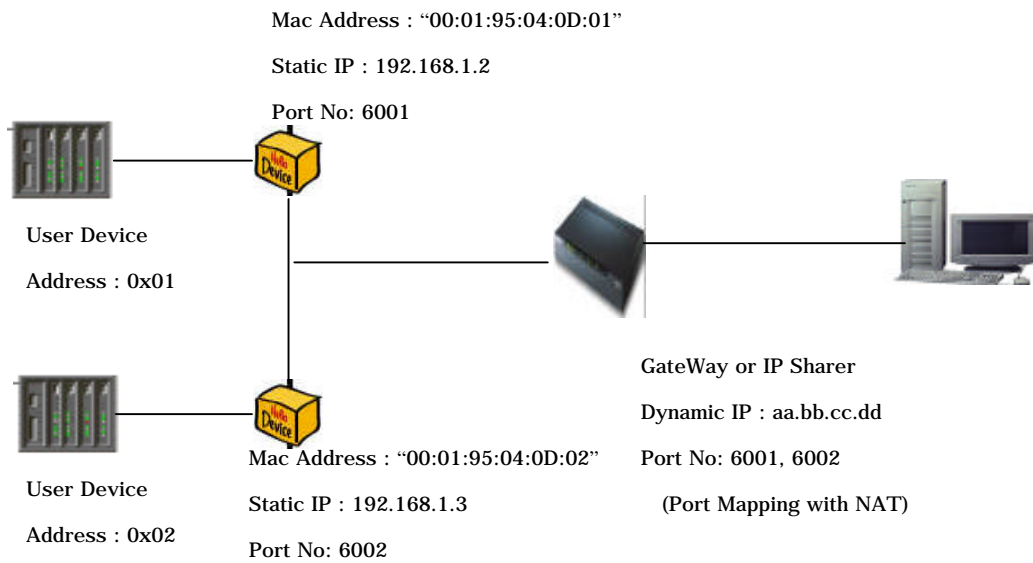
[Fig 2.11.7 Device Registration Window Ex2]

- Case 3



[Fig 2.11.8 Device Registration Window Ex3]

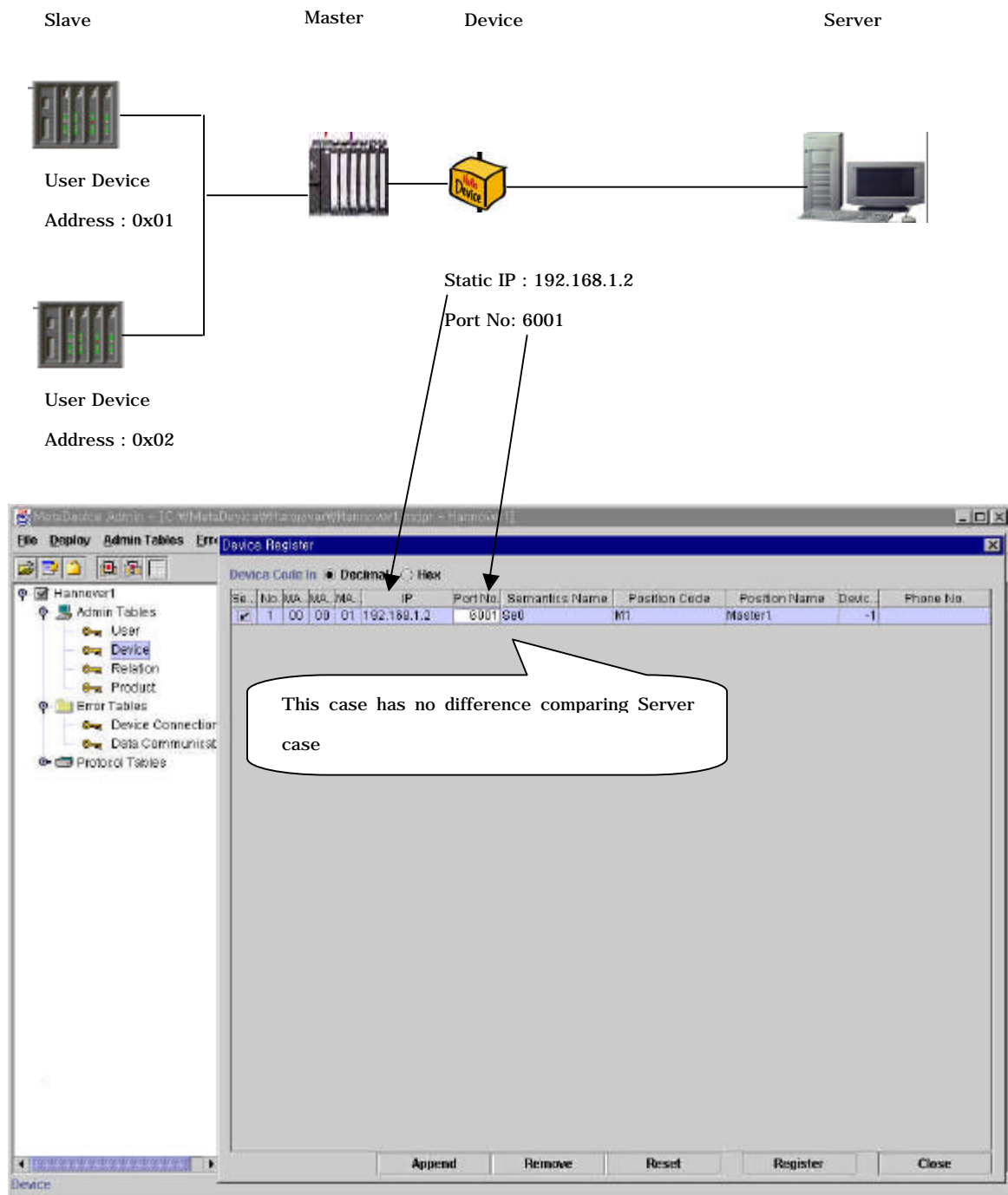
- Case 4



[Fig 2.11.9 Device Registration Window Ex4]

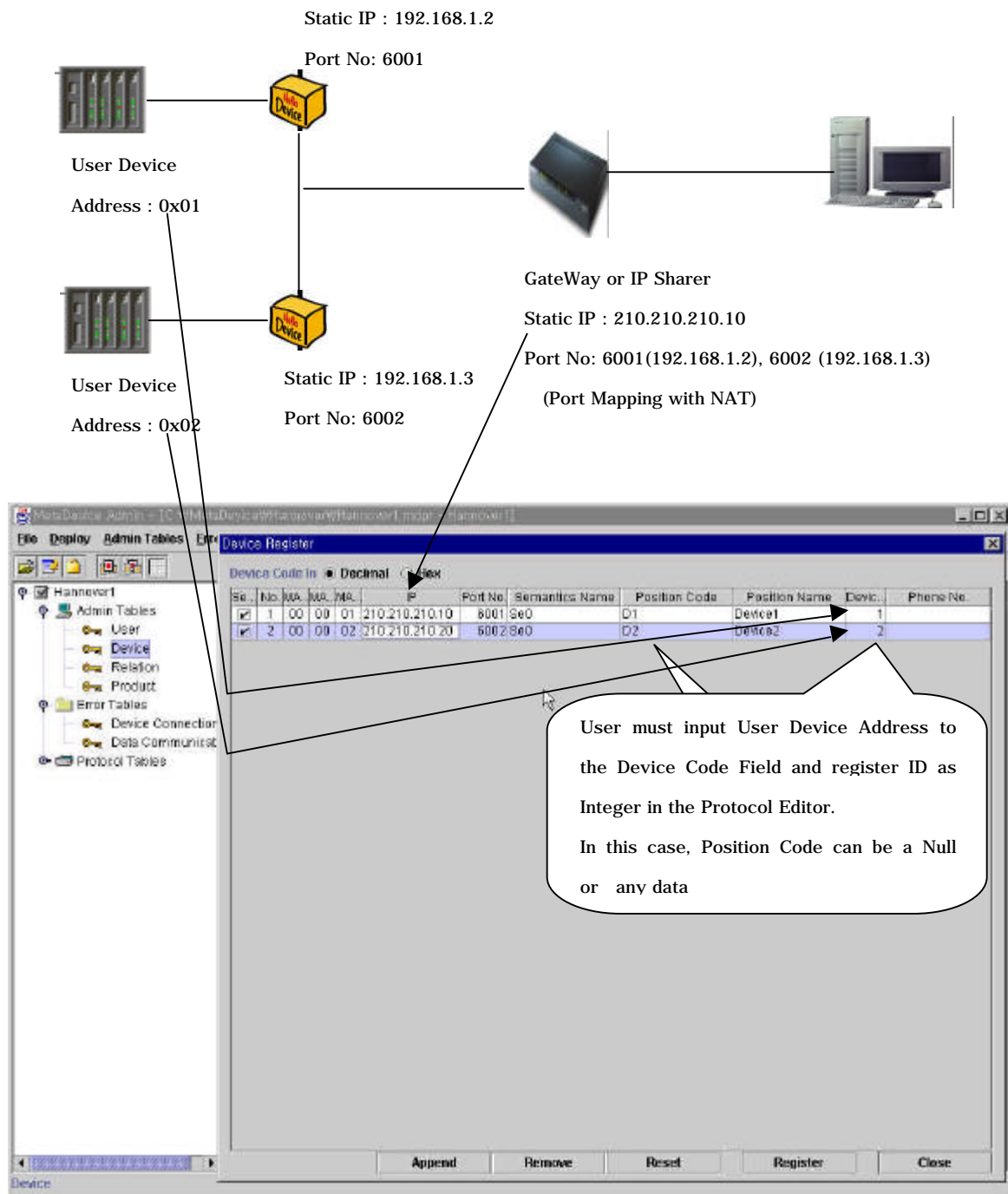
(2) When the Device is used as a Client

- Case 1



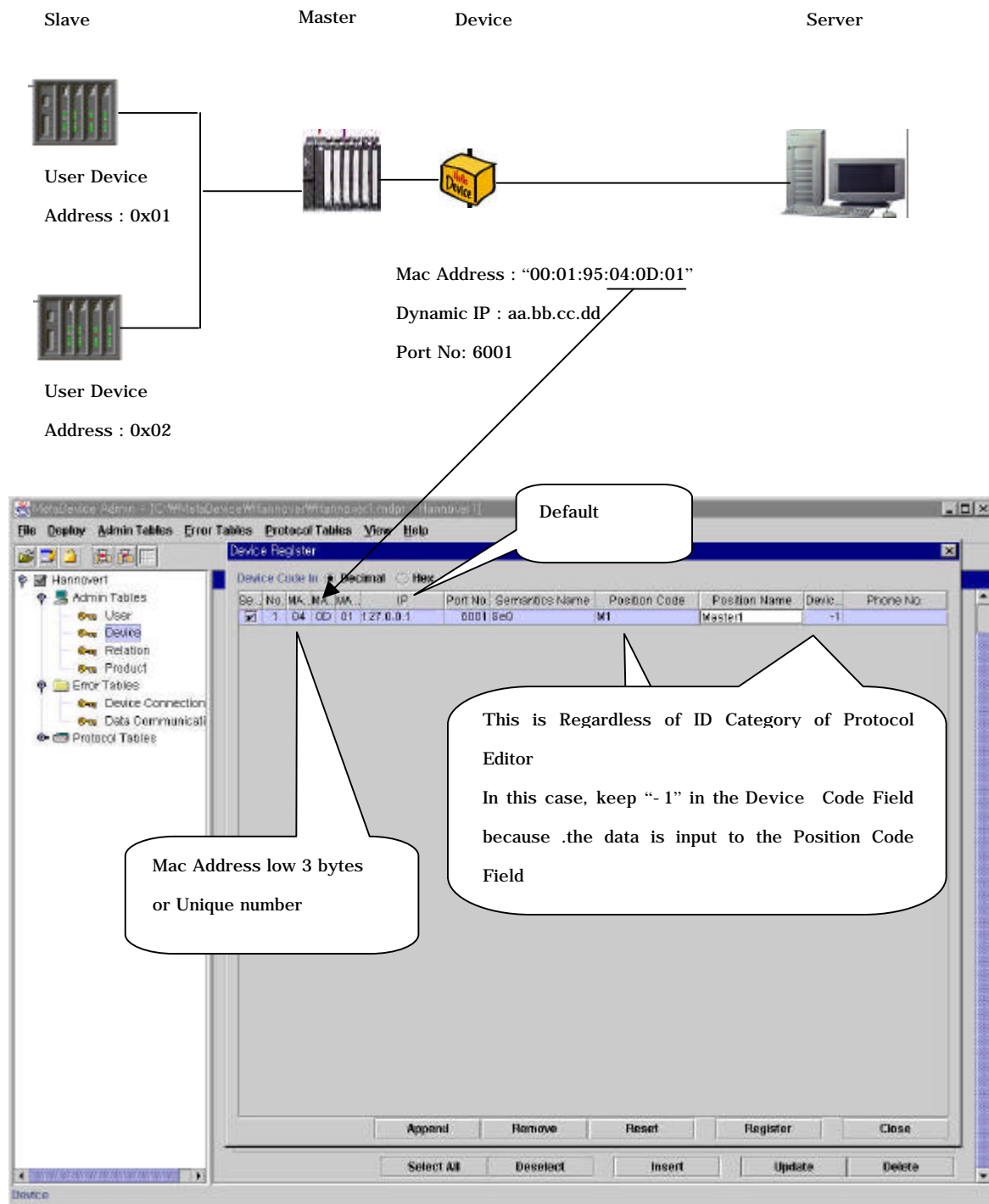
[Fig 2.11.10 Device Registration Window Ex5]

- Case 2



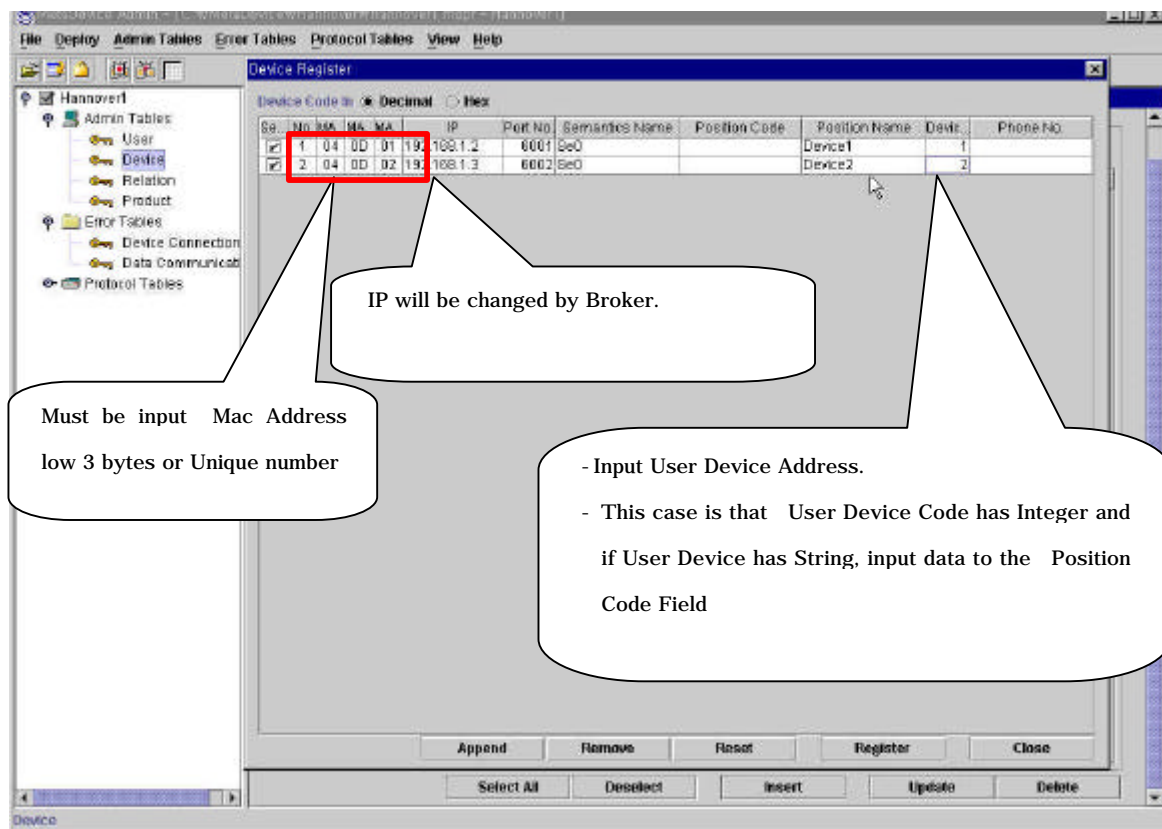
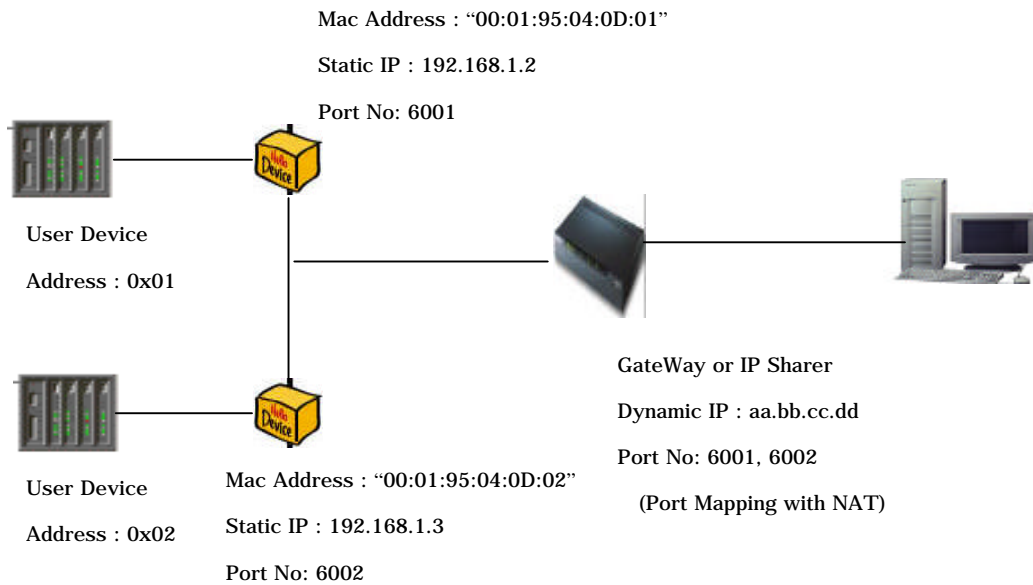
[Fig 2.11.11 Device Registration Window Ex6]

- Case 3



[Fig 2.11.12 Device Registration Window Ex7]

- Case 4



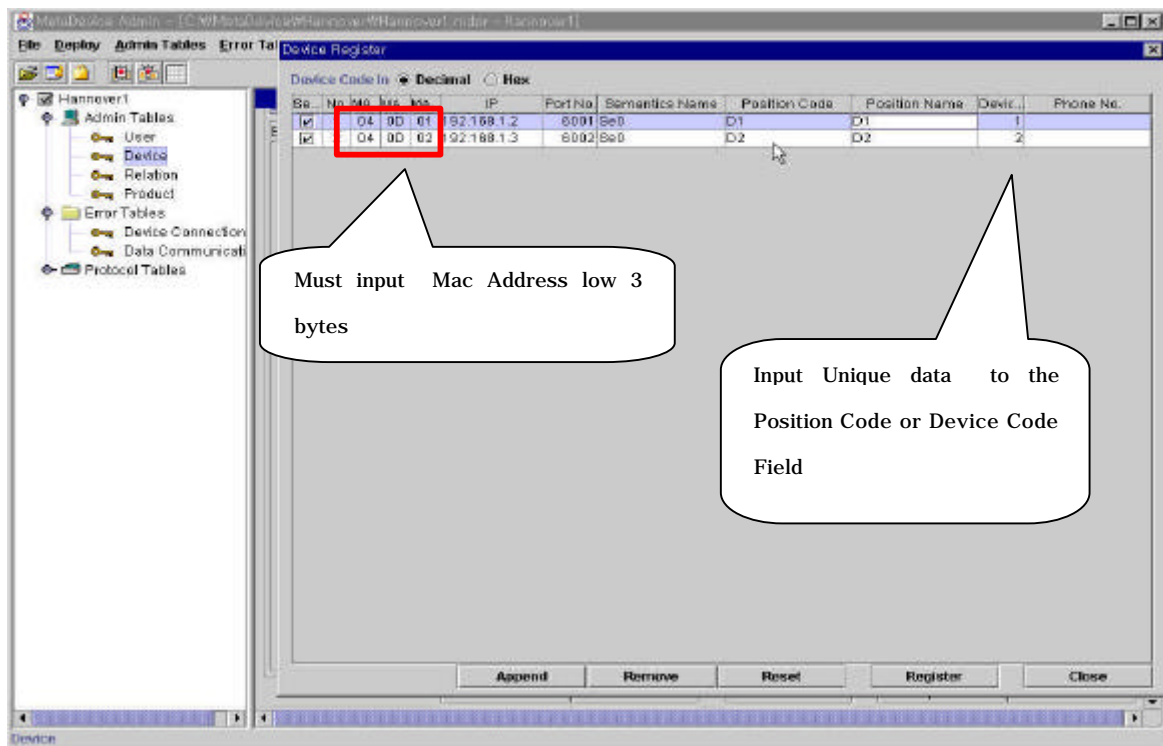
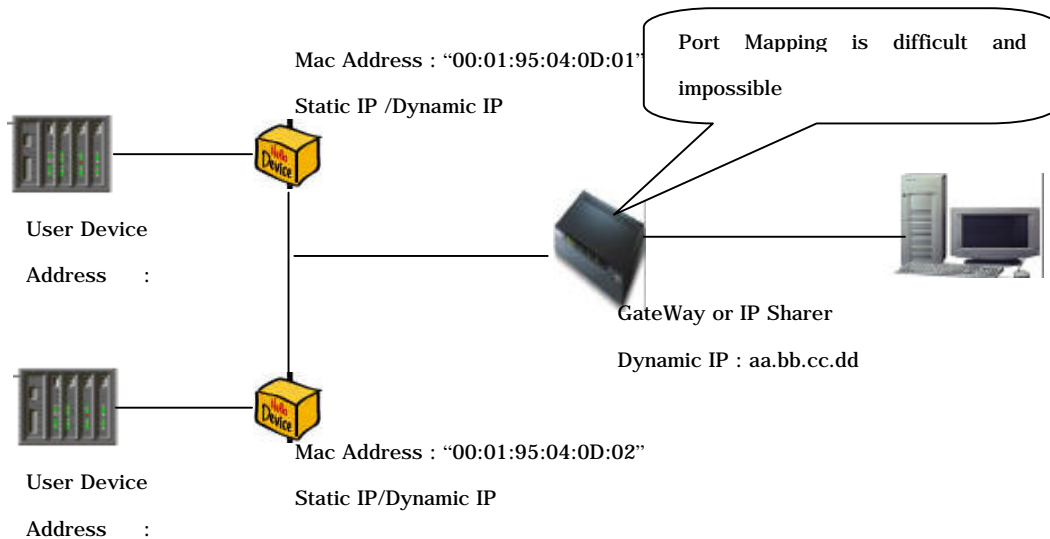
[Fig 2.11.13 Device Registration Window Ex8]

(3) When the Device is used as a Client/Server

Follows the standards when the Device is used as a Ser ver.

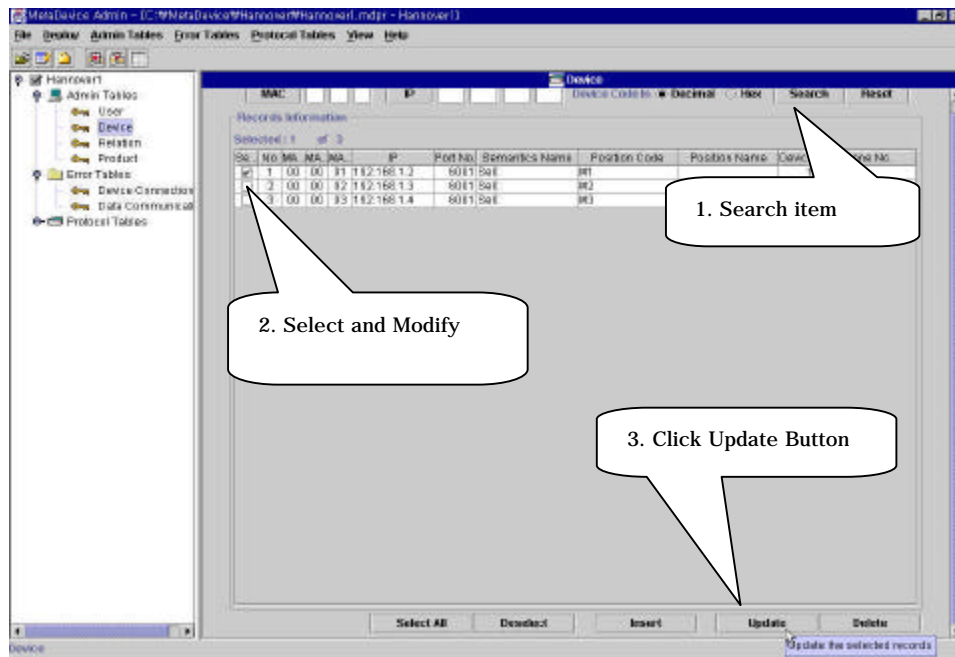
(4) Special Cases

When Port redirecting of Gateway is difficult or when HelloDevice is used as a Private IP, there is no way to connect to the HelloDevice from the Server PC. In order to solve this problem, it is possible to use HelloDevice TCP Connection function. In other words, the Hello Device can periodically connect to the Server and receive commands from it. In Server Semantics, this is registered as “_MAC”.

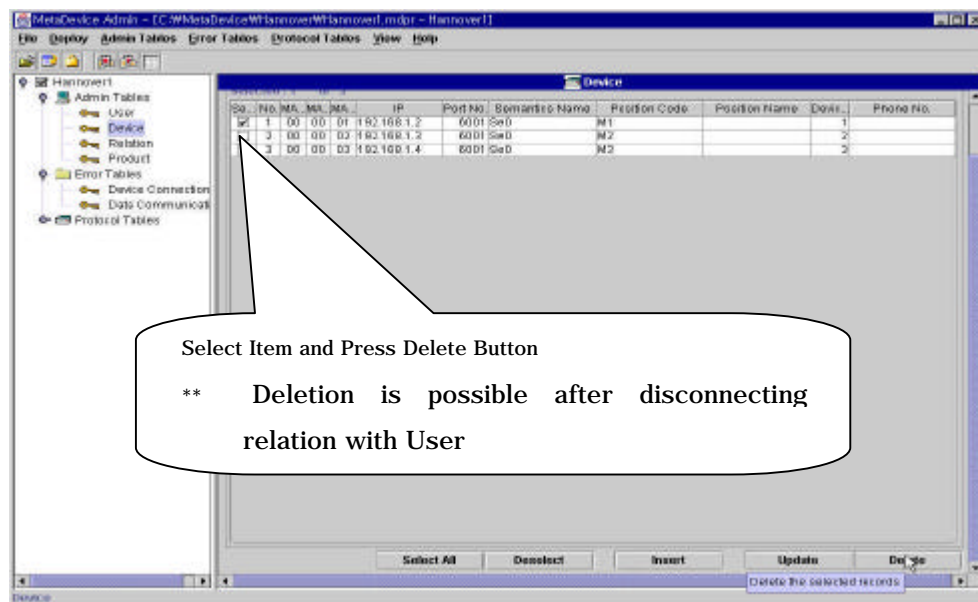


[Fig 2.11.14 Device Registering in Special Cases]

2.4 Device Deletion / Modifying



[Fig 2.12. Device Modifying Window]

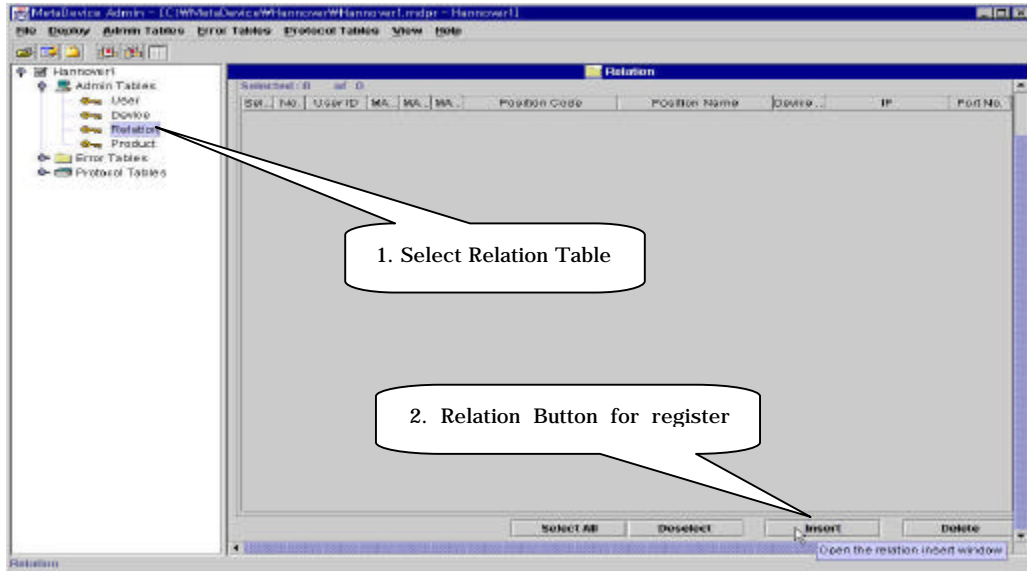


[Fig 2.13 Device Deletion Window]

2.5 User/ Device Relation Management

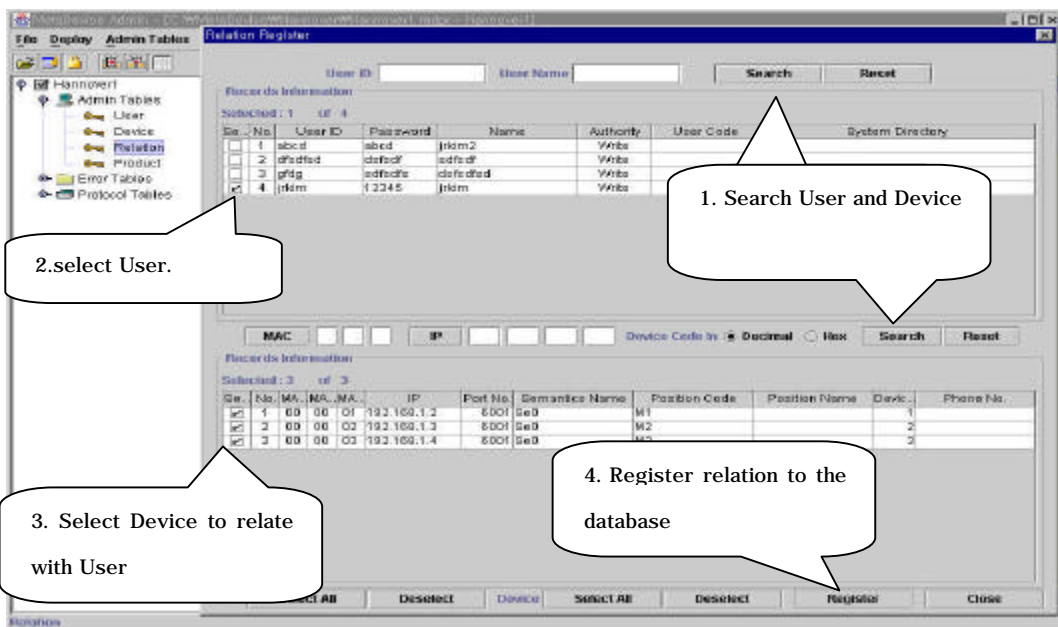
Used to register authority to the Device for the user.

User authorization for each Device can be chosen during UI Deployment or Run-Time.



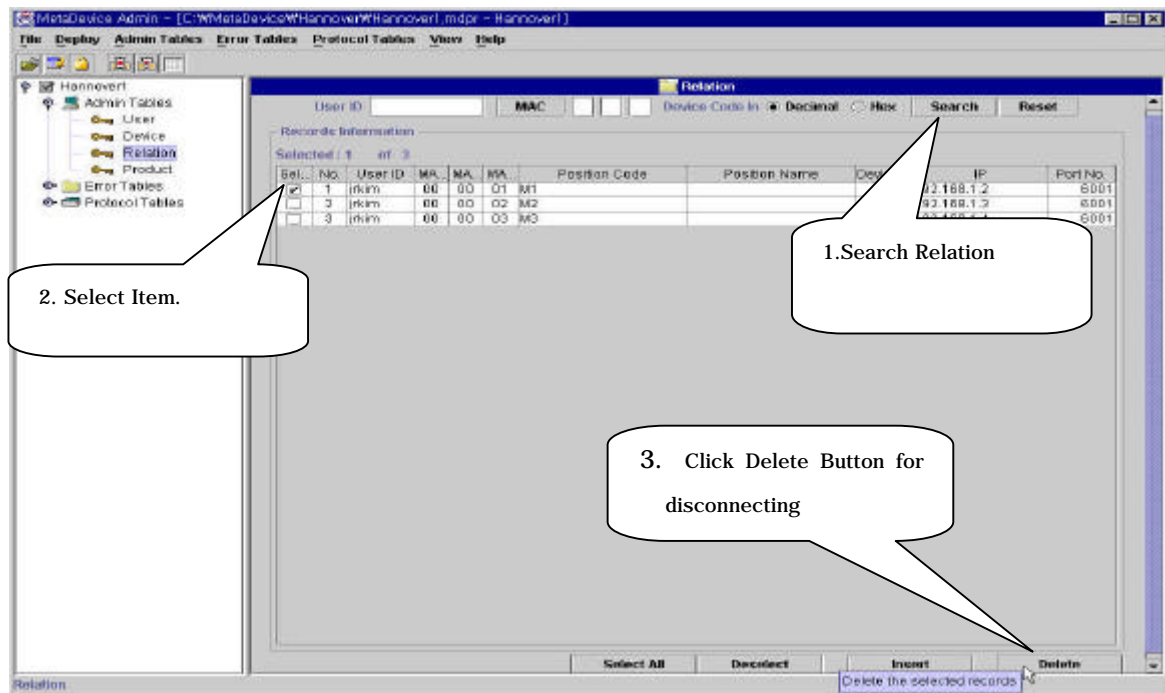
[Fig 2.14 Relation Management Menu]

(1) Setting User/Device Relation



[Fig 2.15 Relation Window]

(2) Disconnecting User/Device Relation



[Fig 2.16 Disconnecting Window]

2.6 Miscellaneous

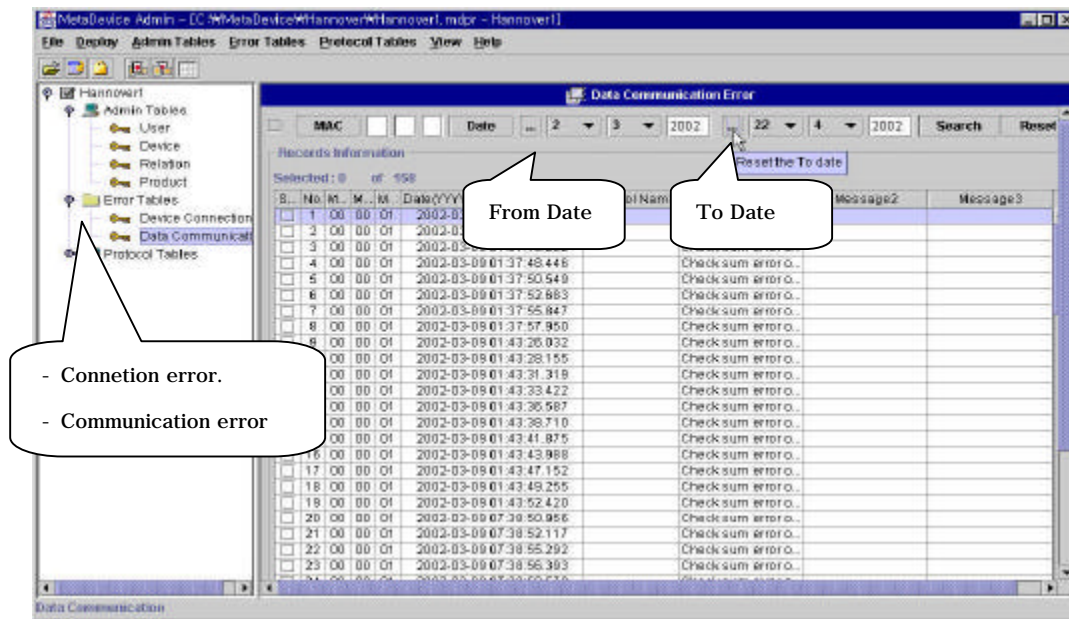
(1) Communication Error Management

The Admin user can see the Error List that occurs during communication. The types of Errors are "Device Connection Error", "No Data Receive", "Data Parsing Error" and "Check Sum Error".

.Device Connection Error: When the Server Daemon Type is Client and fails to connect to the Device

.No Data Receive : When the protocol is not transferred from the Device even though it is defined as such in Semantics

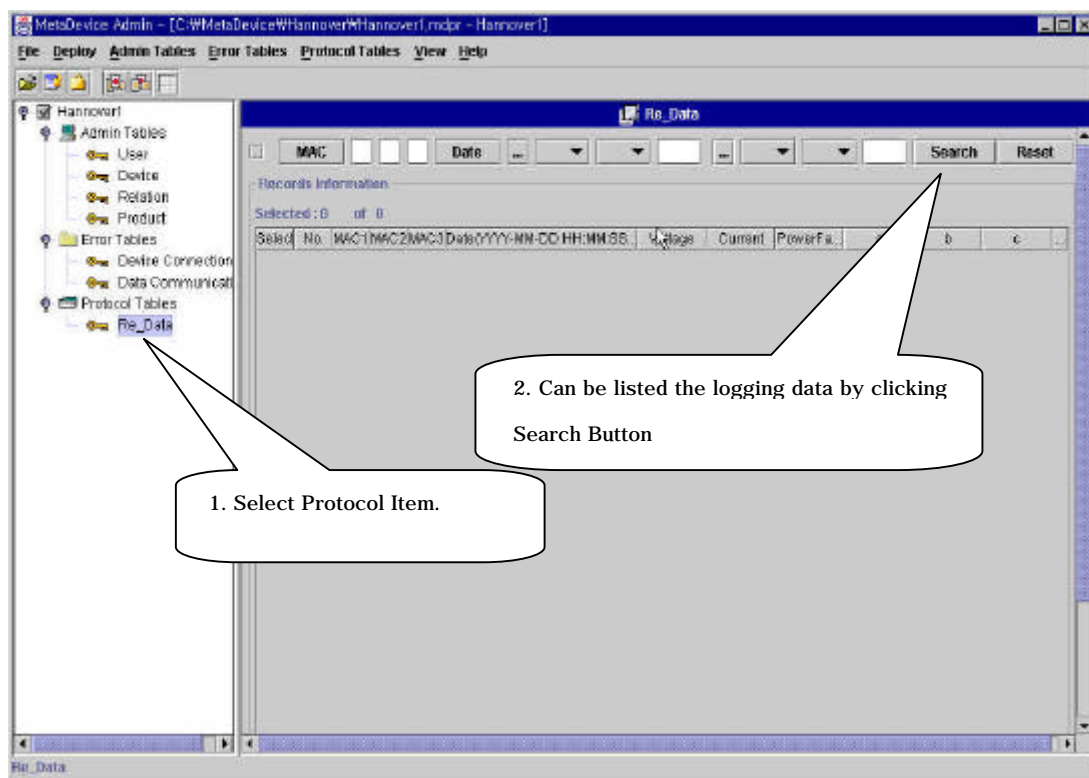
.Data Parsing Error : When the data format is different from the Protocol Format defined in Semantics.



[Fig 2.17 Communication Error Search Window]

(2) Protocol Data Search Window

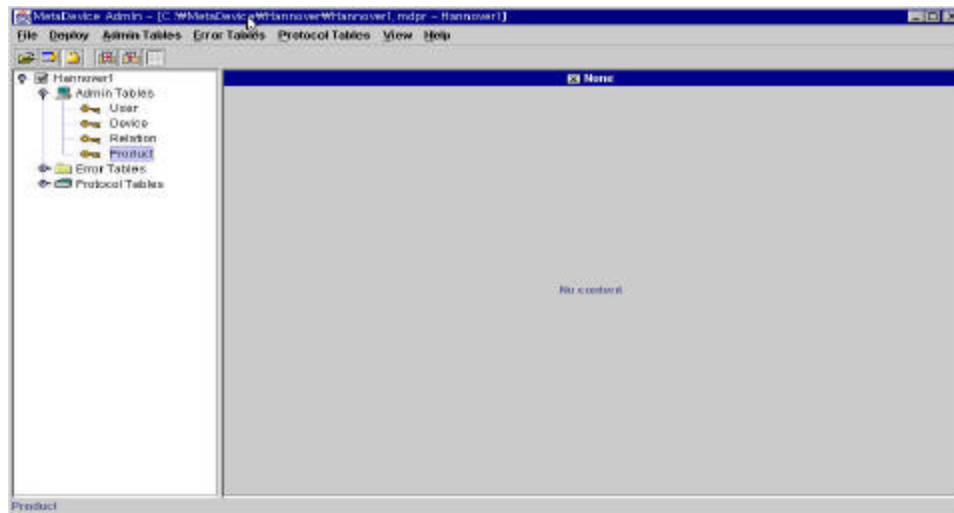
Data Logging information defined in the Server Protocol



[Fig 2.18 Protocol Data Search Window]

(3) Product Management

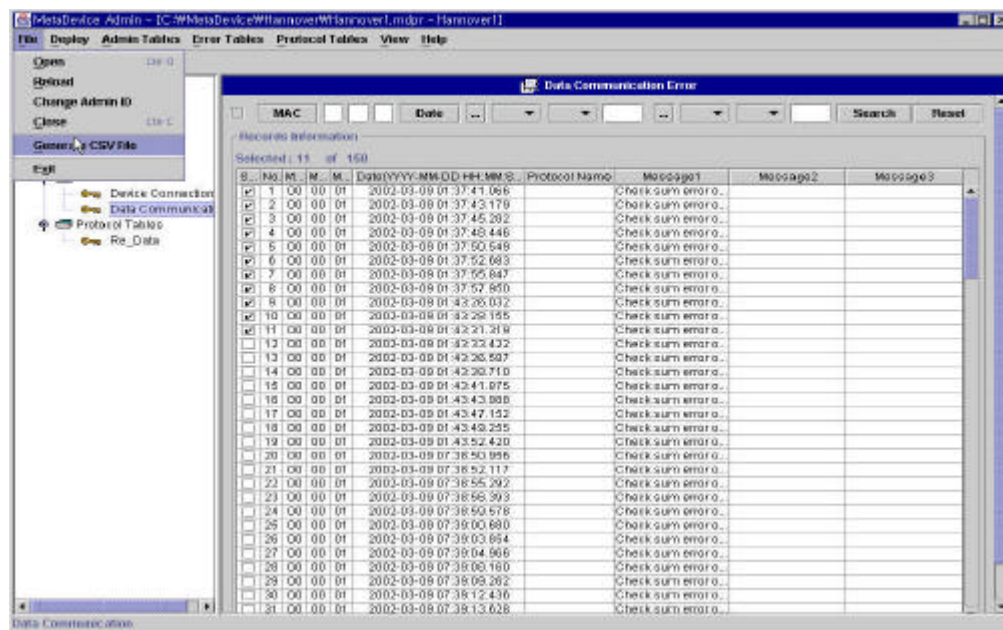
Supported in Ver 1.5. and used when managing equipment related to Multi Port or Device information.



[Fig 2.19 Product Data Window]

(4) Generating CSV File

User can use this function when managing the listed data as an Excel File.

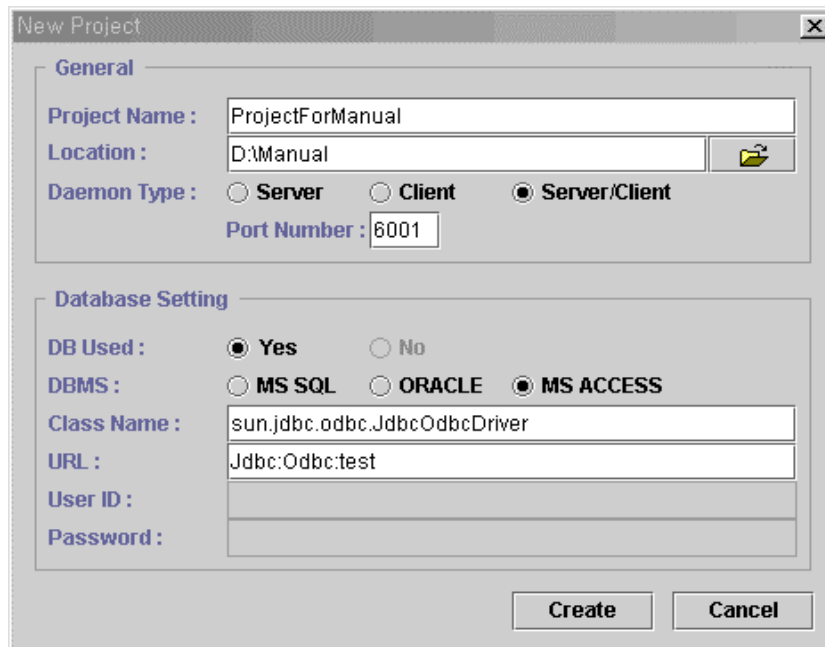


[Fig 2.20 Generating CSV File]

3. General Property

The General Property sets basic information of the Server Program generated by the Server Manager. The General Property includes Project Name, Location, Server Program Daemon Type, Port Number, DB Used, type of DBMS, JDBC Class Name, DBMS URL and DBMS User ID/ Password, etc.

The General Property of the Server Program is first established when the project is newly created (select "File/New" Menu in the New Project window [Fig 2.21.1]) and can be modified in the General Panel [Fig 2.21.2]

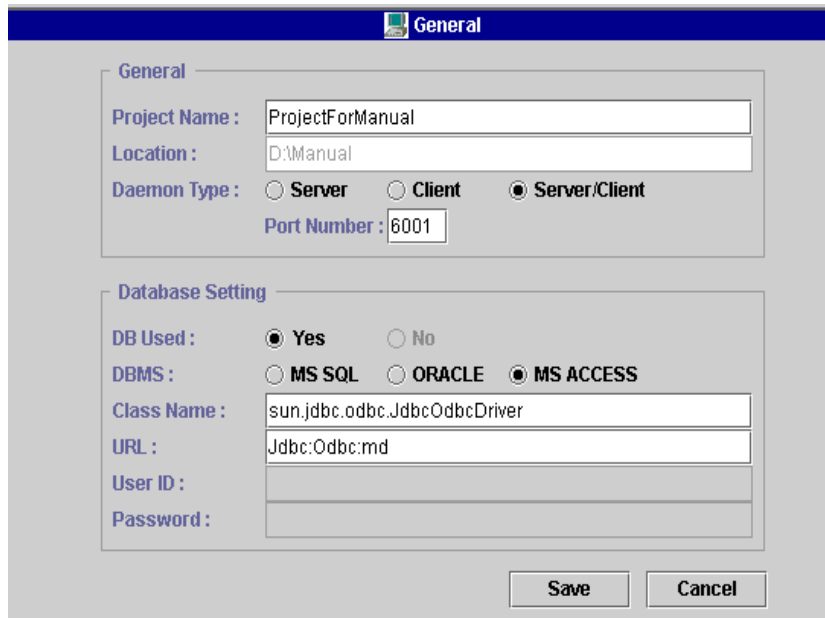


The 'New Project' dialog box is shown with the following settings:

- General**
 - Project Name: ProjectForManual
 - Location: D:\Manual
 - Daemon Type: ☐ Server ☐ Client ☒ Server/Client
 - Port Number: 6001
- Database Setting**
 - DB Used: ☒ Yes ☐ No
 - DBMS: ☐ MS SQL ☐ ORACLE ☒ MS ACCESS
 - Class Name: sun.jdbc.odbc.JdbcOdbcDriver
 - URL: Jdbc:Odbc:test
 - User ID:
 - Password:

Buttons: Create, Cancel

[Fig 2.21.1 New Project Window]



The 'General' panel is shown with the following settings:

- General**
 - Project Name: ProjectForManual
 - Location: D:\Manual
 - Daemon Type: ☐ Server ☐ Client ☒ Server/Client
 - Port Number: 6001
- Database Setting**
 - DB Used: ☒ Yes ☐ No
 - DBMS: ☐ MS SQL ☐ ORACLE ☒ MS ACCESS
 - Class Name: sun.jdbc.odbc.JdbcOdbcDriver
 - URL: Jdbc:Odbc:md
 - User ID:
 - Password:

Buttons: Save, Cancel

[Fig 2.21.2 General Panel]

3.1 Project Name

The Project Name is used for the Server Program Name, and should follow these naming procedures because of its role as a Server Manager Identifier

[Server Manager Identifier]

1. Type : Project Name, Send Protocol Name, Receive Protocol Name, Transfer Name, Semantics Name
2. Possible Characters : Upper/Lower case English, Arabic Numbers, _
3. Restrictions :
 - 1) The first character should be English. (ex : _Abc , 123Abc are not allowed)
 - 2) More than one character should be in lower case English.(ex : ABC123, AAA are not allowed)
 - 4) The length should be within 20 characters.(ex: IAmLongerThan20Letters is not allowed)
 - 3) Recommendation : The Identifier should use detailed definitions and begin with upper case English letters. Use lower case English letters within a word and use upper case letters when beginning a new word.
 - 4) Do not use JAVA Key Word
 boolean, byte, short, int, long, char, float, double, void, true, false, null, if, else, switch, case, default, for, while, do, break, continue, return, class, interface, extends, implements, static, abstract, final, new, instance, of, this, super, public, protected, private, volatile, transient, synchronized, native, package, import, try, catch, finally, throw, throws
 - 5) Do not use Server Manager Key Word or Reserved Word.
 _MAC, NO_DATA_RECEIVE, PARSE_ERROR, CS_ERROR, ALARM_RECEIVE, Broker, CRC, DateUtil, DBUtil, InstantMessage, LogIn, MAC, MACInfo, Protocol, ReceiveProtocol, RMIServer, RSSerializable, RSSerializableImpl, Semantics, SemanticsInterface, SendProtocol, Utility
 - 6) The Identifier should be unique

3.2 Location

The Location is where the project file is saved(extension mdpr). The “Server” folder is created automatically in the Location directory when you deploy the Server Program (Deploy/Make Server Menu). When you create a new project, it is possible to set up from the New Project window, but you can not edit from the General Panel. In Operating Systems such as Windows NT, folder names composed solely of upper case letters are not allowed in some cases. Therefore at least one lower case letter should be used for folders for Location Paths.

3.3 Daemon Type

Defines whether the Server Program type is used as a Server or a Client when communicating with Device. If the Server Program is used both as a Server and a Client, select Server/Client. If the Server Program is used as a Server (if you have selected Server or Server/Client), input the Port Number that will be used for the Device's connection

Semantics Editor ClieN/Server Chek Box in the Semantics Panel depends on Server Daemon Type. If the Daemon Type is set as Server, the Client Check Box of the Semantics Panel cannot be selected (making it impossible to input Client Semantics). If the Daemon Type is set as Client, the Server Check Box of the Semantics Panel can not be selected (making it impossible to input Server Semantics.)

The screenshot shows the 'Semantics / se0[SeForManual]' dialog box. The 'Semantics properties' section includes 'Connection Type' (Continuous, Chunk), 'Set Send Protocol NULL after sending' (Frame Data, Individual Data, Not Set NULL), 'Frame Properties' (Frame Polling Time, Maximum Repetition Times For Connection), and 'Receive Properties' (Maximum Receive Buffer Size, Receive Sleep Time, Maximum Repetition Times For Receiving). The 'Semantics edit' section contains two checkboxes: 'Server' and 'Client'. Both are highlighted with red boxes. Callouts point to these checkboxes with the following text:

- Server Check Box :**
If Daemon Type is set as Server or Server/Client in the General Panel, Server Check Box can be selected
- Client Check Box :**
If Daemon Type is set as Client or Server/Client in the General Panel, Client Check Box can be selected

At the bottom, there is an 'Instant' checkbox, an 'Additional Protocols' field, and 'VERIFY', 'SAVE', and 'CANCEL' buttons.

[Fig 2.22.1 Daemon Type and Semantics Panel]

3.4 Port Number

When the Daemon Tpe is set as Server or Server/Client, the Server Program becomes the Server

while the Device attempt connection to the Server Program. Input the Port Number so that the Device can connect to the Server.

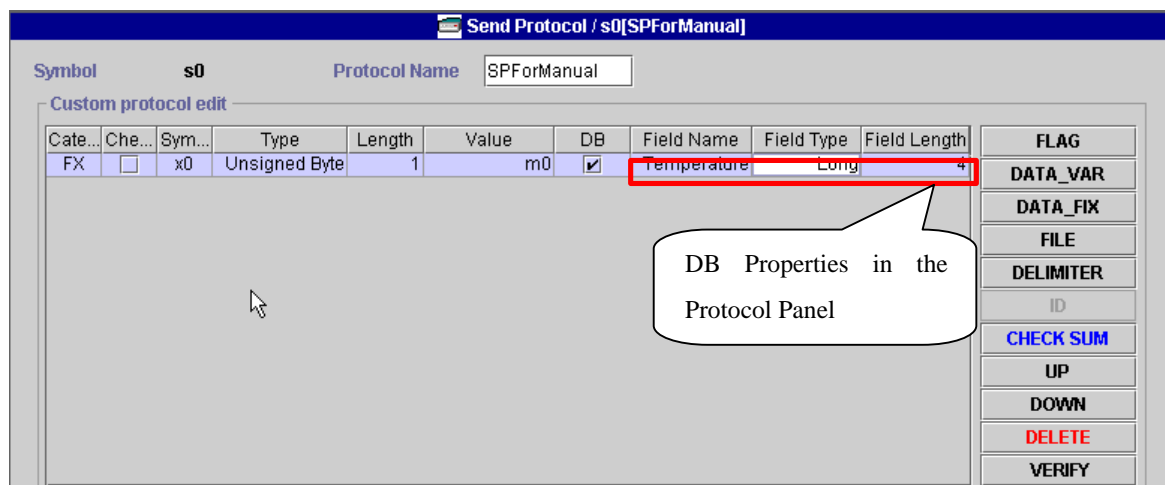
3.5 DB Used

Determine whether the Server Program uses DBMS or not

In Version 1.3., DB Used is always “yes”.

3.6 DBMS

Set the type of DBMS that the Server Program uses. Ver 1.3 supports MS SQL, MS Access and Ver2.0 will support ORACLE. Depending on the type of DBMS, the set up method for the Field Type, Field Length of the Protocol Panel(Send Protocol and Receive Protocol Panel) changes. Since the Field Type, Field Length already set in the Protocol Panel is changed according to the Field Type and Default Field Length that fits the type of DBMS, it is necessary to set these items in the Protocol Panel. Check and reset DB environments in the Protocol Panel when changing the type of DBMS.



[Fig 2.22.2 DB Properties of the Protocol Panel]

3.7 DB Connection Properties

Some DB Connection Properties that are necessary for the Server Program to connect to the DB are Class Name, URL, User ID, and Password. The Class Name is the class name of the JDBC Driver. The URL signifies the URL of the DBMS that will be used, while User ID and Password signifies the registered ID and Password that are registered in the DBMS. Users should be aware of New settings since properties change according to the DBMS and JDBC Driver that are to be used. example)

1. DBMS : Access , DSN(Data Source Name) : DSNName, Driver : When using Sun's JDBC/ODBC Driver (When using DSN from Access DB, User ID, Password not necessary)
Class Name – "sun.jdbc.odbc.JdbcOdbcDriver"
URL – "Jdbc:Odbc:DSNName"
User ID – ""
Password – ""
2. DBMS : MS SQL, DSN : DSNName, Driver : When using Sun's JDBC/ODBC Driver (Register with User ID : kumar Password : kumar)
Class Name – "sun.jdbc.odbc.JdbcOdbcDriver"
URL – "Jdbc:Odbc:DSNName"
User ID – "kumar"
Password – "kumar"
3. DBMS : MS SQL, Driver : Microsoft SQLServerDriver, Server Name : When installing with ServerName (Register with User ID : kumar Password : kumar)
Class Name – "com.microsoft.jdbc.sqlserver.SQLServerDriver"
URL – "jdbc:microsoft:sqlserver:// ServerName:1433"
User ID – "kumar"
Password – "kumar"
4. DBMS : MS SQL, Driver : Microsoft SQLServerDriver, When the installed DBMS IP is : 61.252.52.5 (Register with User ID : kumar Password : kumar)
Class Name – "com.microsoft.jdbc.sqlserver.SQLServerDriver"
URL – "jdbc:microsoft:sqlserver:// 61.252.52.5:1433"
User ID – "kumar"
Password – "kumar"

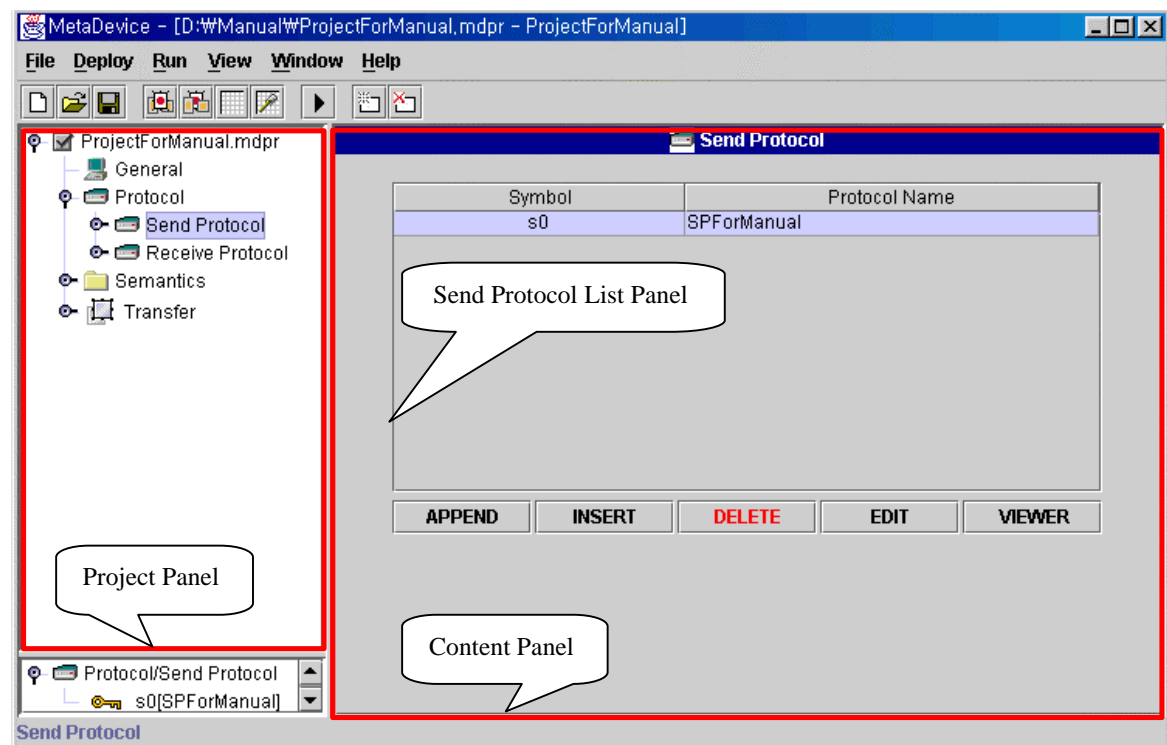
4. Protocol

The Data format communicated between the Server Program and Device is called the Protocol. Data format sent from the Server Program to the Device is called the Send Protocol, while Data format sent from the Device to the Server Program is called the Receive Protocol. The Send Protocol is also used when the user sends Data from the UI Program to the Server Program. The Receive Protocol is also used when Data received from the Device is transferred from the Server Program to the UI Program.

The user can register or delete Protocol from the Protocol List Panel, move to the Protocol Edit Panel, or open Protocol View. Each Protocol can be edited from the Protocol Edit Panel.

4.1 Protocol List Panel

This is the Protocol management window for registering and deleting protocol. Send Protocol List Panels and Receive Protocol List Panels are included. This window can move to Protocol Edit Panel as well as open the Protocol Viwer so that the user can check Protocols when editing Semantics and Transfer.



[Fig 2.23 Send Protocol List Panel]

(1). Opening Protocol List Panel

(1.1) Selecting the Send Protocol Node below the Protocol Node in the Project Panel will open the

Send Protocol List Panel, while selecting the Receive Protocol Node will open the Receive Protocol List Panel.

(1.2) Selecting the View/Protocol/Send Protocol menu will open the Send Protocol List Panel.

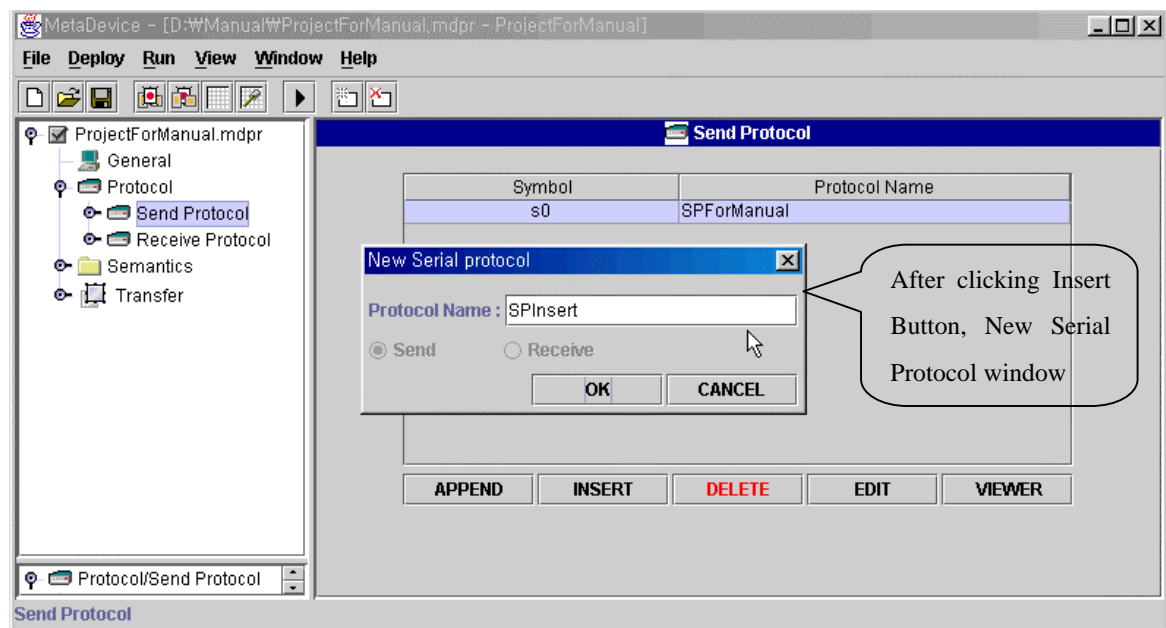
Selecting the View/ Protocol/Receive Protocol menu will open the Receive Protocol List Panel.

(2). Registering Protocol

(2.1) The New Serial Protocol window will be displayed when selecting “Append” from the Protocol List Panel. Type in the Protocol Name and click OK. The new Protocol will be added below the Protocol List of the Protocol List Panel.

(2.2) The New Serial Protocol window will be displayed when selecting “Insert” from the Protocol List Panel. Type in the Protocol Name and click OK. The new Procol will be added in front of the selected Protocol of the Protocol List Panel.

(2.3) The Protocol Name is named according to the Server Manager Identifier's set of rules. (Refer to 3. General Property 3.1 Project Name [Server Manager Identifier])



[Fig 2.24 New Serial Protocol Window]

(3). Deleting Protocol

(3.1) After selecting the Protocol to be deleted from the Protocol List Panel, click “DELETE”.

(3.2) Clicking “Yes” from the Delete a row Dialog Box will delete the Protocol selected in the

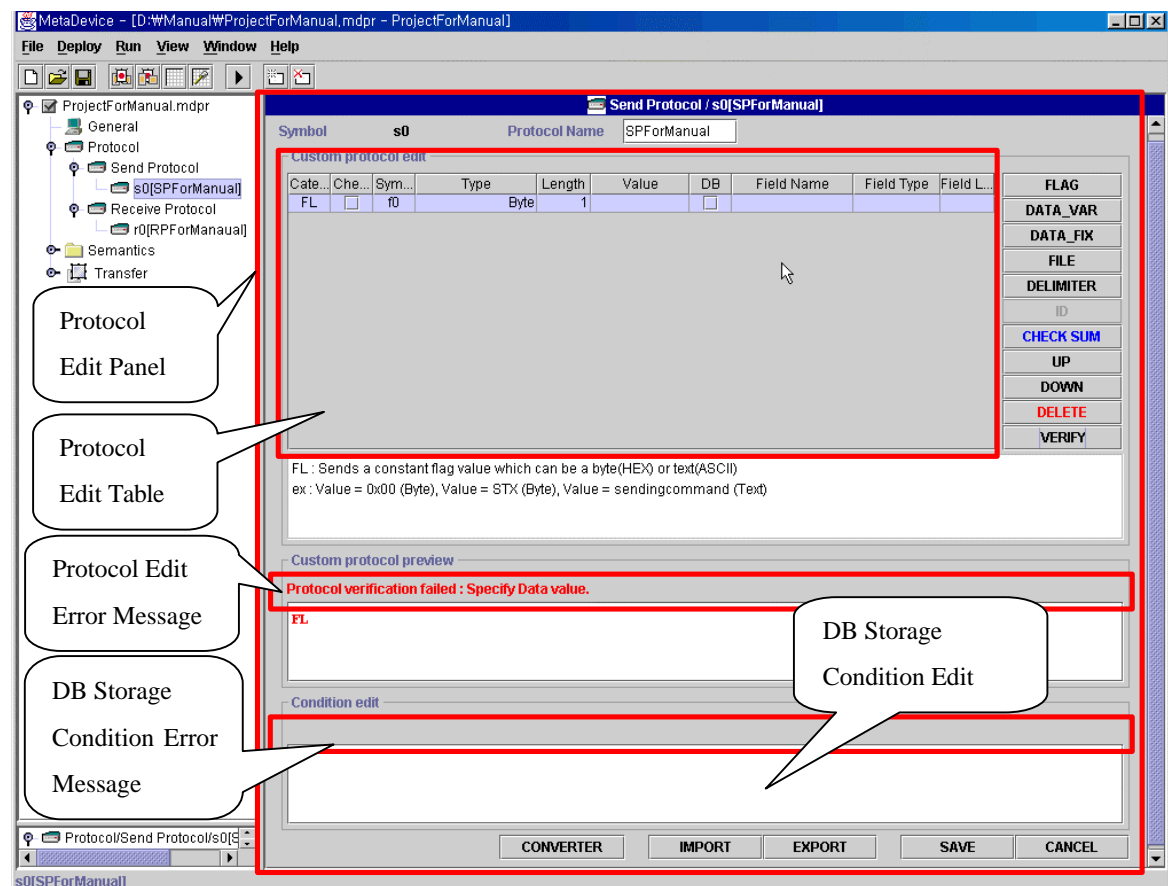
Protocol List Panel.

(4).Opening Protocol Viewer

- (4.1) Select Protocol from the Protocol List Panel.
- (4.2) Clicking "VIEWER" will open the Protocol Viewer window.

4.2 Protocol Edit Panel

This window is for editing registered Protocol data. Each Protocol item is defined by Category, Check Sum Field Selection, Symbol, Type, Length, Value, DB Storing Condition, Field Name of DB Protocol Table, Field Type, and Field Length. Defining methods will differ depending the usage of Send Protocol or Receive Protocol. Edit Protocol elements from the Protocol Edit Table and display editing error results in the Protocol Edit Error Message window. Edit the Protocol's DB storage conditions from the DB Storage Condition Edit window and display the error results from the DB Storage Condition Error Message window.



[Fig 2.25 Protocol Edit Panel]

(1). Opening the Protocol Edit Panel

- (1.1) Select Send Protocol Node or Protocol Node to be edited in the Receive Protocol Node from the Project Panel.
- (1.2) Select the Protocol item to be edited from the Protocol List Pnael and click "EDIT"
- (1.3) Double click the Protocol item to be edited from the Protocol List Panel.

(2). Protocol Editing Category

(2.1) Category : the type of Protocol element

- FLAG : Constant (can be byte or text)
- VARIABLE DATA : Data with variable size
- FIXED DATA : Data with fixed size
- FILE : Data saved in file (Used as Send Protocol item only)
- DELIMITER : Symbol used to signify end of Variable Data
- ID : ID of Device connected with Server Program (Used as Receive Protocol item only)
- CHECK SUM : Checks whether data has been transferred correctly (CRC16, XOR, Ascii2, ...)

(2.2) Check Sum Field : If the Check Sum element exists as part of the Protocol, this category can check if it has been used it caculating the Check Sum.

(2.3) Symbol : Name of Protocol element

(2.4) Type : The type of Protocol element. Can have differing types depending on the Protocol element category.

(2.5) Length : The length of Protocol element. Established differently depending on the Protocol element category.

(2.6) Value : The value of Protocol element. Established differently depending on the Protocol element category.

(2.7) DB Storing: Users can check the Protocol element to see whether the element was saved in the DataBase or not.

(2.8) Field Name : If the Protocol element is checked for DataBase, user can define DB field name.

(2.9) Field Type : DataBase Field Type. The Field Type depends on the DBMS.

(2.10) Field Length : DataBase Field Length. The Field Type depends on the DBMS.

(3). Registering Protocol elements

(3.1) Select Protocol element which comes after the Protocol element that is to be registered.

(3.2) Select the Category of the new Protocol element.

Category	Button	Category	Button
FLAG	FLAG	DELIMITER	DELIMITER
VARIABLE DATA	DATA_VAR	ID	ID
FIXED DATA	DATA_FIX	CHECK SUM	CHECK SUM
FILE	FILE		

[Table 2.1 Category Name]

(3.3) A Protocol element is created with default value in the Protocol Edit Table preceding the selected Protocol element.

(4). Deleting Protocol elements

(4.1) Select Protocol element to be deleted.

(4.2) Click DELETE button.

(4.3) The element will be deleted.

(5). Verification of Protocol Editing

(5.1) After registering and editing the Protocol element, click VERIFY.

(5.2) If there are no errors, a "Protocol verification successful" message will appear in Protocol Edit Error Message window, and a "Condition verification successful" message will be displayed in the DB Storage Condition Error Message window.

(5.3) If there are any errors, the error message will be displayed in the Protocol Edit Error Message window. The DB Storage Condition Error Message window will also display any errors that have occurred.

For detailed information on error messages, refer to [9. Protocol element edit error][11. DB Storage Condition Error].

(6). Saving Protocol Data

(6.1) After editing Protocol elements, click SAVE.

(6.2) If there are no errors after verification, you can save the Protocol data and project

file(mdpr file).

(6.3) If there are any errors, you can see the error messages from the displayed windows.

(7). Editing Send Protocol Elements

(7.1) FLAG

Edit Item	Editing	Value	Description
Category	X	FL	
Check Sum	O	true or false	Select from Check Box
Symbol	X	f + Unsigned Int.	0 or Integer created automatically according to location
Type	O	Byte or Text	Select from List Box
Length	X	Unsigned Integer	Set automatically according to Type and Value If the Type is Byte, "1" If the Type is Text, length of text displayed in Value
Value	O	Byte, Byte Symbol, Text	1) Type is Byte Hex Byte value ex) '0x02' Byte Symbol (refer [Byte Symbol Table]) ex) 'STX' 2) Type is Text Text value ex) 'C', 'abcd'
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB
Field Length	X		Can not be stored in the DB

[Table 2.2 Editing Send Protocol FLAG]

[Byte Symbol Table]

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
NUL	0x00	BS	0x08	DLE	0x10	CAN	0x18
SOH	0x01	HT	0x09	DC1	0x11	EM	0x19
STX	0x02	LF	0x0A	DC2	0x12	SUB	0x1A
ETX	0x03	VT	0x0B	DC3	0x13	ESC	0x1B
EOT	0x04	FF	0x0C	DC4	0x14	FS	0x1C
ENQ	0x05	CR	0x0D	NAK	0x15	GS	0x1D
ACK	0x06	SO	0x0E	SYN	0x16	RS	0x1E
BEL	0x07	SI	0x0F	ETB	0x17	US	0x1F

[Table 2.3 Byte Symbol Table]

(7.2) VARIABLE DATA

Edit Item	Editing	Value	Description
Category	X	VA	Sets Send Condition
Check Sum	O	true or false	Select from Check Box
Symbol	X	v + Unsigned Int.	0 or Integer created automatically according to location
Type	X	Byte Array	
Length	O	FLAG Symbol, DELIMITER Symbol, FIXED DATA Symbol, end	<p>VARIABLE DATA must know the end of data</p> <p>1) FLAG Symbol and DELIMITER Symbol are used for the end of the VARIABLE DATA. In this case, FLAG and DELIMITER VARIABLE should immediately succeed VARIABLE DATA. ex) 'f0', 'de'</p> <p>2) FIXED DATA Symbols are possible when FIXED DATA is an Integer. It displays the length of VARIABLE DATA. FIXED DATA should immediately precede VARIABLE DATA. ex) 'x0'</p> <p>3) End signifies that the relevant Data is the end of the Protocol Data. Therefore, no more Protocol elements can succeed it. ex) 'end'</p>
Value	O	v + Unsigned Int	ex) 'v0', 'v100'
DB	O	true or false	Select from Check Box
Field Name	O	Field name	Can be edited only when DB category is true. Contains Data Type of string Type. Refer to [DB related category settings].
Field Type	O	Field Type of String Type	
Field Length	O	Field length	
Send Condition Configuration	O	Send even if null Send only if not null	Double click CATEGORY. Set "send condition to the Device" when the data is null from UI Program.

[Table 2.4 Send Protocol VARIABLE DATA]

[Setting DB related elements]

1. Field Names can be restricted according to DBMS used.(refer to DBMS books)
2. Field Types and Field Lengths should be set differently according to Data Type and DBMS.

Data Type \ DBMS	Access		MS SQL	
	Field Type	Field Length	Field Type	Field Length
String Type	Text Varchar Memo	1 ~ 255 1 ~ 255 65535	char varchar	1 ~ 8000 1 ~ 8000

Integer Type	Long	4	int	4
	Integer	2	smallint	2
	Byte	1	tinyint	1
Float Type	Double	8	float	8
	Single	4	real	4

[Table 2.5 Field Type and Field Length of each Data Type]

(7.3) FIXED DATA

Edit Item	Editing	Value	Description
Category	X	FX	
Check Sum	O	true or false	Select from Check Box
Symbol	X	x + Unsigned Int.	0 or Integer created automatically according to location
Type	O	Unsigned Byte	0 ~ 255 value, short type
		Byte	-126 ~ 125, byte type
		Short	-32768 ~ 32767, short type
		Unsigned Short	0 ~ 65535, integer type
		Short(LE)	Short or Byte array is Little Endian.
		Unsigned Short(LE)	Unsigned Short and Little Endian
		Integer	-65536 ~ 65535, integer type
		Unsigned Integer	0 ~ 4294967295, long type
		Integer(LE)	Integer and Little Endian
		Unsign Integer(LE)	Unsigned Integer and Little Endian
		Float	Floating value, float type
		Float(LE)	Float or Little Endian
		Double	Double value, double type
		Double(LE)	Double and Little Endian
		Text	Text array, String type
		Bit(Byte)	bit value in 1 Byte
Length	O	Unsigned Integer	If Type is Text, input Text length
	X	Unsigned Integer	The length is fixed according to Type. Impossible to edit.
Value	O	v + Unsigned Int.	If Type is Text ex) 'v0', 'v100'
		set or null	Double click Value, open the Bitwise Configuration dialog and edit. Refer to([Bitwise Configuration])
		m + Unsigned Int.	Numeric Type ex) 'm0', 'm100'
DB	X	true or false	If Type is Bit(Byte), edit from Bitwise Configuration dialog
	O	true or false	For other Types, select from Check Box

Field Name	O or X	field name	If Type is Bit(Byte), edit from Bitwise Configuration dialog. In other cases, editing is possible only when DB item is true. If the Type is Text, select Data Type of String Type. In other cases it would be Integer or Float Data Type. Refer to [setting DB related elements].
Field Type		According to Type	
Field Length		field length	

[Table 2.6 Editing Send Protocol FIXED DATA]

[Unsigned]

1. Save as Unsigned data when retrieving data with a defined byte amount and saving it in the system.
2. When reading Protocol data Read data with defined length but when saving in the system, the data is saved larger data type than the data type read.

ex) Type : Unsigned Integer

Length: 4 (When reading from or writing Protocol Data, the data is 4bytes)

System Data Type : long

System Data Length : 8

3. Notice

There is no verification error in the Transfer editor when the shorter length variable is assigned from the longer length variable . But the error message of “possible loss of precision” is shown when you deploy Server Program. (Select Deploy/Make Server menu) in the Code Generator.

If sp.m0 Type is Unsigned Integer and rp.m0 Type is Integer

rp.m0 = sp.m0;

Transfer statements makes error “possible loss of precision” because sp.m0 is 8byte long type and rp.m0 is stored with 4byte integer type.

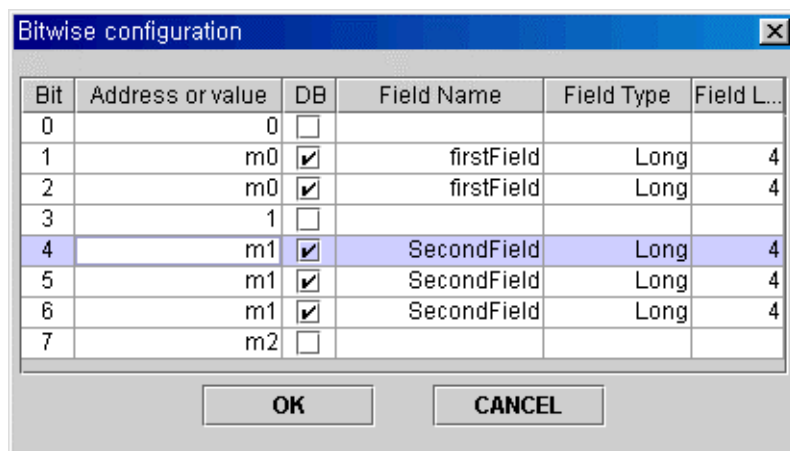
[Little Endian]

1. Big Endian is that the big byte (the biggest byte of the array) ordering, Little Endian is that the little byte (the least byte of the array) ordering. For example, Big Endian system stores value of Hex " 4F52" with "4F52" (If 4F is stored in the address no. 1000, 52 is stored in the address no 1001). Little Endian sytem store the value with "524F.
2. If the Type is Little Endian(LE), when UI Program sends data to Server Program or Server Program sends data to the Device, the byte array is stored with the little Byte ordering

- When the system communicates with the Device, check the Protocol Data whether it has Big Endian or Little Endian and select the data type in the protocol editor.

[Bitwise Configuration]

- If FIXED DATA Protocol element has Bit(Byte) Type, user can set bit by bit in 1Byte
- If Value item is not set value bit, null data is set, If set, "set" is displayed.
- Double click Value item and edit with opening Bitwise Configuration dialog window.



[Fig 2.26 Bitwise configuration Dialog]

- Bit item is from 0 to 7 bit.
- Address or value field can have '0','1' or 'm' + Unsigned Integer. '0','1' is fixed value of the bit, 'm' + Unsigned variable is that bit item is set by UI Program or Server Program
- Check the DB check box whether you use DB field or not, When you store data to DB, set the Field Name, Field Type, Field Length with referring [Set the DB Item]. Data Type is Integer Type. [Fig 2.26] shows when General Panel DBMS item is Access database.
- The value will be byte in Server Program system.
ex) if Byte value is 01010101, Server Program variable value m0 = 00000010(2), m1 = 00000010(2) and m2 = 00000001(1).
- Press OK button and the data is stored, Press CANCEL button and the dialog window is closed without saving the data.
- When you set DB item of Bitwise configuration, Protocol element in the Protocol Edit Table of DB, Field Name, Field Type, Field Length is displayed "set".

(7.4) FILE

Edit Item	Editing	Value	Description
Category	X	FI	Double Click to open Send Condition window
Check Sum	O	true or false	Select Check Box
Symbol	X	I + Unsigned Int.	Given automatically
Type	X	File	File that UI Program sends.
Length	X	s(+ Value +)	Displayed automatically if Value is selected
Value	O	I + Unsigned Int	ex) 'I0', 'I100'
DB	O	true or false	Select Check Box
Field Name	O	field name	Can be edited only when DB element is true. Use String Data Type and store file name that has been sent by UI Program. Refer to [Setting DB related elements] .
Field Type	O	String Field Type	
Field Length	O	field length	
Send Condition Configuration	O	Send even if null Send only if not null	Double click to open Category. When data from the UI Program is null, determine whether the data should be sent to Device.

[Table 2.7 Editing Send Protocol FILE]

(7.5) DELIMITER

Edit Item	Editing	Value	Description
Category	X	DE	
Check Sum	O	true or false	Select Check Box
Symbol	X	de	
Type	X	Byte	
Length	X	1	
Value	O	Byte, Byte Symbol	HEX Byte ex) '0x02' Byte Symbol (refer to [Byte Symbol Table]) ex) 'STX' When registering Delimiter for the first time the Value is unchecked, but when using existing Delimiter previous Value settings are automatically used. If the Value is changed, all of the Value elements of DELIMITER in the Protocol will be changed automatically.
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB
Field Length	X		Can not be stored in the DB

[Table 2.8 Editing Send Protocol DELIMITER]

(7.6) ID

For Receive Protocol only (related with database Position Code and Device Code).

(7.7) CHECK SUM

Edit Item	Editing	Value	Description
Category	X	CS	
Check Sum	X	false	
Symbol	X	cs	
Type	O	Byte	Check Sum value is Exclusive OR of the protocol item selected
		Ascii2	Transfer Byte Check Sum above to Hex Ascii.
		CRC16	16bit Table CRC
Length	X	1 or 2	The length of Check Sum
Value	X	CHECK_SUM	
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB
Field Length	X		Can not be stored in the DB

[Table 2.9 Editing Send Protocol CHECK SUM]

(8). Editing Receive Protocol data

(8.1) FLAG

Edit Item	Editing	Value	Description
Category	X	FL	
Check Sum	O	true or false	Select Check Box
Symbol	X	f + Unsinged Int	Set automatically according to location
Type	O	Byte or Text	Select List Box
Length	X	Unsigned Int	Set automatically according to Type and Value If the Type is Byte, the value is "1" If the Type is Text, the value is the length of text input in Value
Value	O	Byte, Byte Symbol, Text	1) When Type is Byte Hex Byte value ex) '0x02' Byte Symbol (refer [Byte Symbol Table]) ex) 'STX' 2) When Type is Text Text value ex) 'C', 'abcd'
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB

Field Length	X		Can not be stored in the DB
--------------	---	--	-----------------------------

[Table 2.10 Editing Receive Protocol FLAG]

(8.2) VARIABLE DATA

Edit Item	Editing	Value	Description
Category	X	VA	
Check Sum	O	true or false	Select Check Box
Symbol	X	v + Unsigned Int	Created automatically
Type	O	Byte Array	Save Data as String
		Byte Array(integer)	Save Data after Parsing data as integers
		Byte Array(double)	Save Data after Parsing data as double
Length	O	FLAG Symbol, DELIMITER Symbol, FIXED DATA Symbol, end	<p>Since length of VARIABLE DATA is variable, the end of data must be specified.</p> <p>1) In case of FLAG Symbol and DELIMITER Symbols the VARIABLE DATA is until the relevant value is displayed. In this case, the FLAG and DELIMITER should be located immediately after the VARIABLE DATA. ex) 'f0', 'de'</p> <p>2) The FIXED DATA Symbol is used at the end of the VARIABLE DATA. In this case the FIXED DATA is an Integer. FIXED DATA precedes VARIABLE DATA. ex) 'x0'</p> <p>3) Since end is the end of the Protocol Data, no more data can succeed the end of Protocol Data. ex) 'end'</p>
Value	O	v + Unsigned Int	Type Byte Array ex) 'v0', 'v100'
		m + Unsinged Int	Numeric Value ex) 'm0', 'm100'
DB	O	true or false	Select Check Box
Field Name	O	field name	<p>Can be edited only when DB selection is true.</p> <p>If Data is Byte Array, DB Data has String Type, If Data isByte Array(integer), DB Data has Integer Type. If Data isByte Array(double), DB Data has Float Type.</p> <p>Refer to [set DB item] .</p>
Field Type	O	fileld type	
Field Length	O	field length	

[Table 2.11 Editing Receive Protocol VARIABLE DATA]

(8.3) FIXED DATA

Edit Item	Editing	Value	Description
Category	X	FX	
Check Sum	O	true or false	Select Check Box
Symbol	X	x + Unsinged Int	Given automatically
Type	O	Unsigned Byte	0 ~ 255, save as short type in system
		Byte	-126 ~ 125, save as byte type in system
		Short	-32768 ~ 32767, save as short type in system
		Unsigned Short	0 ~ 65535, save as integer type in system
		Short(LE)	Short value , Little Endian
		Unsigned Short(LE)	Unsigned Short value, Little Endian
		Integer	-65536 ~ 65535, save as integer type in system
		Unsigned Integer	0 ~ 4294967295, save as long type in system
		Integer(LE)	Integer value, Little Endian
		Unsign Integer(LE)	Unsigned Integer value, Little Endian
		Float	Floating point value
		Float(LE)	Float, Little Endian
		Double	Double value
		Double(LE)	Double, Little Endian
		Text	String type in the system
		Text(integer)	String is changed to integer
		TextHex(integer)	Hex String is changed to Hex integer
		Text(double)	String is changed to double
		Bit(Byte)	Bit value in 1Byte
Length	O	Unsined Integer	When Type begins with Text, input Text length
	X	Unsigned Integer	The length is fixed according to Type. Impossible to edit.
Value	O	v + Unsigned Int.	Type is Text ex) 'v0', 'v100'
		set or null	Double click value selection to open Bitwise Configuration dialog. Refer to ([Bitwise Configuration])
		m + Unsigned Int.	Numeric Type ex) 'm0', 'm100'
DB	X	true or false	If theType is Bit(Byte), edit from the Bitwise Configuration dialog
	O	true or false	Select Check Box for other types
Field Name	O or X	field name	If the Type is Bit(Byte), edit from the Bitwise Configuration dialog. In other cases, editing is possible only when DB selection is True. If the Type is Text, Data Type is String Type In other cases, it is Integer or Float Data Type. Refer to [DB item].
Field Type		According to Type	
Field Length		field length	

[Table 2.12 Editing Receive Protocol FIXED DATA]

(8.4) FILE

For Send Protocol only.

(8.5) DELIMITER

Edit Item	Editing	Value	Description
Category	X	DE	
Check Sum	O	true or false	Select Check Box
Symbol	X	de	
Type	X	Byte	byte
Length	X	1	1
Value	O	Byte, Byte Symbol	HEX Byte ex) '0x02' Byte Symbol (refer [Byte Symbol Table]) ex) 'STX' . When you change the Value item, all of the DELIMITER in the Protocol element are changed automatically.
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB
Field Length	X		Can not be stored in the DB

[Table 2.13 Editing Receive Protocol DELIMITER]

(8.6) ID

Edit Item	Editing	Value	Description
Category	X	ID	
Check Sum	O	true or false	Select Check Box
Symbol	X	id	
Type	O	Byte	1byte (related with Device Code in the DB)
		Short	2byte (related with Device Code in the DB)
		Integer	4byte (related with Device Code in the DB)
		Text	String (related with Position Code in the DB)
Length	O	Unsigned Int.	Input the length if Type is Text
	X	1, 2, 4	Set automatically according to other types
Value	O	v + Unsigned Int	Text Type ex) 'v0', 'v100'
		m + Unsigned Int	Numeric Type ex) 'm0', 'm100'
DB	O	true or false	Select Check Box
Field Name	O	field name	If the Type is Text the data type is String Type. In other cases it will have Integer Data Type. Refer to [Set DB Item].
Field Type		type	
Field Length		field length	

[Table 2.14 Editing Receive Protocol ID]

(8.7) CHECK SUM

Edit Item	Editing	Value	Description
Category	X	CS	
Check Sum	X	false	
Symbol	X	cs	
Type	O	Byte	Check Sum value is Exclusive OR of the protocol item selected
		Ascii2	Byte Check Sum is changed to Hex Ascii.
		CRC16	16bit Table CRC
Length	X	1 or 2	The length of Check Sum
Value	X	CHECK_SUM	
DB	X	false	Can not be stored in the DB
Field Name	X		Can not be stored in the DB
Field Type	X		Can not be stored in the DB
Field Length	X		Can not be stored in the DB

[Table 2.15 Editing Receive Protocol CHECK SUM]

(9). Protocol Editing Errors List

- Continuous "DE" is not allowed
DELIMITER is set twice consecutively.
- Only one CS field in a protocol definition
Two or more CHECK SUM elements exist in one Protocol.
- No check sum field is selected
When CHECK SUM Category is contained protocol element but Check Sum data is not selected.
- At least 2 check sum fields are required
. Check Sum elements should be more than one.
- Every check sum field should be put prior to the Check sum (CS).
Check Sum data is located after CHECK SUM Category.
- There is(are) check sum field(s) without the Check sum (CS).
When CHECK SUM Category is not in the protocol element but Check Sum data is selected.
- Specify Data value.
When Value editing field is not input.
- Specify Data length.
When Length field is not input.
- No more fields are required
When the length of VARIABE DATA is input as "end" but another Category item succeeds the data.
- Invalid VA-Data length.

When an inadequate value is input for Length.

- VA-length specifying field [LENGTH VALUE] must be placed right behind the VA field

When the length of VARIABLE DATA has DELIMITER or FLAG Category Symbol, but the Symbol does not succeed the VARIABLE DATA.

- Variable Data Relation Length

.Text type FX can't specify VA data length.

.Text (double) type FX can't specify VA data length.

.Float type FX can't specify VA data length.

.Double type FX can't specify VA data length.

.Bit type FX can't specify VA data length.

When the length of VARIABLE DATA is a symbol of FIXED DATA, but the type of FIXED DATA is not an Integer type(Byte , Short).

- VA-length specifying field must be placed prior to the VA field.

When the length of VARIABLE DATA is a symbol of FIXED DATA, but the FIXED DATA does not precede the VARIABLE DATA.

- Data symbol specified by this VA is missing.

When the symbol of DELIMETER, FLAG, or FIXED DATA input as the Length of VARIABLE DATA does not exist in Protocol.

- Wrong data in value field. Value must be "v0 or v1 .."

When a wrong value is input.

- Wrong data in value field. Value must be 0"m0 or m1 .."

When a wrong value is input.

- Value does not match type "[TYPE VALUE]". "m0 or m1" matches "Byte Array(integer) or Byte Array(double)".

- Value does not match type "[TYPE VALUE]". "v0 or v1" matches "Byte Array".

When the Type Value and the type of data does not match.

- Specify the bitwise configuration in the "[VALUE]" field.

If FIXED DATA Type is Bit(Byte), Bitwise configuration is not set or set as null.

- [ValueField] is allowed only in type Text.

When FIXED DATA Type is numeric but the value field is text type.

- Data length mismatch.

Value of Length is incorrect.

- Field name "[VALUE]" is duplicated.

When Field Name is repeated . Also check whether it is overlapped with Bitwise configuration.

- Field name contains space.

When the Field Name contains space.

- Field name is not filled.

When the Field Name is not input.

- Value "[VALUE]" is duplicated.

The value of value field is repeated. Also check whether it is overlapped with Bitwise configuration.

- Wrong data in value field. value must be "\I0 or I1 ..\"

When the suffix of FILE type Value is not I (L).

- Only one ID field in a protocol definition

When two or more ID elements exist in one Protocol.

(10) Editing DB Storing Conditions

Edit DB Storage Condition Statements referring to syntax in the DB Storage Condition Edit window. Users can verify and store DB Storage Conditions by following "5. Protocol Editing and Verification" and "6. Protocol Storing".

- Possible Operators

Symbol	Priority	Operand	Desription	Symbol	Priority	Operand	Desription
()	1	1	Priority	-	4	2	Minus
+	2	1	sign	<<	5	2	left shift
-	2	1	sign	>>	5	2	right shift
~	2	1	bitwise not	<	6	2	less than
!	2	1	logical not	>	6	2	greater than
(byte)	2	1	casting	<=	6	2	less equal
(short)	2	1	casting	>=	6	2	greater equal
(integer)	2	1	casting	==	7	2	equal
(long)	2	1	casting	!=	7	2	not equal
(float)	2	1	casting	&	8	2	bitwise and
(double)	2	1	casting	^	9	2	exclusive or
*	3	2	Multiply		10	2	bitwise or
/	3	2	Divide	&&	11	2	logical and
%	3	2	Remainder		12	2	logical or
+	4	2	Plus				

[Table 2.16 DB Storing Condition Operators]

- Possible Operands

1 m + Unsigned Integer type Protocol Value ex) 'm0' , 'm100'

2 Integer ex) 12, -24, 100L, 0x0b, 0x12341234ffL

3 Real ex) -12.5, 0.

- The results of Statement operations should be boolean type.

(11) DB Storing Condition Error list

- Operand is duplicated.
Operand is duplicated without operator. ex) $m0 > 100 m1$
- Variable "[VARIABLE]" must be "m0 or m1"...
The suffix of value is m but the Protocol value type is m + Unsigned Integer
. ex) $mix > 100$
- Variable "[VARIABLE]" is not found in protocol
[VARIABLE] is not registered as Protocol Value.
ex) If m1000 is not registered when the statements use $m1000 > 100$
- Operand "[OPERAND]" has wrong number format.
OPERAND is not in numeric format. ex) $m100 > 0xfg$
- Operator "[OPERATOR]" is not supported.
When using an Operator not supported. ex) $m100 = 100$
- Operator "[OPERATOR]" has a missing operand.
When the operator does not have the necessary number of operands. ex) $m100 >= 3 +$
- Operator "[OPERATOR]" must have the first operand.
. ex) $m100 > * 2$
- Operator "(" has no right parenthesis in pair.
. ex) $m1 > 2 * (m2 + m3$
- Operator ")" has no left parenthesis in pair.
ex) $m1 > 2 * m2 + m3)$
- This operand is not proper type for operator "[OPERATOR]".
ex) $m100 > (m1 == 100)$
- Operand "[OPERAND]" is left after operation.
When Operands are left after operation.
- There is no result after operation.
When there are no results after operation.
- The result of operation is not boolean.
.ex) $m0 + 100 * m1$

(12). Additional Functions

(12.1) CONVERTER

- When you want to know the value of FLAG or DELIMITER as decimal, hex, or ascii value.
- Click CONVERTER to open the Hex-Dec-ASCII convert window.
- Type Hex value and click Convert to display decimal and Ascii value.
- Type Dec value and click Convert to display hex and Ascii value.

- Type Ascii value and click Convert to display decimal and Ascii value.
- Click Close to close window.

(12.2) IMPORT / EXPORT

- Click EXPORT to open File Save dialog. Select Folder, input File Name, and click SAVE to created Protocol file. Send Protocol file name extension is mdsp and Receive Protocol file name extension is mdrp.
- Click IMPORT to display Open File dialog. Select Folder, file and click Open to display saved Protocol in the Protocol Edit Panel.

.

(12.3) CANCEL

Returns to most recent storage state.

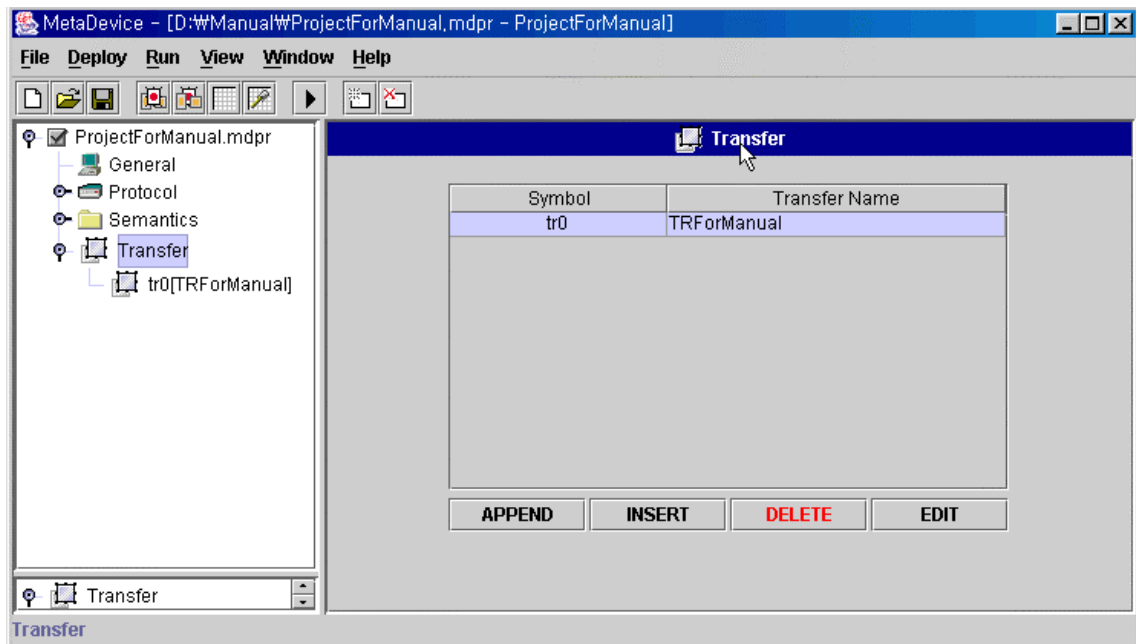
5. Transfer

This section describes how to exchange and transfer protocol from the Transfer Edit Panel. The protocol used for transfer is called the Source Protocol, and the transferred protocol is called the Destination Protocol. The Destination Protocol and Source Protocol can be multiple Protocols. Since Transfer only describes how to transfer the Source Protocol to Destination Protocol, the actual transferring of Protocols is done through the Semantics TRANSFER() Method.

Transfer can be registered or deleted from the Transfer List Panel and can be moved to the Transfer Edit Panel.

5.1 Transfer List Panel

The Panel which can register or delete Transfer and move to the Transfer Edit Panel.



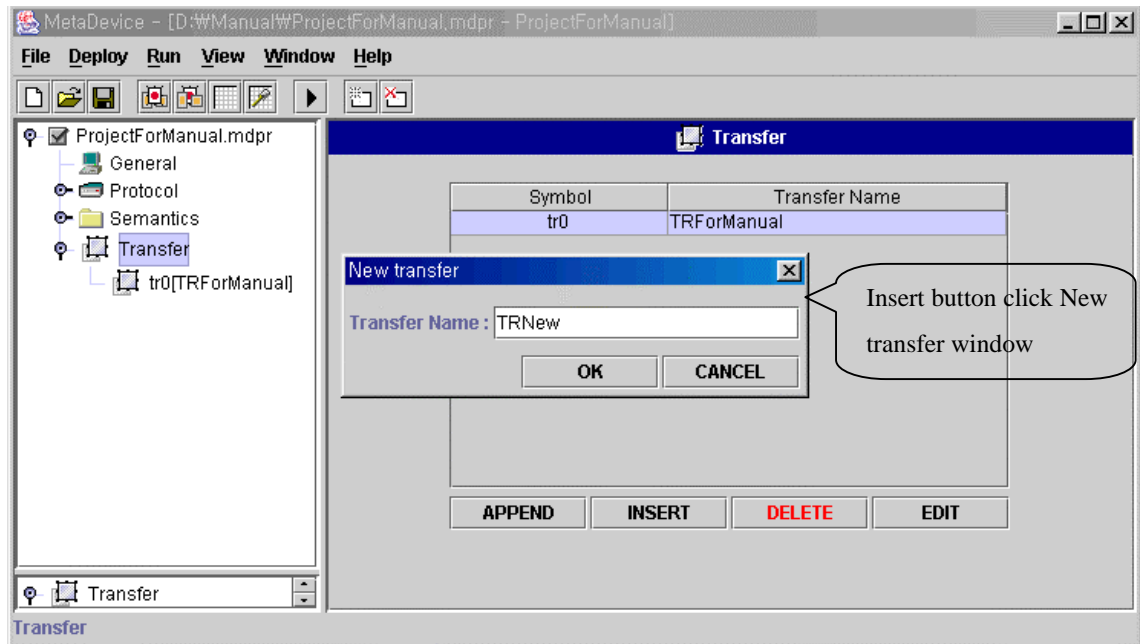
[Fig 2.27 Transfer List Panel]

1. Opening the Transfer List Panel

- 1) Select Transfer Node from the Project Panel to open the Transfer List Panel.
- 2) Select View/Transfer from the menu to open the Transfer List Panel.

2. Registering Transfer

- 1) Click APPEND from the Transfer List Panel to open the New transfer window. Type in the Transfer Name from this window and click OK. The registered Transfer will be added at the end of the List in the Transfer List Panel.
- 2) Click INSERT from the Transfer List Panel to open the New transfer window. Type in the Transfer Name in this window and click OK. The registered Transfer will be added in front of the List in the Transfer List Panel.



[Fig 2.28 New transfer window]

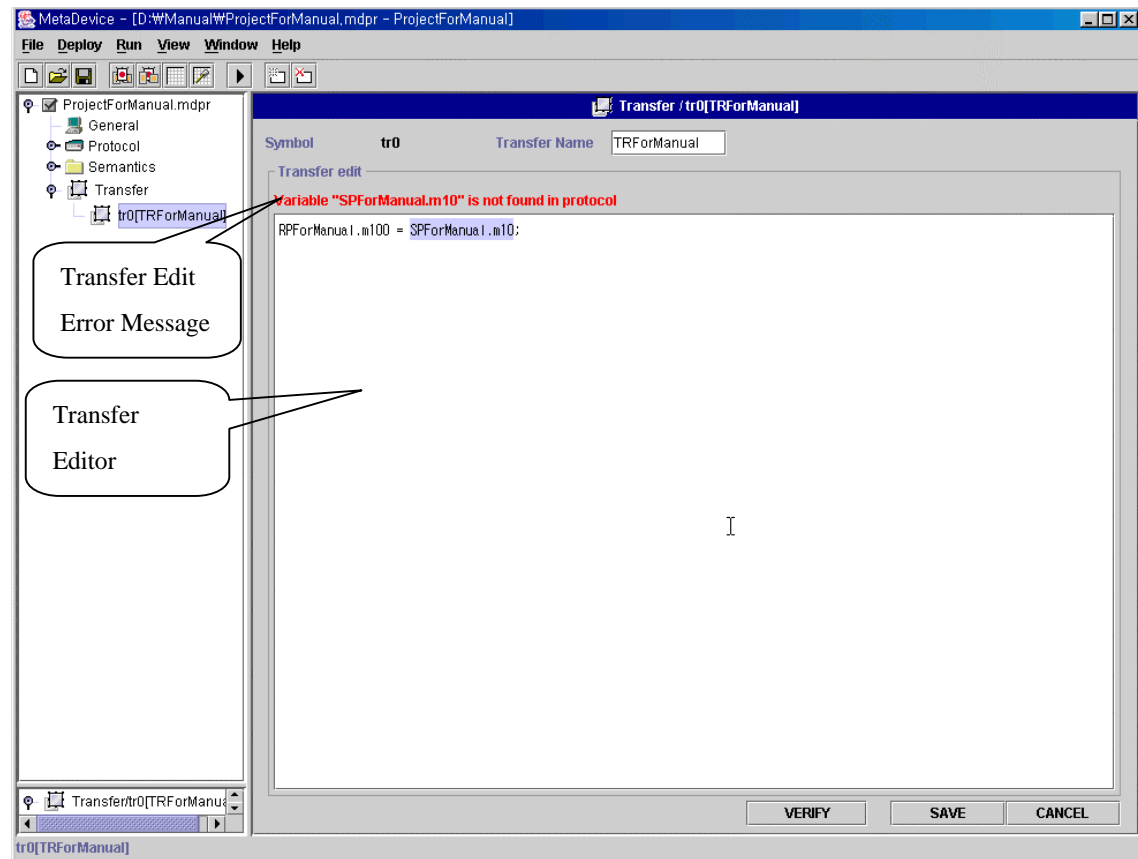
- 3) Transfer Name is selected according to the Server Manager Identifier's Naming Rules. (refer to 3. General Property 3.1 Project Name [Server Manager identifier])

3. Deleting Transfer

- 1) Click DELETE after selecting the Transfer to be deleted from the Transfer List Panel.
- 2) Click YES from the "Delete a row dialog box" to delete the Transfer selected from the Transfer List Panel.

5.2 Transfer Edit Panel

This window edits the registered Transfer. This window edits Transfer according to Transfer Statement Rules from the Transfer Editor and displays editing errors on the Transfer Edit Error Message window.



[Fig 2.29 Transfer Edit Panel]

1. Opening the Transfer Edit Panel

- 1) Select the Transfer Node to be edited from Transfer Node in the Project Panel.
- 2) Select Transfer to be edited from the Transfer List Panel and click EDIT.
- 3) Double click Transfer to be edited from the Transfer List Panel.

2. Verification of Transfer Statements

- 1) Click VERIFY after editing Transfer.
- 2) If there are no errors, a Condition verification successful message will be displayed in the Transfer Edit Error Message window.
- 3) If there are any errors in the Transfer statements, the error message will be displayed in the Transfer Edit Error Message window. Refer to [5. Transfer Editing Error] about the error messages.

3. Saving Transfer Editing

- 1) Click SAVE after finishing Transfer editing.
- 2) If there are no errors after verification, save Transfer.
- 3) If there are errors, check the error messages in the Transfer Edit Error Message window.

4. Editing Transfer

1) Statements

- (1) Numerical expression statements (arithmetic operations)
- (2) Condition statements (if , else if, if)
- (3) Block statements : combines multiple statements into one ({ })

2) Numerical Expressions

(1) Numerical Expression Operators

Symbol	Priority	Operand	Description	Symbol	Priority	Operand	Description
()	1	1	Priority	/	3	2	Divide
+	2	1	sign	%	3	2	Remains
-	2	1	sign	+	4	2	Plus
~	2	1	bitwise not	-	4	2	Minus
(byte)	2	1	Casting	<<	5	2	left shift
(short)	2	1	Casting	>>	5	2	right shift
(integer)	2	1	Casting	&	6	2	bitwise and
(long)	2	1	Casting	^	7	2	exclusive or
(float)	2	1	Casting		8	2	bitwise or
(double)	2	1	Casting	=	9	2	assignment
*	3	2	Multiply				

[Table 2.17 Transfer Numerical Expression Operators]

(2) Operands

[Protocol Name].[Value] : Value is m + 0 or Positive Integer, v + 0 or Positive Integer

ex) SPForManual.m0, RPForManual.v0

Integer : Constant, The end has L or l : long type

ex) 12, -24, 100L, 0x0b, 0x12341234ffL

Real : Constant ex) -12.5, 0.

String : Constant ex) "TestString"

- (3) ;(Semi Colon) : Numerical Expression Statements must end with a semi colon.

3) Condition Statements (if , else if , else)

(1) if statements syntax

if ([boolean expression]) [statements]

(2) else if statements syntax

else if ([boolean expression]) [statements]

- (3) else statements syntax
 - else [statements]
- (4) There is no need for “else if” or “else” when following “ if “ statements
- (5) “else if” or “else” can not be used without “if”
- 4) boolean expression of “if “or “else if”
 - (1) Since DB saving conditions editing in 4.2 Protocol Edit Panel also relates to boolean expressions refer to 10. DB Store Condition
 - (2) Valid Operators

Symbol	Priority	Operand	Description	Symbol	Priority	Operand	Description
()	1	1	Priority	-	4	2	Subtract
+	2	1	Sign	<<	5	2	left shift
-	2	1	Sign	>>	5	2	right shift
~	2	1	bitwise not	<	6	2	less than
!	2	1	logical not	>	6	2	greater than
(byte)	2	1	Casting	<=	6	2	less equal
(short)	2	1	Casting	>=	6	2	greater equal
(integer)	2	1	Casting	==	7	2	equal
(long)	2	1	Casting	!=	7	2	not equal
(float)	2	1	Casting	&	8	2	bitwise and
(double)	2	1	Casting	^	9	2	exclusive or
*	3	2	Multiply		10	2	bitwise or
/	3	2	Divide	&&	11	2	logical and
%	3	2	Remains		12	2	logical or
+	4	2	Add				

[Table 2.18 boolean Numeric Expressions]

- (3) Operands
 - [Protocol Name].[Value] : Value is m + 0 or Positive Integer
 - ex) SPForManual.m0
 - Integer : Constnat Integer, The end suffix is L or l: long type
 - ex) 12, -24, 100L, 0x0b, 0x12341234ffL
 - Real : Constant ex) -12.5, 0.
- (4) The arithmetic results should be a boolean type.
- 5) if, else if or else statements
 - (1) Supports all statements in Transfer editing.
 - (2) Supports Multiple if ... else ... statements.
 - (3) Supports Multiple block statements.
- 6) Block Statements
 - (1) Multiple statements surrounded by {and} to form one statement.
 - (2) Used when putting multiple statements in place where only one statement is allowed.

5. Transfer Editing Errors

- 1) Operand is duplicated.
Occurs when Operand is duplicated without Operators.
ex) RPForManual.m100 = 100 SPForManual.m0;
- 2) Variable "[VARIABLE]" is not found in protocol
Occurs when [VARIABLE] is not registered as Protocol Value.
ex) RPForManual.m1000 = SPForManual.m0; (m1000 is not registered as Protocol Value)
- 3) Operator "[OPERATOR]" is not supported.
Occurs when using Operators that are not supported.
ex) RPForManual.m100 = SPForManual.m0 > 0; ('>' is not supporting.)
- 4) Operator "[OPERATOR]" has a missing operand.
Occurs when the operator does not have sufficient number of Operands.
ex) RPForManual.m100 = SPForManual.m0 +;
- 5) Operator "[OPERATOR]" must have the first operand.
Occurs when Operator does not have first operand.
ex) RPForManual.m100 = * SPForManual.m0;
- 6) Operator "(" has no right parenthesis in pair.
Occurs when parentheses are not closed.
ex) RPForManual.m100 = (SPForManual.m0;
- 7) Operator ")" has no matching "(".
Occurs when ")" does not have its pair "(".
ex) RPForManual.m100 = SPForManual.m0);
- 8) This operand is not proper type for operator "[OPERATOR]".
Occurs when Operand Type is mismatched with operator.
ex) RPForManual.m100 = "abcd"; (Numeric Value can not have string.)
- 9) The result of operation is not boolean.
Occurs when operation results of "if or else if" statements are not boolean.
ex) if (RPForManual.m100)
- 10) Condition is null.
Occurs when boolean expressions of if or else if has not been input.
ex) if ()
- 11) Operand "[OPERAND]" has wrong string format.
Occurs when operand beginning with " does not end in ".
ex) RPForManual.v100 = "a;
- 12) Operand "[OPERAND]" has wrong character format.
Occurs when operand beginning with ' does not end in '.

- ex) RPForManual.m100 = 'a;
- 13) Operand "[OPERAND]" has wrong transfer format.
Occurs when wrong Protocol Names, wrong Transfer Statement Operand rules, or unsupported Tokens are input.
ex) RPForManual.m100 = 100; (RPForManual wrong Protocol Name.)
- 14) [VALUE] is not a value of a protocol.
Occurs when the Protocol Value does not exist.
ex) RPForManual.m1000 = SPForManual.m0; (m1000 is not defined in the protocol element.)
- 15) [VALUE] is not a assignment("=").
Occurs when arithmetic symbol (=) is missing.
ex) RPForManual.m100 * SPForManual.m0;
- 16) if statement format is wrong.
Occurs when "if "statement does not follow statement rules.
- 17) "(" has no right parenthesis in pair.
ex) if (RPForManual.m100 > 0 {
- 18) Statement can not begin with "[TOKEN]"..
ex) RPForManual.m100 = SPForManual.m0
* RPForManual.m100 = SPForManual.m0
- 19) "else if" statement should follow if or else if statement.
Occurs when "else if " statements are used without " if ".
- 20) "else" statement should follow if or else if statement.
Occurs when "else" statements are used without "if".
- 21) "[CONDITION]" has no condition clause.
Occurs when if or else if statements can not find boolean expressions.
ex) if RPForManual.m100 > 0) {
- 22) "{" has no right brace in pair.
Occurs when there are no "}" in the block statements.
- 23) "possible loss of precision"
There are no verification errors in the Transfer editor when shorter length variables are assigned from the longer length variables . But when deploying the Server Program (Select Deploy/Make Server menu), the error message "possible loss of precision" is shown in the Code Generator. If sp .m0 Type is Unsigned Integer and rp.m0 Type is Integer
- rp.m0 = sp.m0;**
- Transfer statements errors as "possible loss of precision" occur because sp.m0 is 8byte long type and rp.m0 is stored with 4byte integer type.
- To solve this problem, use Casting for explicit assigning.
- ex) When the Category of RPForManual Protocol m100 Value is FIXED DATA and Type is

Byte, and the m0 Value of the SPForManual Protocol is FIXED DATA and Type is Integer, the following statement will make “possible loss of precision” error in Code generator.

RPForManual.m100 = SPForManual.m0;

To avoid this, use explicit casting type.

RPForManual.m100 = (byte) SPForManual.m0;

Errors can be avoided using this method but be careful and aware of data loss risks.

24) Data Transferring Errors

- (1) If the Category of Destination Protocol Value is FIXED DATA and Type is Text(integer), TextHex(integer), Text(double), or if the Category is VARIABLE DATA and Type is Byte Array(integer), Byte Array(double) , the Destination Protocol Value are properly allotted. But be aware of errors when reading Protocol Data from the UI Program.
- (2) If the Destination Protocol Value is the Receive Protocol taken from the UI Program, errors will occur during transfer if all Protocol Values are not assigned.

6. Canceling Transfer Edit

Click CANCEL while editing to return to the conditions saved most recently.

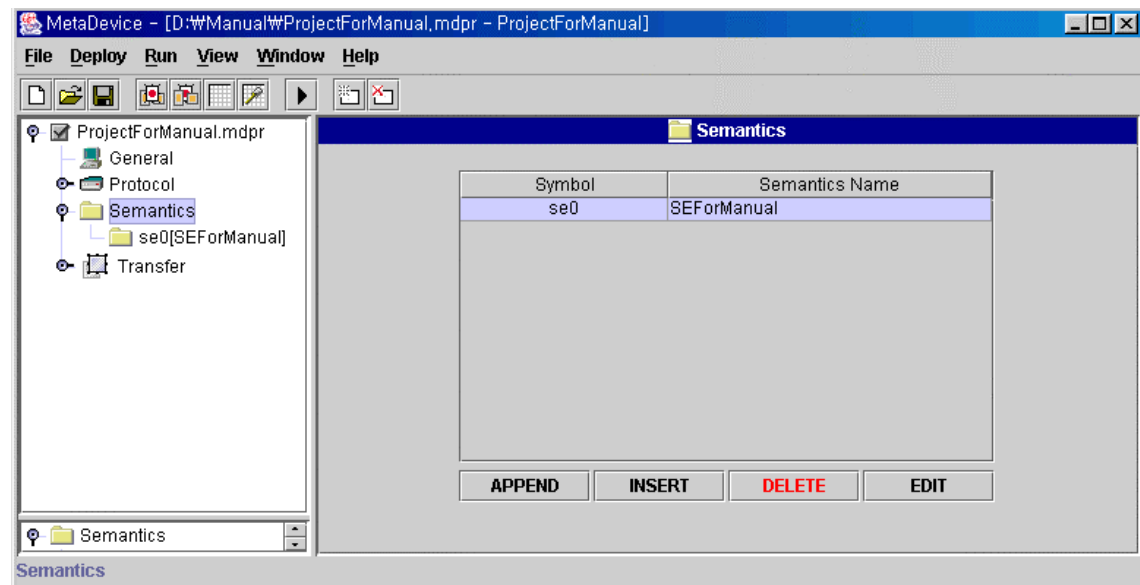
6. Semantics

The Semantics Edit Panel allows the user to specify how the server program connected to the device should work. The Semantics is informed to a server program by inserting the Semantics Name at se_name field of the hd Table, in one of the Admin Tables. If the user inserts an unregistered semantics name in se_name field, the error message “[MAC1][MAC2][MAC3] machine uses not available semantics.” will be displayed when the server program starts. So, the user has to carefully insert the semantics name registered in the server manager project.

The User can register or delete Semantics in the Semantics List Panel and move to the Semantics Edit Panel.

6.1 Semantics List Panel

This is where the user can register or delete Semantics as well as move to the Semantics Edit Panel



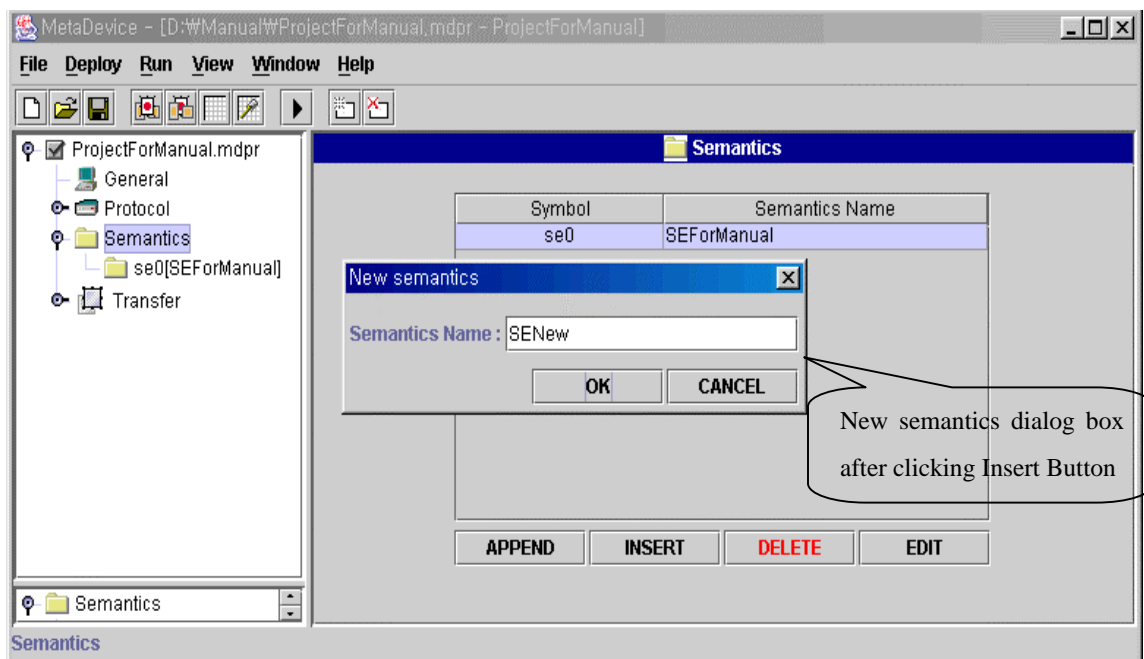
[Fig 2.30 Semantics List Panel]

1. Opening the Semantics List Panel

- 1) User can open the Semantics List Panel by selecting the Semantics Node in the Project Panel.
- 2) User can open the Semantics List Panel by selecting the View/Semantics menu.

2. Registering Semantics

- 1) A new semantics dialog box opens by clicking APPEND in the Semantics List Panel. New Semantics can be registered and appended to the Semantics List at the Semantics List Panel by inserting a Semantics Name and clicking OK in the New semantics dialog box.
- 2) A new semantics dialog box opens by clicking INSERT in the Semantics List Panel. The After inserting the Semantics name and clicking OK, the registered Semantics will be added in front of the selected Semantics in the Semantics List Panel.



[Fig 2.31 New semantics dialog box]

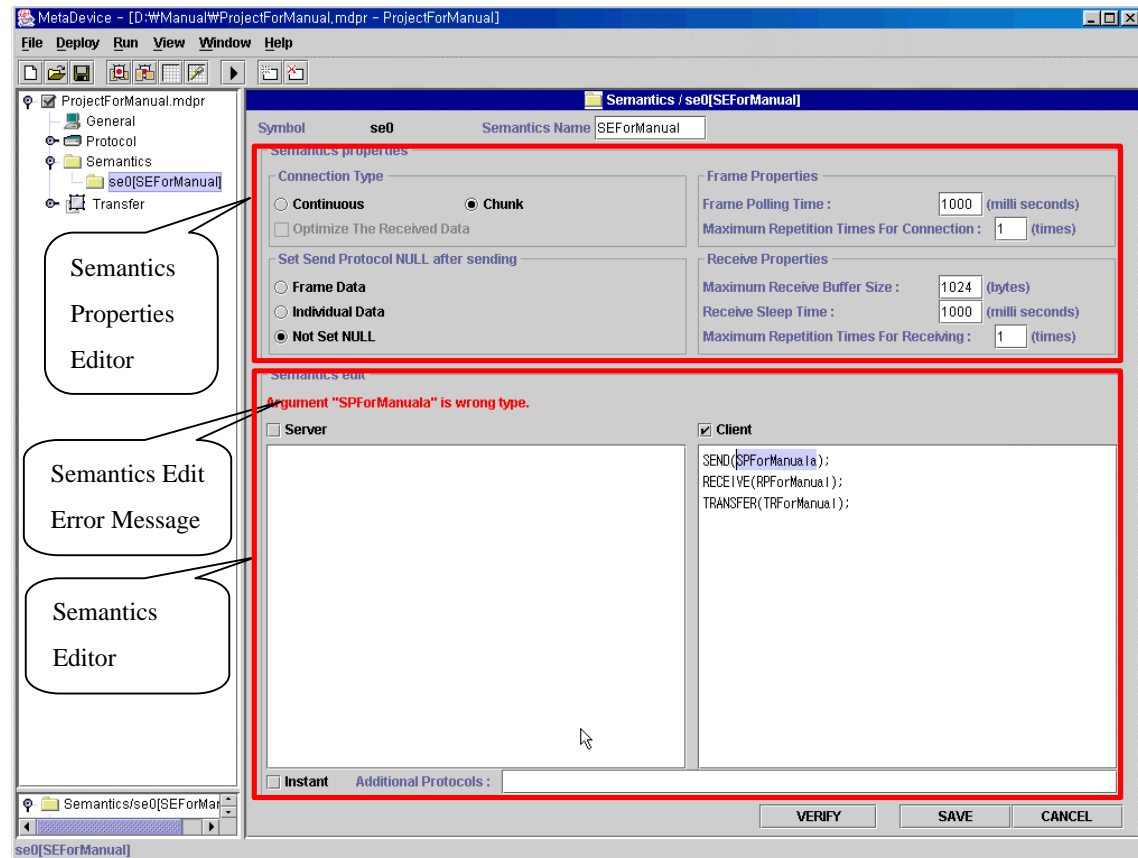
- 3) The Semantics Name should be named according to Server Manager identifier naming rules. (Refer to 3. General Property 3.1 Project Name [Server Manager identifier])

3. Deleting Semantics

- 1) Select the Semantics to be deleted and then click DELETE.
- 2) The selected Semantics will be deleted from the Semantics List Panel by clicking Yes from the Delete a row dialog box.

6.2 Semantics Edit Panel

This is where the user can edit registered Semantics. Semantics Properties can be edited from the Semantic Properties Editor, and Semantics can be edited from the Semantics Editor according to Semantics Statement rules. If Semantics contains some errors, error messages will be displayed in the Semantics Edit Error Message window.



[Fig 2.32 Semantics Edit Panel]

1. Opening the Semantics Edit Panel

- 1) Select Semantics Node to be edited from the Project Panel.
- 2) Click EDIT after selecting the Semantics to be edited from the Semantics List Panel.
- 3) Double Click the Semantics to be edited from the Semantics List Panel.

2. Verifying the Semantics Contents

- 1) Click VERIFY after editing Semantics at Semantics Editor.
- 2) If there are no errors, the 'Semantics verification successful' message will be displayed in the Semantics Edit Error Message window.
- 3) If there are some errors in editing Semantics, specific error messages will be displayed in the Semantics Edit Error Message window. Errors and subsequent error messages is described

in detail in [6. Semantics Editing Error].

3. Storing Semantics contents

- 1) Click SAVE after editing Semantics.
- 2) Verify the Semantics contents and then store the changed Semantics content without editing errors.
- 3) An Error Message will be displayed in the Semantics Edit Error Message window if there are any errors.

4. Editing Semantics Properties

1) Connection Type

- (1) Continuous : Socket Connection between device and server program continues connection even after completing a frame of Semantics. (This option can be set only when Semantics works as Server or Client. Even though it can be set when server program works as Sever/Instant, the server program will not work with contiuous connection in the current version of MetaDevice. It is recommended to set Chunk as Connection Type when server program works as Client/Server.)
- (2) Chunk : Socket Connection between device and server program is closed after completing a frame of Semantics.
- (3) Optimize The Received Data : When the server program parses data received from device, and if a part of the received data matches the expected Receive Protocol , the rest of the received data will be discarded without being parsed any more. This option is used when the Device continuously sends data to the Server Program more often than when the Server Program receives data(This option can be set only when connection type is set as Continuous.)

2) Set Send Protocol NULL after sending

- (1) Send Protocol set as NULL will not send to device even with send command(SEND) of Semantics.
- (2) Frame Data : All Send Protocols which are used in Semantics are set as NULL after completing a frame of Semantics..
- (3) Individual Data : If data has been sent successfully from the Server Program to device, set Send Protocol as NULL if no errors had occurred.
- (4) Not Set NULL : Semantics does not set any Send Protocol as NULL. Server program always sends the most recent Send Protocol data which has been received through UI Program.
- (5) If Send Protocol consists only of constants (Elements whose type is one of FLAG, DELIMITER, CHECK SUM), the data will be sent to device regardless of property

settings. (It works as Not Set NULL option.)

3) Frame Properties

- (1) Frame Polling Time : The period between completion of a frame of Semantics and start of the next frame of Semantics is set in milli seconds. (This option does not affect Semantics when server program does not work as Client and connection type is Chunk or when server program works only as Instant.)
- (2) Maximum Repetition Times For Connection : If Connection Type is Chunk, the socket between device and server program has to be connected when a new frame starts. Then, the number of times the server program has to try to connect to the socket is set at this property.

4) Receive Properties

- (1) Maximum Receive Buffer Size : Buffer size used for server program in receiving data from device.
- (2) Receive Sleep Time : Set in milliseconds, the waiting period of the server program for the data from the device to arrive.
- (3) Maximum Repetition Times For Receiving : Specifies the number of times the server program attempts to receive data from device.
- (4) In cases where the server program succeeds in receiving data from device, the server program will try to receive data once more for Receive Sleep Time to confirm that the device is not sending anymore data.
- (5) When the device is not sending anymore data to the server program, the period during which the server program waits is Receive Sleep Time X Maximum Repetition Times For Receiving.

5. Editing Semantics

1) Setting type of Semantics

- (1) User can specify the type of Semantics by selecting the Server Check Box, Client Check Box, and Instant Check Box.
- (2) Server Check Box : Semantics can be set to work as Server. If the server program works as a Server, it waits for the device to connect and starts to communicate after receiving data from the device. The Server Semantics Editor specifies how the server program works.
- (3) Client Check Box : Semantics can be set to work as Client. If the server program works as a Client, it tries to connect to the device and then starts to communicate by sending data to device if it succeeds in connection. The Client Semantics Editor specifies how the server program works.
- (4) Instant Check Box : The Server program deals with Instant Message from UI Program.

The priority of Instant Message is lower than Server but higher than Client.

2) Supported Statements

- (1) Expression Statement: SEND, RECEIVE, TRANSFER function
- (2) Selection Statement : IF, ELSE IF, ELSE, if, else if, if
- (3) Block Statement : Multiple statements enclosed with { and }

3) Expression Statements

(1) SEND function

Prototype : SEND(Send Protocol Name);

Function : Server program sends Send Protocol data to device.

(2) RECEIVE function

Prototype : RECEIVE(Receive Protocol Name{, Receive Protocol Name...});

Argument can be more than Receive Protocol Name. _MAC, Receive Protocol constant can be also be a Receive Protocol Name.

Function : Server program receives Receive Protocol data from device.

(3) TRANSFER function

Prototype1 : TRANSFER(Transfer Name);

Function : Server program transfers data between protocols in device related to Semantics.

Prototype2 : TRANSFER(Transfer Name, MAC Number String);

MAC Number String : String constant Ex) "04:08:E0"

Function : Server program transfers source protocol data of the current device to the destination protocol data of the device designated by MAC Number String.

Prototype3 : TRANSFER(Transfer Name, MAC Number String, Send Protocol Name);

Function : After transferring the source protocol data of the current device to the destination protocol data of the device designated by MAC Number String, the server program executes instant message commands whose input data is the send protocol data designated by Send Protocol Name and whose destination device is the device designated by MAC Number String.

Prototype4 : TRANSFER(Transfer Name, MAC Number String, Send Protocol Name, Receive Protocol Name);

Function : After transferring the source protocol data of the current device to the destination protocol data of the device designated by MAC Number String, the server program executes instant message commands whose input data is the send protocol data designated by Send Protocol Name, whose output is the receive protocol data designated by Receive Protocol Name and whose destination device is the device designated by MAC Number String.

4) IF, ELSE IF, ELSE Selection Statements

- (1) IF selection statement follows RECEIVE function. It is used to inspect which receive protocol data the RECEIVE function receives or at which state the RECEIVE function is at.
- (2) IF structure
IF ([Receive Protocol expression]) [statement]
- (3) ELSE IF structure
ELSE IF ([Receive Protocol expression]) [statement]
- (4) ELSE structure
ELSE [statement]
- 5) Receive Protocol expression of IF, ELSE IF
 - (1) Available operator
|| : logical or
 - (2) Available operand
Receive Protocol Name used in RECEIVE function as the argument (including Receive Protocol Constant, _MAC) and data receiving state constant(NO_DATA_RECEIVE, PARSE_ERROR, CS_ERROR, ALARM_RECEIVE)
 - (3) NO_DATA_RECEIVE : When server program does not receive any data from the device through the RECEIVE function.
 - (4) PARSE_ERROR : When received data does not match any Receive Protocols that are described in the argument of the RECEIVE function.
 - (5) CS_ERROR : When received data matches one of the Receive Protocols, but Check Sum is not correct.
 - (6) ALARM_RECEIVE is a constant for the next version of MetaDevice. Therefore, it is not supported at this version.
- 6) if, else if, else selection statements
 - (1) if, else if, else selection statements have to be inside IF, ELSE IF selection statement. They are used to inspect the value of the receive protocol data which is matched by IF, ELSE IF selection statements. They can be used when only one receive protocol is used in the Receive Protocol IF, ELSE IF selection statements.
 - (2) if structure
if ([boolean expression]) [statement]
 - (3) else if structure
else if ([boolean expression]) [statement]
 - (4) else structure
else [statement]
- 7) boolean statement of if, else if
 - (1) Available operator
Refer to 5. Transfer 5.2 Transfer Edit Panel 4. Transfer Editing 4) boolean expression

of if, else if

(2) Available operand

[Receive Protocol Name].[Value] : Receive Protocol Name has to belong to Receive Protocol expression of IF or ELSE IF selection statements, and the Value format has to be m + 0 or positive integer or v + 0 or positive integer.

EX) RPForManual.m0

Integer : constant integer, if it ends with L or l, it is long type.

EX) 12, -24, 100L, 0x0b, 0x12341234ffL

Real : constant EX) -12.5, 0.

(3) The result type of operation has to be boolean value.

8) IF, ELSE IF, ELSE, if, else if, else Statements

(1) All statements supported by Semantics editing are possible.

(2) Multiple statements formed of block statements are also possible.

9) Block statements

(1) Multiple statements enclosed with { and }

(2) It can be used to put multiple statements where only one statement is allowed.

10) Additional Protocols

(1) The user can specify protocols which are used at instant message or Transfer but not used in SEND or RECEIVE functions of Semantics Statements.

(2) Multiple protocols separated with , can be used.

EX) SPNew, RPNew, SPTransfer

6. Semantics Editing Errors

1) Statement can not begin with "[TOKEN]".

When Semantics does not start with an expression.

2) Function "[FUNCTION]" has no argument.

When the argument is not recognized. EX) SEND (SPForManual);

3) Function "[FUNCTION]" has no statement terminator ";".

When a function does not end with ";". EX) SEND (SPForManual)

4) Function "[FUNCTION]" has wrong formatted argument.

When function Prototype does not match the statement.

5) Argument "[ARGUMENT]" is not the string type.

6) Argument "[ARGUMENT]" is not the character type.

7) Argument "[ARGUMENT]" is not the number type.

8) Argument "[ARGUMENT]" is wrong type.

9) Argument "[ARGUMENT]" is duplicated.

When the Receive Protocol Name is duplicated in the RECEIVE function.

- 10) "(" has no right parenthesis in pair.
EX) SEND (SPForManual;
- 11) "[CONDITION]" has no condition clause.
When Receive Protocol expression is not found at IF, ELSE IF statements or when boolean expression is not found at if, else if statements. EX) IF RPForManual || _MAC) {
- 12) "[CONDITION]" has no statement.
When IF, ELSE IF, ELSE, if, else if, else selections have no statements.
EX) IF (RPForManual || _MAC) {}
- 13) "{" has no right brace in pair.
When block statement has no right brace "}" in pair.
- 14) "[SEND PROTOCOL]" is not SEND PROTOCOL.
When Send Protocol Name argument of SEND function is not a registered Send Protocol.
- 15) SEND statement is not allowed at HOST server.
When Server Statement starts with SEND function. If server program works as a server, the communication begins by receiving data from device.
- 16) RECEIVE statement should follow SEND statement at HOST client.
When the statement starts with RECEIVE function without SEND function in Client or selection statements. If server program works as a client, the communication starts by sending data to device. It applies to the selection statement.
- 17) "[MAC]" is not a proper MAC number for transfer.
When the format of MAC Number String argument at TRANSFER function is wrong.
- 18) "[RECEIVE PROTOCOL]" is duplicated.
When the Receive Protocol of the Receive Protocol expression at an IF selection statement is duplicated.
- 19) IF statement should follow RECEIVE statement.
When IF selection statement is used without RECEIVE function.
- 20) IF statement should not contain receive protocol not specified at RECEIVE statement.
When a receive protocol which is not specified at a RECEIVE function is used as Receive Protocol Name in the receive protocol expression of an IF selection statement.
EX) RECEIVE(_MAC);
IF (RPForManual) {...}
(RPForManual is used as receive protocol name at the receive protocol expression of IF selection statement without being specified at RECEIVE function argument.)
- 21) ELSE IF statement should follow IF or ELSE IF statement
When ELSE If selection statement is used without IF or ELSE IF.
- 22) ELSE statement should follow IF or ELSE IF statement.
When ELSE selection statement is used without IF or ELSE IF.

- 23) "if" statement should belong to IF or ELSE IF statement.

When if selection statement does not belong to IF, ELSE IF.

- 24) "[PROTOCOL]" in condition of if statement does not belong to protocols related to IF statement.

When a receive protocol is not used at the receive protocol expression of an IF selection statement .

```
EX) IF(RPForManual) {
    if (RPNew.m0 > 100 { ...}
}
```

(RPNew receive protocol at the boolean expression of if statement is not specified at the Receive Protocol expression of IF statement.)

- 25) "else if" statement should follow if or else if statement.

When else if selection statement is used without if or else if.

- 26) "else" statement should follow if or else if statement.

When else selection statement is used without if or else if.

- 27) There are more than two protocols related to IF statement.

When the receive protocol expression of IF or ELSE IF statement to which an if statement belongs to has more than two receive protocols.

```
EX) IF (RP1 || RP2) {
    if (RP1.m0 > 100) { ...
```

(When the receive protocol expression of If statement has two receive protocols – RP1 and RP2, if or else if statement can not be used.)

- 28) "else if" statement should follow if or else if statement.

When else if statement is used without if or else if statement.

- 29) "else" statement should follow if or else if statement.

When else statement is used without if or else if statement.

- 30) Addition protocols format is wrong. Enter the protocol name seperated by ",".

- 31) "[PROTOCOL]" protocol is not registered.

When protocol specified at Additional Protocols Edit window is not registered.

- 32) Refer to 5. Transfer 5.2 Transfer Edit Panel 5. Transfer Edit Error about boolean expression errors of if or else if selection statement.

7. Canceling Semantics Editing

User can return to the Semantics most recently saved by clicking CANCEL.

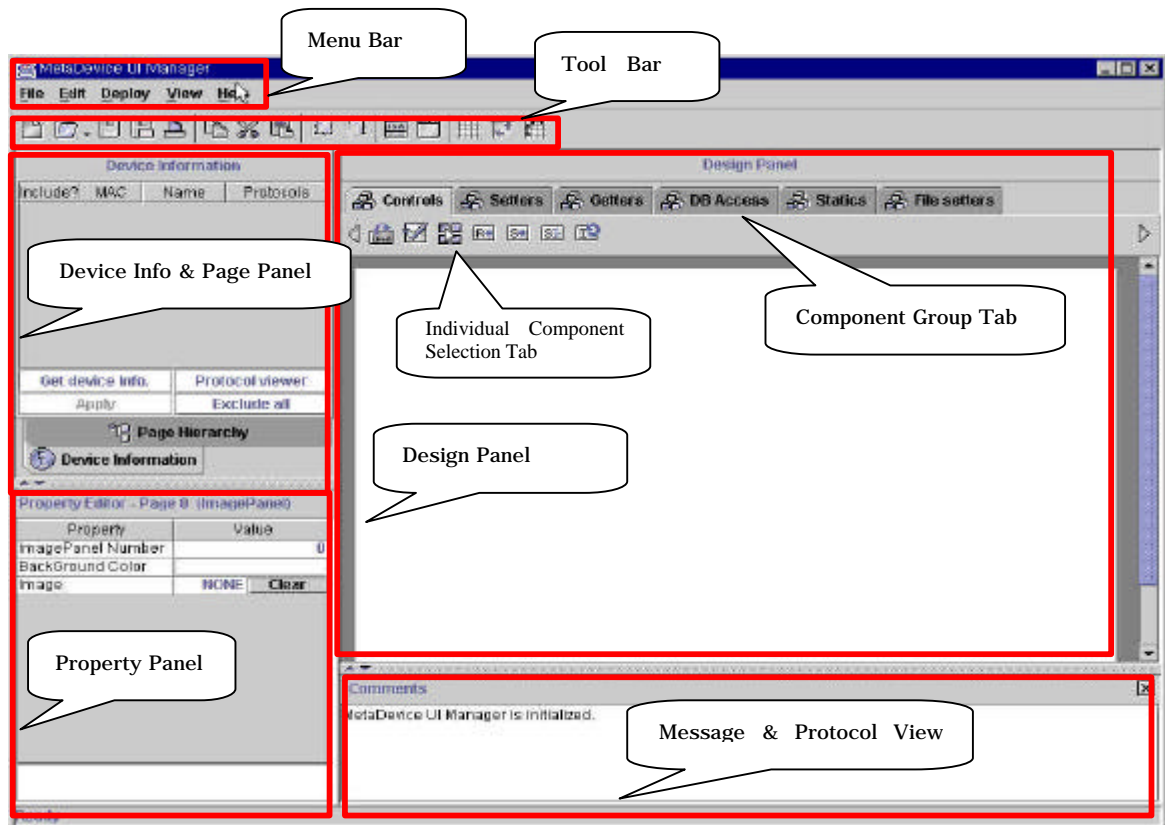
Part III UI Manager

1. UI Manager Overview

1.1 Role of UI Manager

The UI Manager supports interface functions between the Server and User, allowing easy construction of GUI through pre-defined Components and their related Properties.

The UI Manager consists of “Menu Bar”, “Tool Bar”, “Device Info & Page Panel”, “Property Panel” and “Design Panel,” as in [Fig 3.1]. The Design Panel has a “Component Group Tab” which can be selected as Component Group Category and “Individual Component Selection Tab” which can be selected as a real Component.



[Fig 3.1 UI Manager Initial Window]

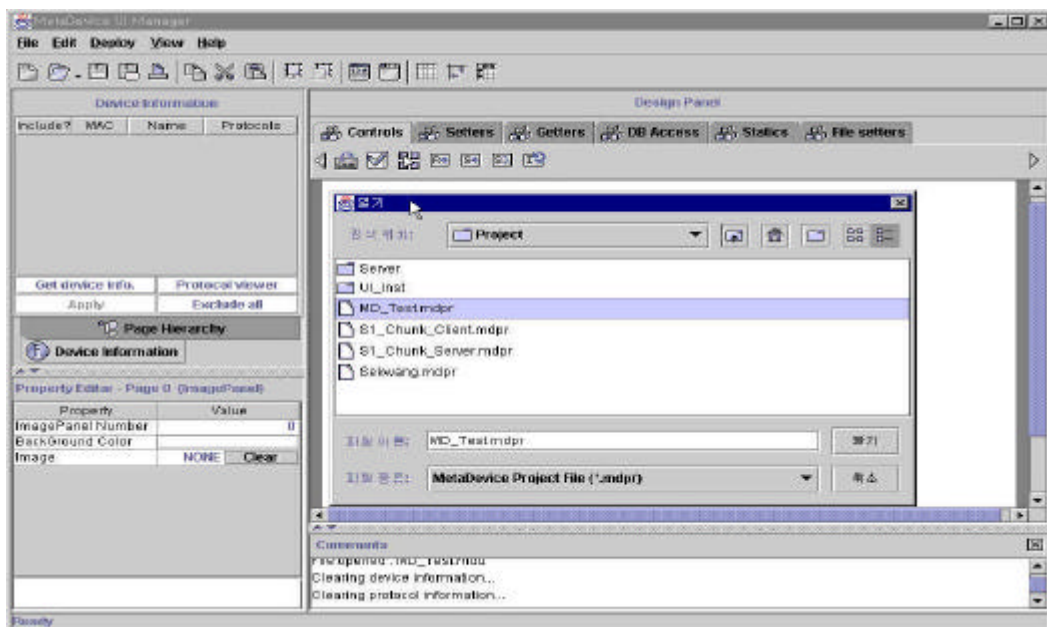
- Menu Bar : MDU File Controlling Bar, including functions such as MDU File Open/Save, UI Component Applet/Application Deployment.
- Tool Bar : Component Handling Bar including partial functions of Menu Bar, and Component Pop Up, Pop Down / Cut & Paste functions while designing GUI.
- Device Info & Page Panel : The Panel which displays information of MDPR File created in the Server Manager and Design Panel information while designing GUI.

- Property Panel : Input/modifying Panel of selected Component Property (displayed when double clicking component).
- Message & Protocol View Panel : The Panel displaying Server Protocol information and Component processing Messages during design.
- Design Panel: Panel designing window that will be actually displayed during Run-Time. Includes “Component Group Tab” and “Individual Component Selection Tab”.
- Component Group Tab: Component Group is divided into six parts. By selecting each Component Group, the Component Selection Tab displaying each Individual Component Selection Item can be seen.
- Individual Component Selection Tab: Window selecting Component to be input in Design Panel.

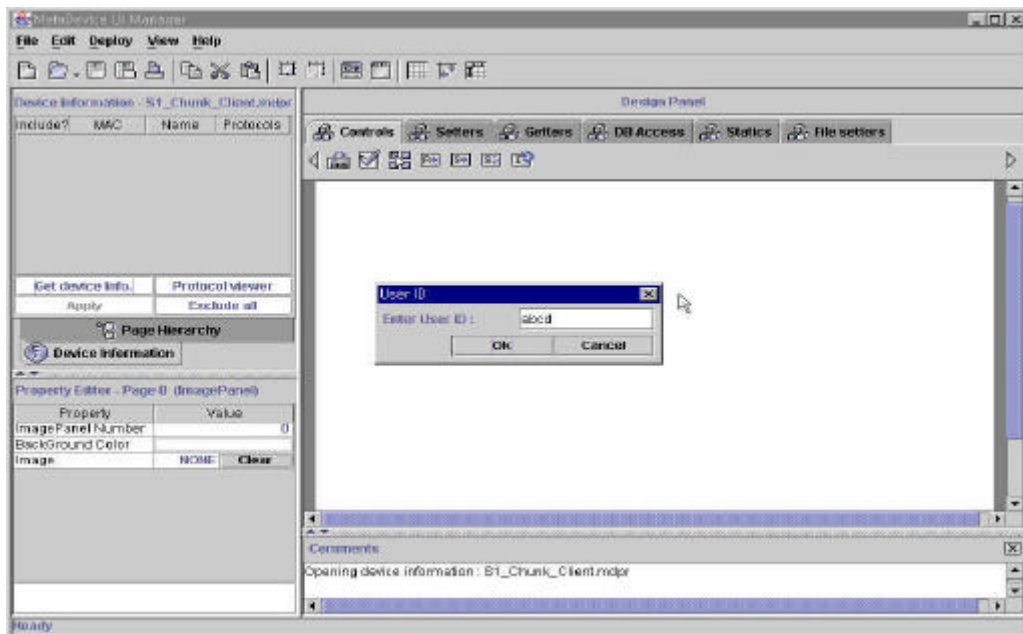
After selecting Component and location of Design Panel, the actual Component will be shown in the Design Panel.

1.2 Notes and Method of Usage

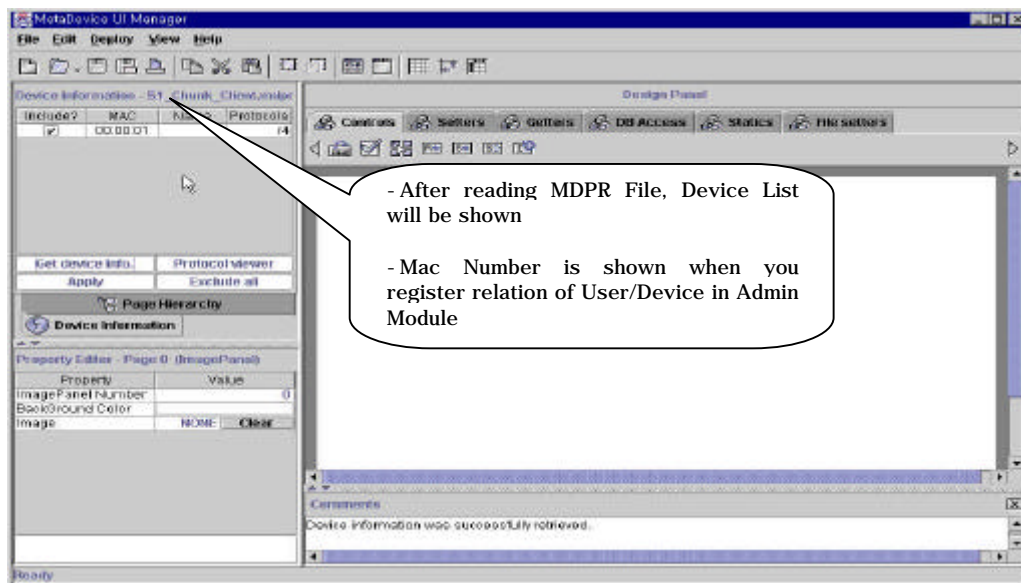
Since the UI Manager provides Interface between Server and User it is necessary to read MDPR File information saved in the Server Manager. The program to be deployed by the UI Manager communicates with the Server Program. Determine which device should be used when getting MDPR information and User information by clicking “Device Information”. [Fig 3.2.1] ~[Fig 3.2.3] shows how select MDPR and User information.



[Fig 3.2.1 Read MDPR File]



[Fig 3.2.2 Get User Information]



[Fig 3.2.3 Window after reading MDPR]

The most important aspect when designing GUI is inputting Device information(MAC Number), Protocol Symbol(Must be symbol Name not Protocol Name) and the variables defined in Protocol to the Component Property. (Please refer to Part I, Section 3.5 “Server Protocol Data and UI Manager Component Data Interdependency”).

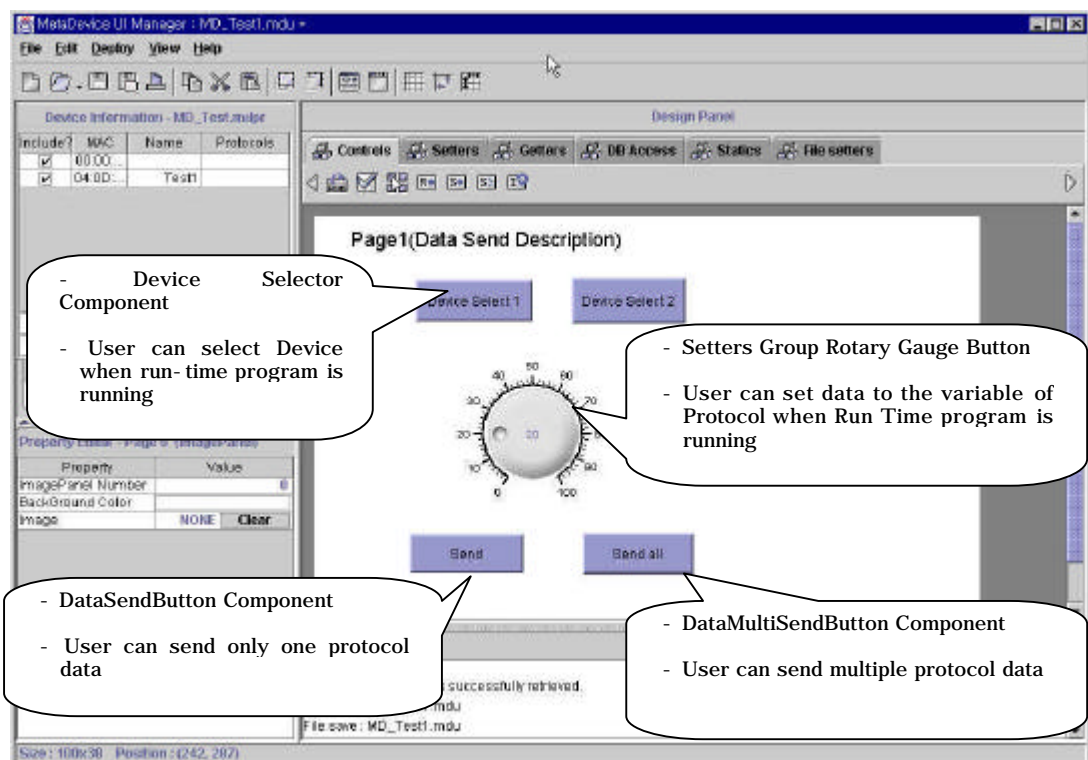
(1) Data Send

(1.1) General Description

For the Data to communicate with the Device after it is sent to the Server, basic components are needed. These Components are the “Device Selector” Component, the “Setters” Component defining variables, and the “Send Button” Component sending data to the Server. The User can set data to a certain Device during Run-Time GUI. The user can first select the Device and assign value to the “Setters” Component, and then click Send to send commands to the Device. [Fig 3.3] is an example of UI Design for sending commands to Multiple Device.

The User can send commands to two Devices and send assigned values to a selected device by using the “Rotary Gauge Component”. Components sending commands to the Server are composed of two types, including one that sends single protocols and one that sends multiple protocols simultaneously.

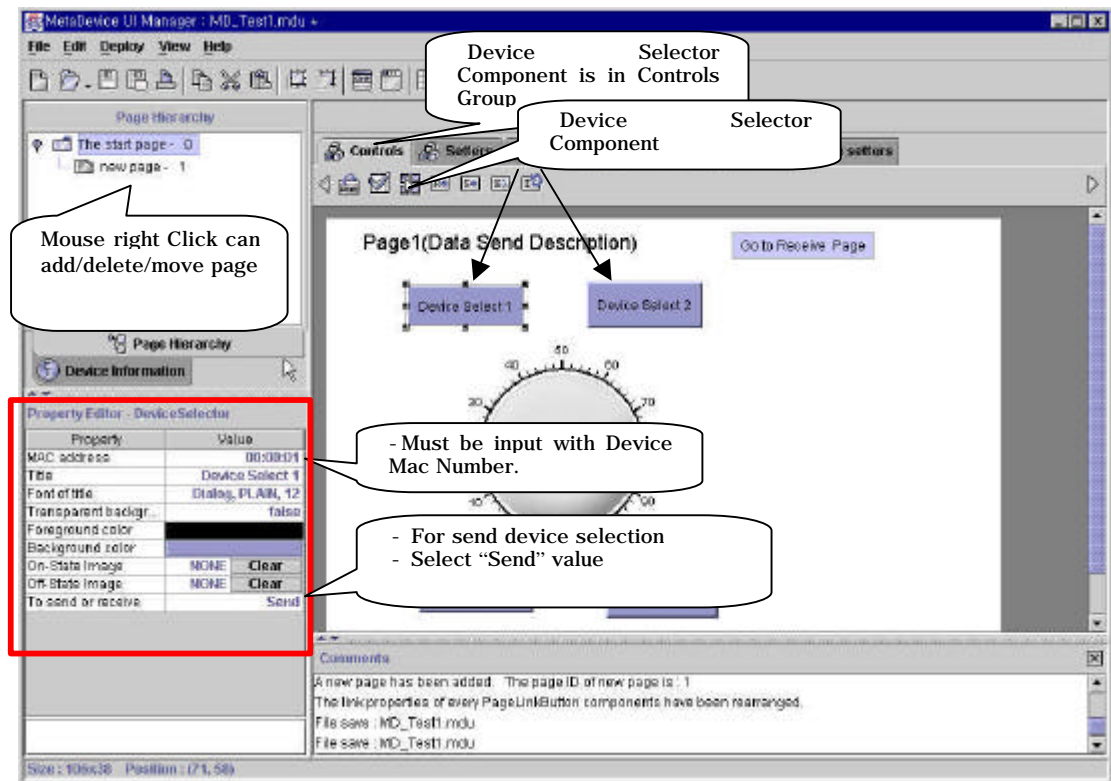
The “DataSendButton” Component sends only one Protocol to the Server and “DataMultiSendButton” Component can send all of the defined Protocols to the Server. When sending data, the user can select multiple Devices and send data to the Server simultaneously. On the other hand, when receiving data from Server, the user can only select single Device because multiple device data cannot be displayed in one type of Component.



[Fig 3.3 Basic Component Example for Data Sending to the Server]

(1.2) Component Property Description

- DeviceSelector Component for sending Data



[Fig 3.4.1 Device Selector Component]

.MAC address : Input the Mac Number registered in the Server and related with User in the Admin module. It is not absolutely necessary to input a Mac Number related with the User, but if you input a Mac Number without any relations to the User, the Device can not be selected during Run Time.

Input as follows : "00:00:01".

.Title : Text to be displayed in Button.

.Font of title : Set Title Font

.Transparent background :

true : The Color is not shown because the Button is transparent

false: The Color is shown because the Button is not transparent.

.Foreground color : Set Title Text Color

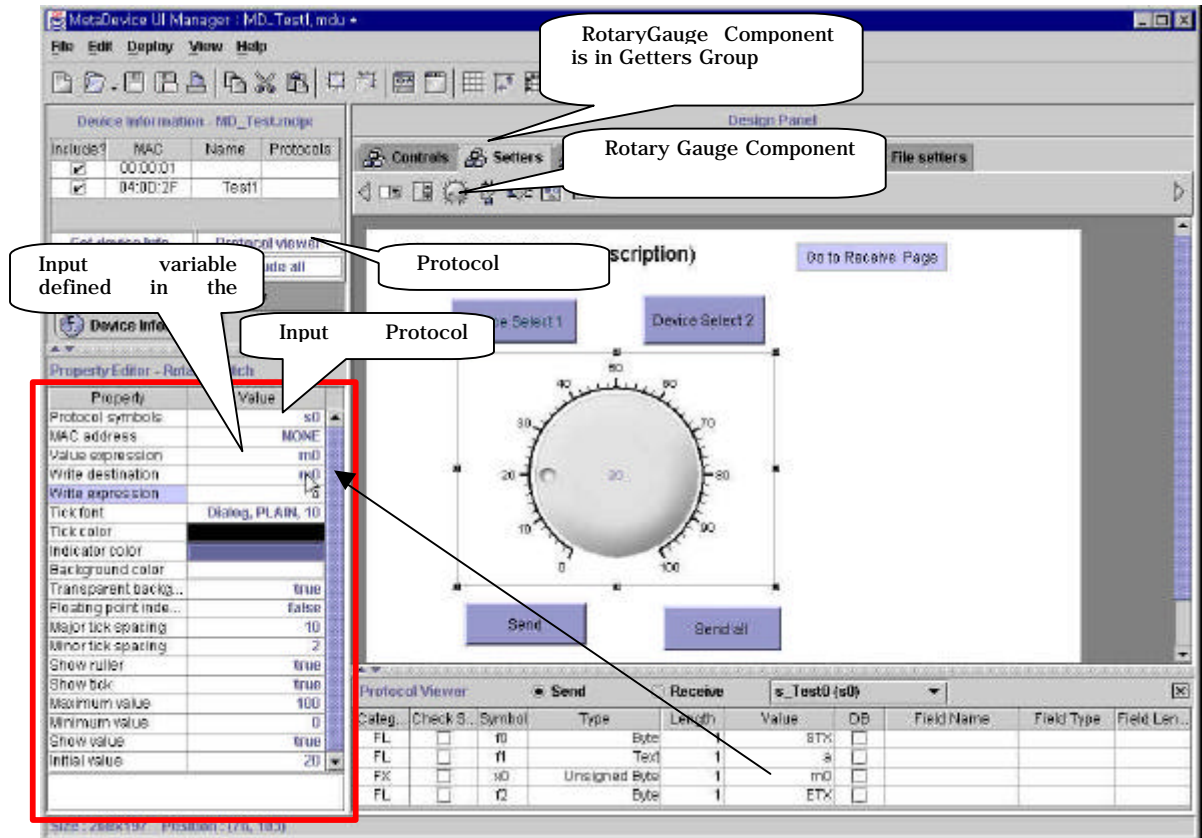
.Background color : Set Color of Button

.On State Image: The Image displayed with Button On. If Image is not set, displayed in color.

.Off State Image: The Image displayed with Button Off. If Image is not set, displayed in color.

.To Send or Receive : When the Button is used for Send Mode , Select “Send”. When the Button is used for Receive Mode, Select “ Receive”

- RotaryGauge Component for Sending Data



[Fig 3.4.1 RotaryGauge Component]

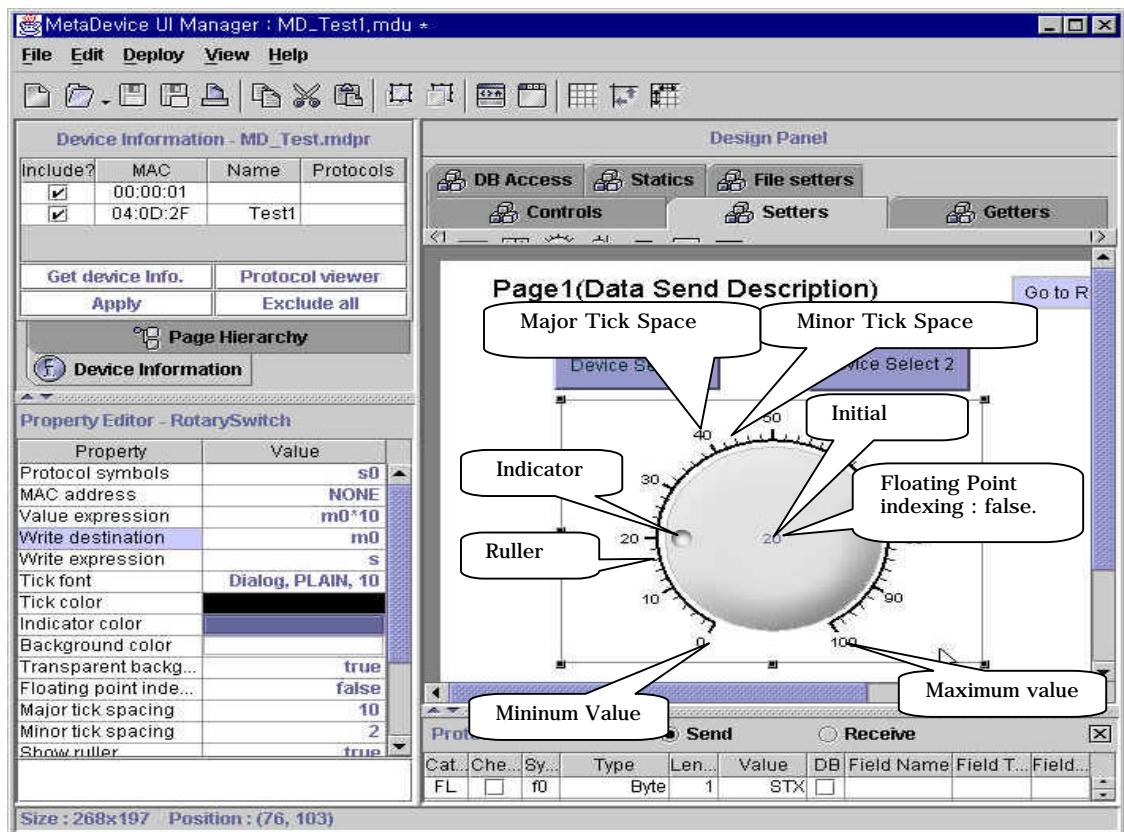
.Protocol symbols : Input **Protocol symbol not Protocol Name**. When Protocol Symbol is not clear, use Protocol Viewer.

.MAC address : Input Mac Number. When Mac Number is set to Component, the value is sent to the defined Mac Number Device. But if Mac Number is not set (**“NONE” state**), **The Device is dominated by “DeviceSelector”**. In other words, the Device variable defined in the Device Selector Button is sent.

.Value expression: Displays the value of defined Protocol in Component. Supports numerical operations.

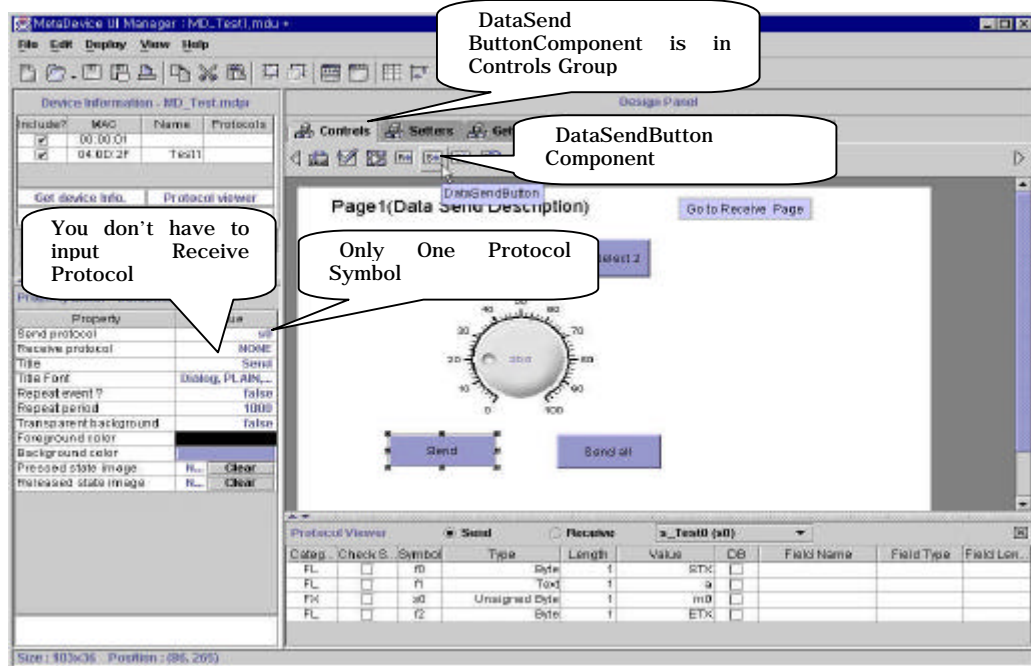
ex) if Value expression is defined as $m0*2$, Write expression is defined as $m0$, the value of Gauge shown is 20, and the actual value of Protocol is 10.

.Write expression: Assign the value of Protocol.



[Fig 3.4.2 Rotary Gauge Component Detail]

- DataSendButton Component for Sending data



[Fig 3.4.3 DataSendButton Component]

.Send Protocol : Input one Protocol Symbol.

.Receive Protocol : Normally keep as “NONE” state. However when attempting to receive data from the Server as soon as possible, input Receive Protocol Symbol.
The reason for not inputting Protocol Symbol is because the user can get data from the monitoring window.

.Title : Text to be displayed in Button.

.Title Font : Set Title Font

.Repeat event? :

 true :Send Setters Component value selected by Device Selector Button to the Server repeatedly.

 false: Conduct command only once when clicking Send.

.Repeat period : Input Repeat event Period when the Repeat event is true.

 Every 1 second : 1000

.Transparent background :

 true : Button is transparent. Color is not shown

 false: Button is not transparent. Color is shown.

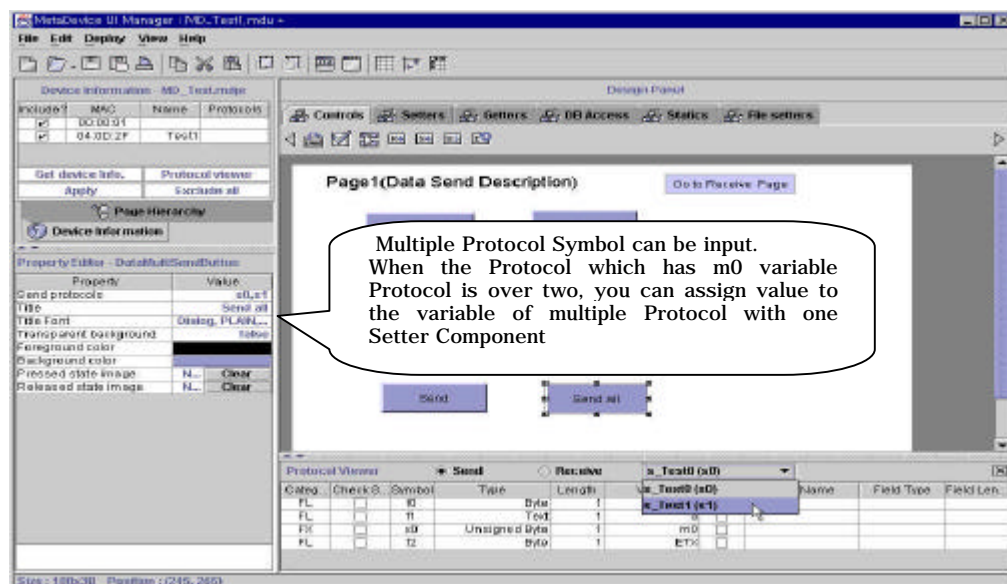
.Foreground color : Title Text Color

.Background color : Button Color

.Pressed State Image: Image displayed when pressing Button.

.Released State Image: Image displayed when releasing Button.

- DataMultiSendButton Component for Sending Data



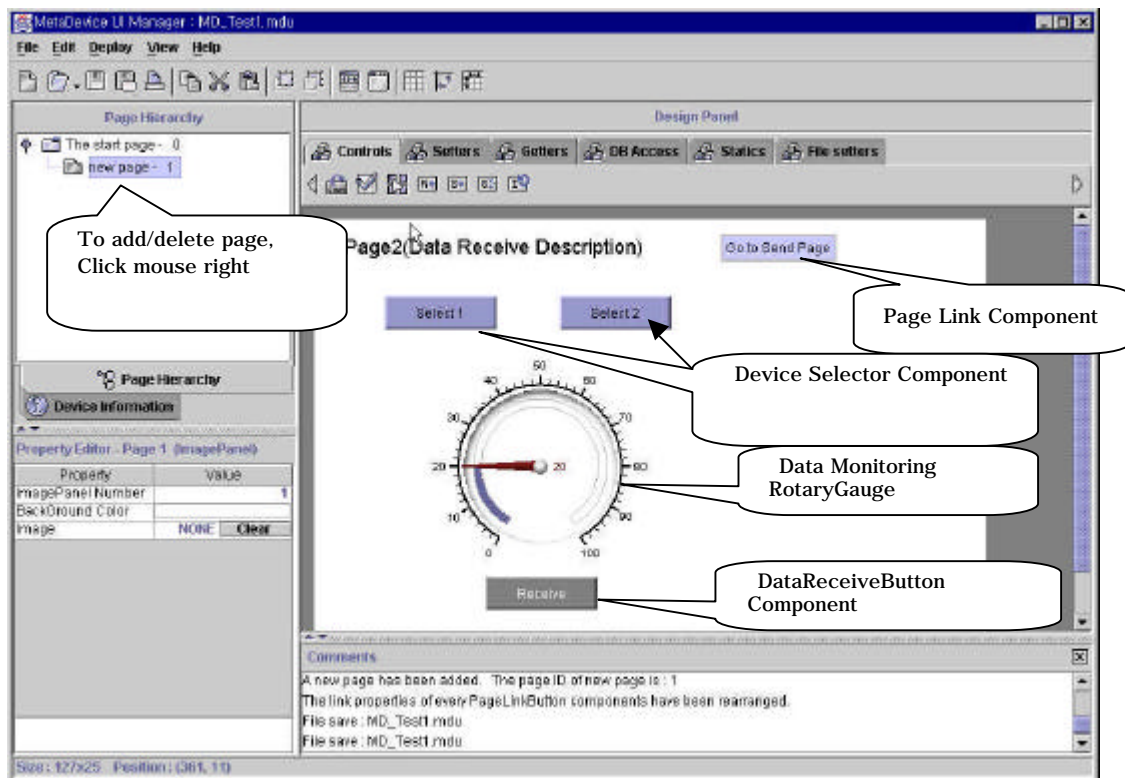
[Fig 3.4.4 DataMultiSendButton Component]

.Send Protocol : You can input over 2 Protocol Symbols.
.Title : Button Text
.Title Font : Title Font
.Transparent background :
 true : Button is transparent, Color is not shown
 false: Button is not transparent. Color is shown.
.Foreground color : Title Text Color.
.Background color : Button Color.
.Pressed State Image: Image displayed when pressing Button.
.Released State Image: Image displayed when releasing Button.

(2) Data Receive

(2.1) General Description

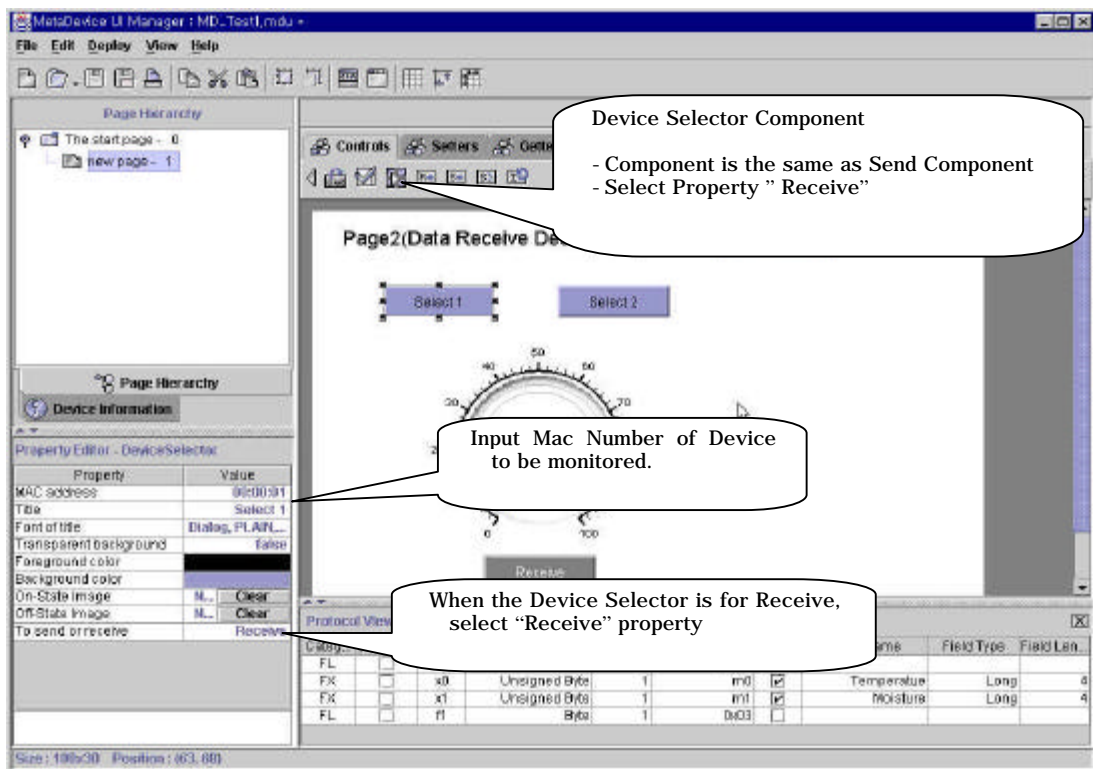
The Server always stores data to the memory and database while communicating with the Device. The user can see the Server memory and database data using the GUI Client program. To see the data, the GUI program should basically consist of "ReceiveButton" Component ,"Device Selection" Component and "Getters" Group Component. In other words, while the Run-Time program is running, the user should select a Device and monitor the data received from it through the Getters Component. [**Fig 3.5**] shows a basic Component for Device Monitoring. When receiving data from the Server, the user can select only one Device because one getter component cannot display multiple device data at the same time.



[Fig 3.5 Example of Component for Data Monitoring]

(2.2) Component Property Description

- DeviceSelector Component for Data Monitoring



[Fig 3.5.1 DeviceSelector Component]

.MAC address : Input Mac Number registered in the Server as well as related with User in the Admin module.

.Title : Text to be displayed in Button

.Font of title : Set Title Font

.Transparent background :

true : Color is not shown because Button is transparent

false: Color is shown because Button is not transparent.

.Foreground color : Set Title Text Color

.Background color : Set Color of Button

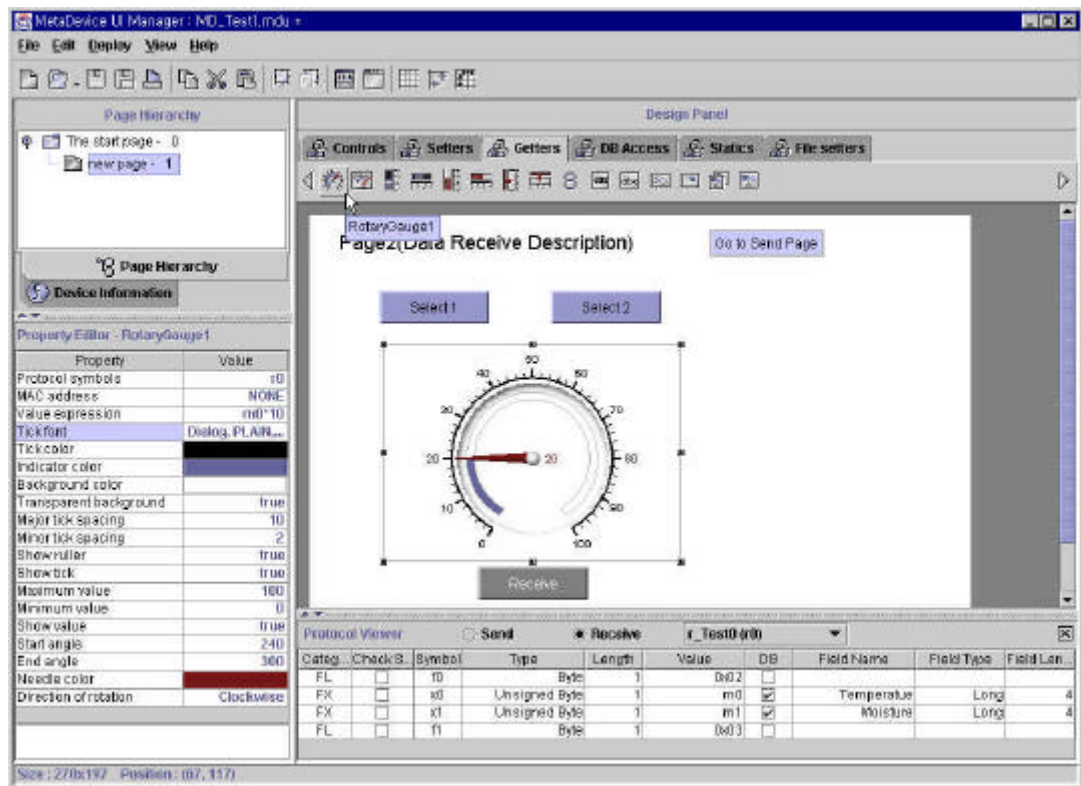
.On State Image: Image displayed with Button On. If Image is not set, displayed in Color.

.Off State Image: Image displayed with Button Off. If Image is not set, displayed in Color.

.To Send or Receive : When the Button is used for Send Mode , Select "Send". When the Button is used for Receive Mode, Select "Receive".

Note : Receive Device Selector Components are exclusive from each other during Run Time. In other words, only one Selector Component can be selected at a given time

- RotaryGauge1 Button Component of Getters Group for Data Monitoring

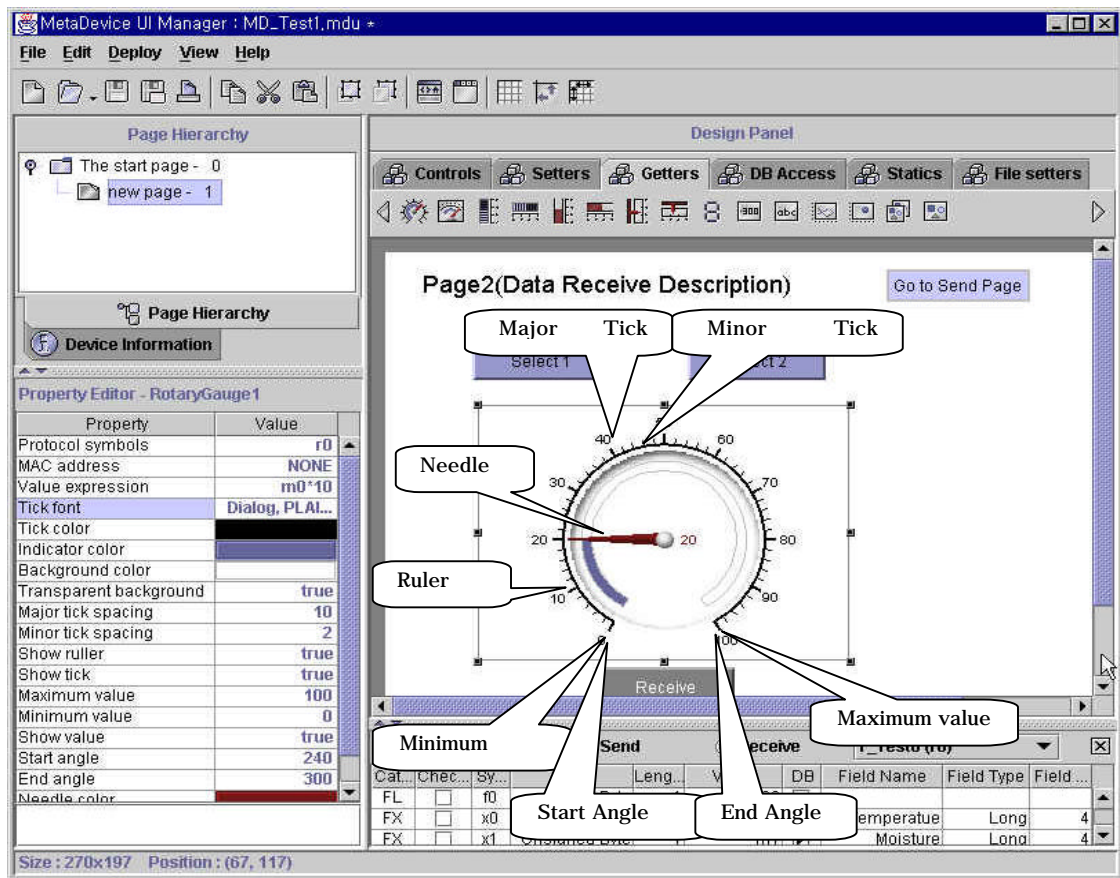


[Fig 3.5.2 RotaryGauge Component]

.Protocol symbols : Input **Protocol symbol not Protocol Name**. When Protocol Symbol is not clear, use Protocol Viewer. Only Receive Protocols can be input.

.MAC address : Input actual Mac Number. When Mac Number is set to Component, the value is shown from the defined Mac Number Device. But if Mac Number is not set (**“NONE” state**), **The Device is dominated by “DeviceSelector”**. In other words, the Device variable is shown in the Device defined by Device Selector Button.

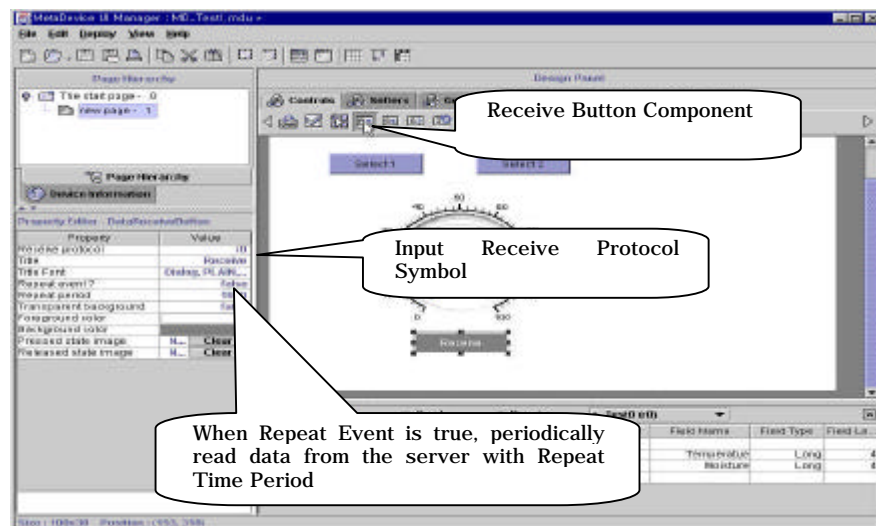
.Value expression: Displays the value of Protocol in Component. Supports numerical operations.



[Fig 3.5.3 RotaryGauge Component Detail]

- ReceiveButton Component for Data Monitoring

To display value in Component, select one Device and click Receive.



[Fig 3.5.4 Receive Button Component]

.Receive Protocol : input one Receive Protocol Symbol.

.Title : Text to be displayed in Button.

.Title Font : Set Title Font

.Repeat event? :

 true : Repeatedly monitors Device selected by Device Selector Button.

 false: Conducts once when clicking Receive.

.Repeat period : Input Repeat event Period when the Repeat event is true.

 Every 1 second : 1000

.Transparent background :

 true : Button is transparent. Color is not shown

 false: Button is not transparent. Color is shown.

.Foreground color : Title Text Color

.Background color : Button Color

.Pressed State Image: Image displayed when pressing Button.

.Released State Image: Image displayed when releasing Button.

(3) Value Expression Detail

The GUI Program (Applet or Application deployed by UIManager) and MetaDevice Sever Program communicate with each other. The Receive Protocol variable defined in the Component “Value expression” Property item displays data received from the Server.

“Value expression” Property variable types are two types: m<address> type and v<address>.

m<address>

- Variable type is Integer, or Real(Double).
- Value expression Property item : supports arithmetic operations and logical operations.

$((m100 - 10) / 5 + (m100 + 10) / 10)$

- Applicable Component

RotaryGauge1, RotaryGauge2, VerticalLinearGauge1,2,3, HorizontalLinearGauge1,2,3

SevenSegmentDisplay, LabelValueGetter, Graph2D, DataMappingPanel

ImageDisplayValueGetter, ImageON/OFFGauge

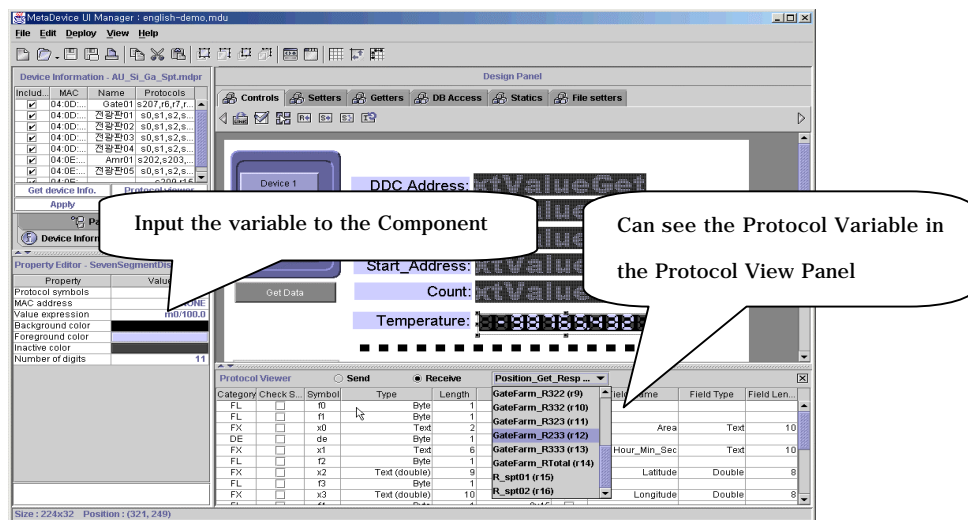
v<address>

- Variable type is Text. Fixed length Text or variable length Text
- Value expression supports Text add(+) operations

ex) v0 + “ABCD”

- Applicable Component

TextLabelValueGetter



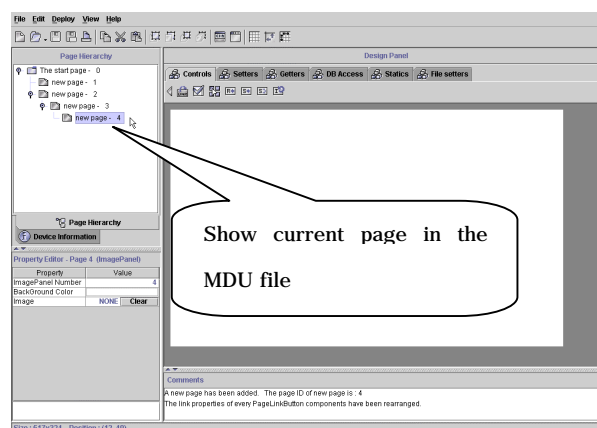
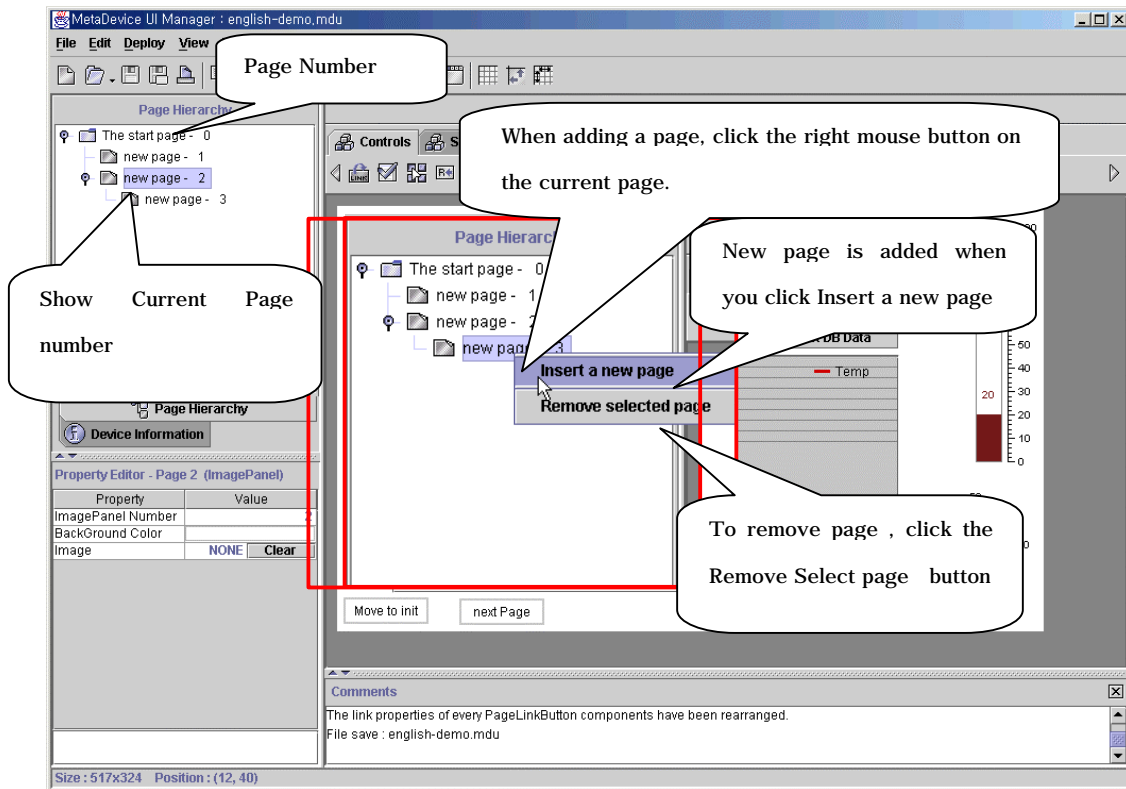
[Fig 3.6.1 Set Value expression]

(4) Page Add/Delete/Move

For cases when multiple pages are needed for application, the UIManager supports multiple page handling functions. The user can add or delete pages.

“PageLinkButton” in the Controls Group Component supports moving windows during Run-Time.

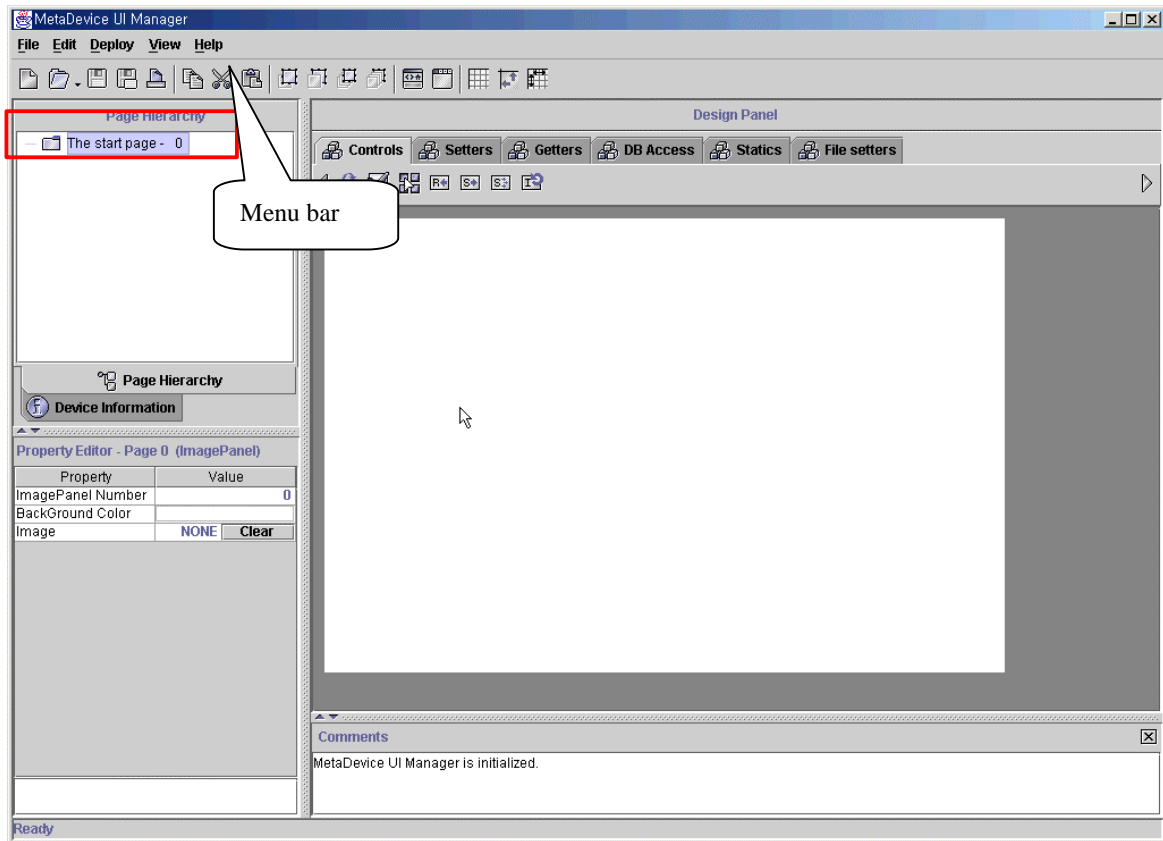
“Page to move” Property is set with page numbers.



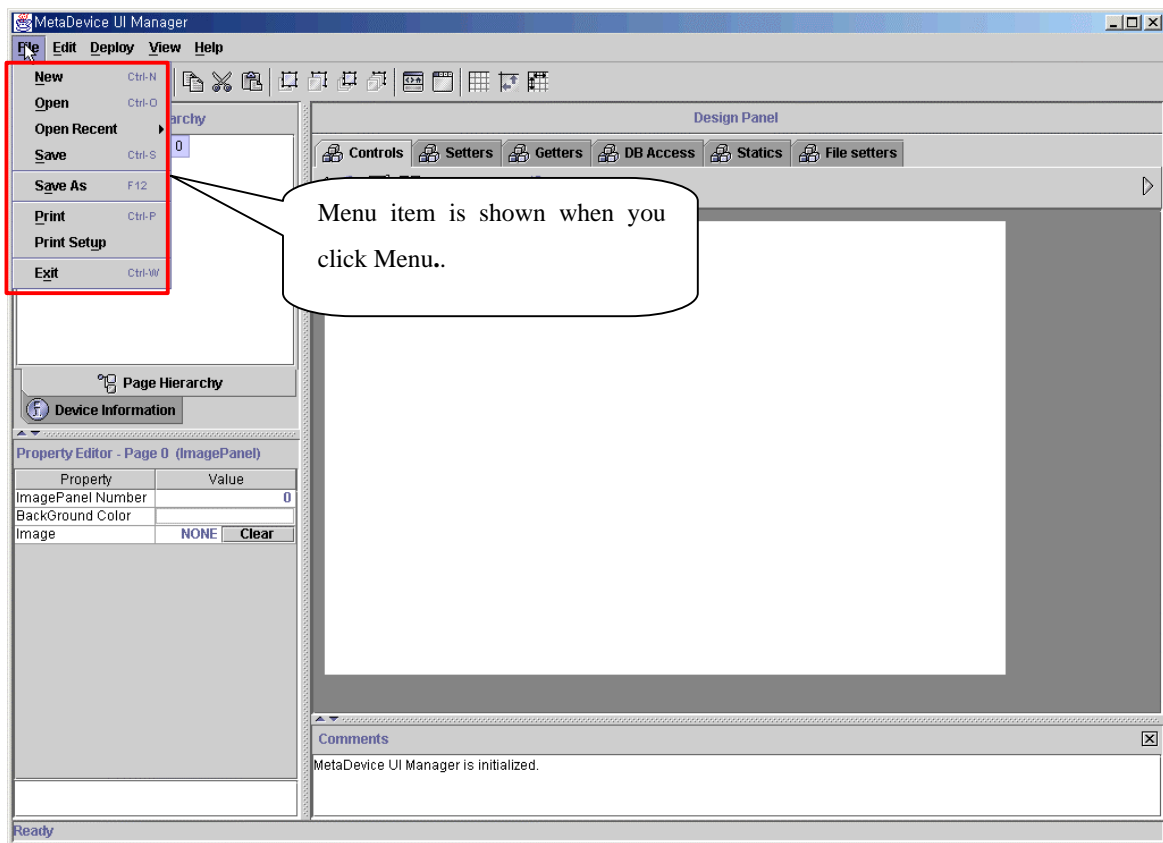
[Fig 3.6.2 Page Add/Delete]

2. Menu Bar

The **Menu Bar** [Fig 3.7.1] is shown when the UI Manager is running. Each menu has its own menu item which has functions such as file saving or file reading. The **Menu Bar** function has **Tool Bar** functions and has menu selections such as [Fig 3.7.2] .



[Fig 3.7.1 Menu Bar]



[Fig 3.7.2 File Menu item]

(1) File Menu

Open/Save/Create MDU file.

(1.1) New

Create New MDU file.

(1.2) Open

Open existing MDU file.

(1.3) Open Recent

List and Open recent MDU file.

(1.4) Save

Save the working MDU file.

(1.5) Save As

Save the working MDU file in another file name.

(1.6) Print

Print the Design Panel.

(1.7) PrintSetup

Set the Printer.

(1.8) Exit

Exit UI Manager.

(2) Edit Menu

Edit Component.

(2.1) Cut

Cut and Save selected Component to clipboard.

(2.2) Copy

Copy & Save selected Component to clipboard.

(2.3) Paste

Paste saved Component in clipboard to DesignPanel.

(2.4) Select All

Select all Components in the DesignPanel.

(2.5) Delete

Delete Component in the DesignPanel.

(3) Deploy

Deploy MDU file as Applet / Application.

(3.1) Make Applet

Deploy MDU file as Applet.

(3.2) Make Application

Deploy MDU file as Application..

(4) View

Set Grid.

(4.1) Grid ON/OFF

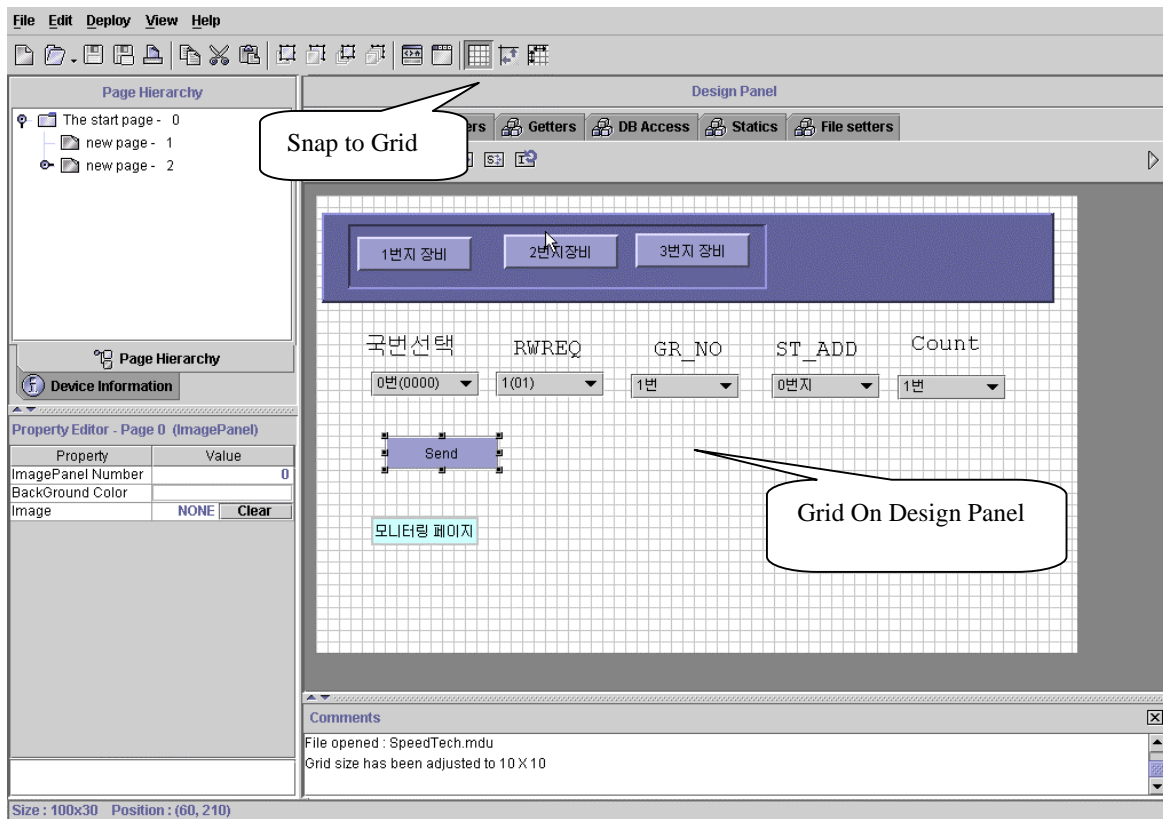
Turn Grid On/Off in Design Panel.

(4.2) Grid property

Set size of Grid.

(4.3) Snap to Grid

Locate Component according to Grid line.



[Fig 3.7.3 Grid ON/OFF]

(5) Help

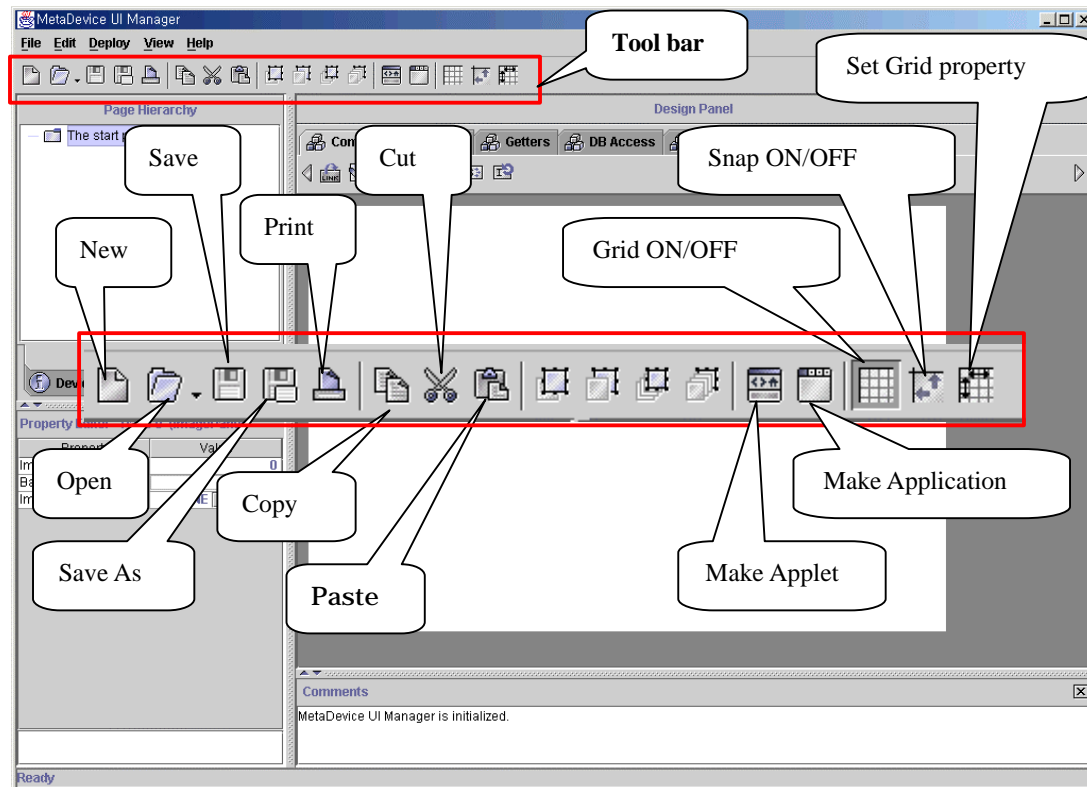
(5.1) Help topics

(5.2) About





Shows MetaDevice current version.

3. Tool Bar

The **Tool Bar** is shown as [Fig 3.8] in the UI Manager. The **Tool Bar** has functions controlling components and files.



[Fig 3.8 Tool bar]

-  Increase Z-order Selected components by 1: Used when displaying selected component on top of another component.
-  Decrease Z-order Selected components by 1: Used when displaying selected component under another component.
-  Make the Z-order Selected components top most: Used when displaying selected component at the very top of all other components.
-  Make the Z-order Selected components bottom most: Used when displaying selected component at the very bottom of all other components.

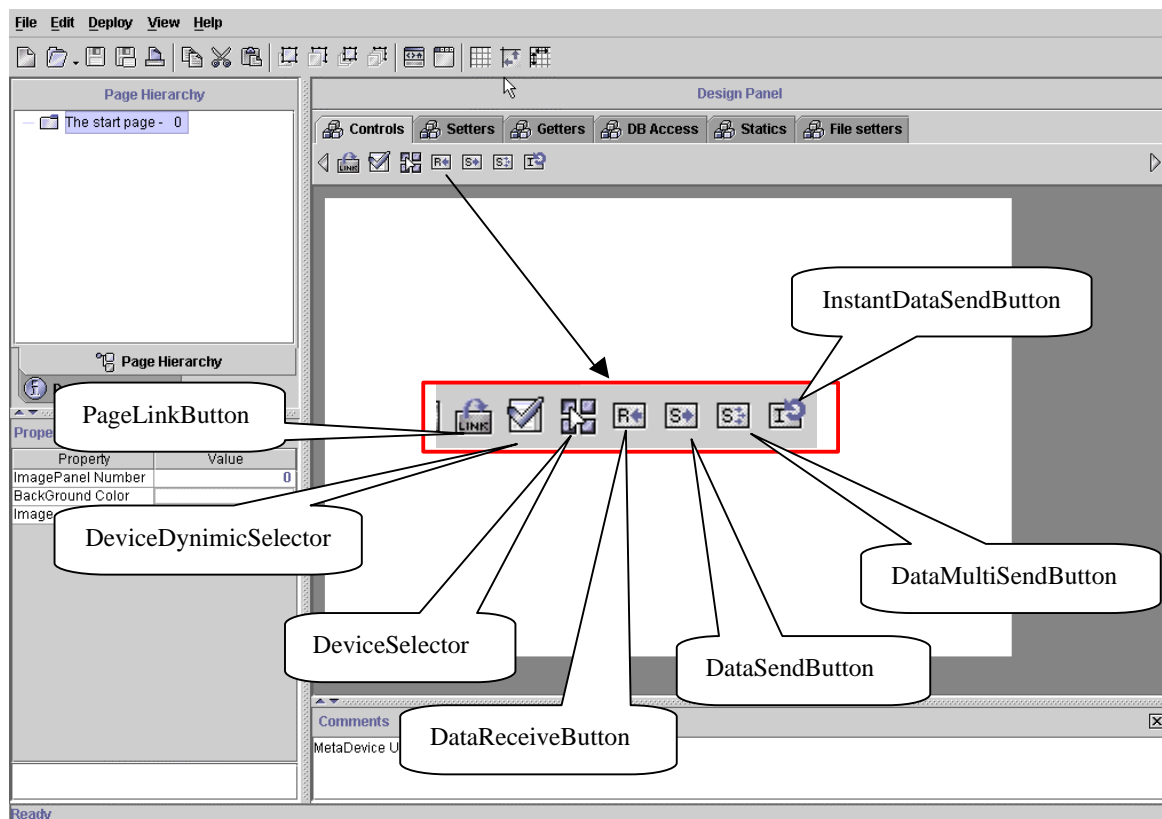
4.Controls Component Group

(1) What are **Controls Components**?

Controls Component supports page moving, Device selecting, sending send protocol data and receiving receive protocol data when the applet/application program is running

(2) **Controls Components Item**

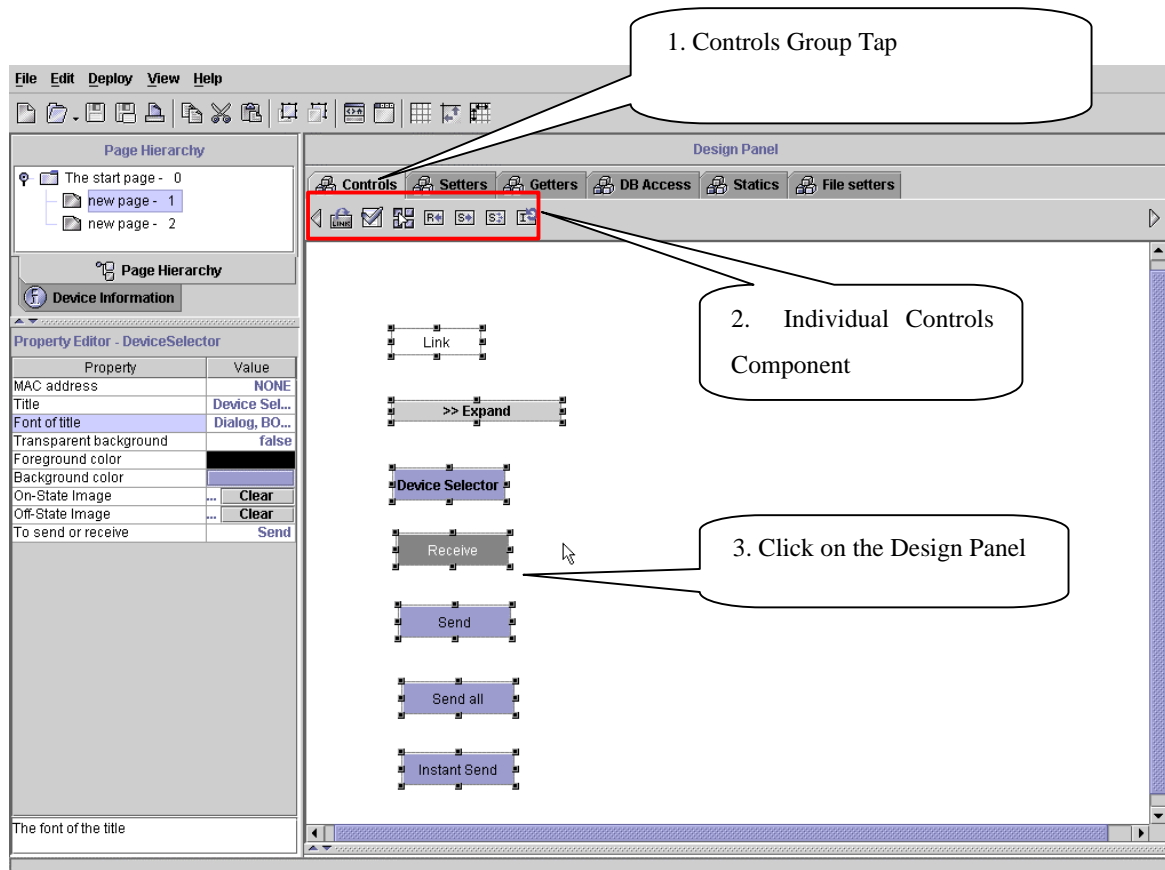
- PageLinkbutton : Button moving page at Run-Time
- DeviceDynamicSelector : Device Selector List is opened explicitly by User ID at Run Time
- DeviceSelector : Device Selector is defined implicitly at Design Time .
- DataReceiveButton : Reads protocol data from the Server.
- DataSendButton : Sends only one protocol data to the Server.
- DataMultiSendButton : Sends multi protocol data to the Server.
- InstantDataSendButton : Sends one protocol data to the Server instantly.



[Fig 3.9.1 Controls Group Tab]

(3) Design Controls Component on Design Panel

- Click Controls Group Tab.
- Click the location where the Control Component should be located on the Design Panel after selecting individual component
- Double click the Component to see Property.
- Edit Property of the Control Component.

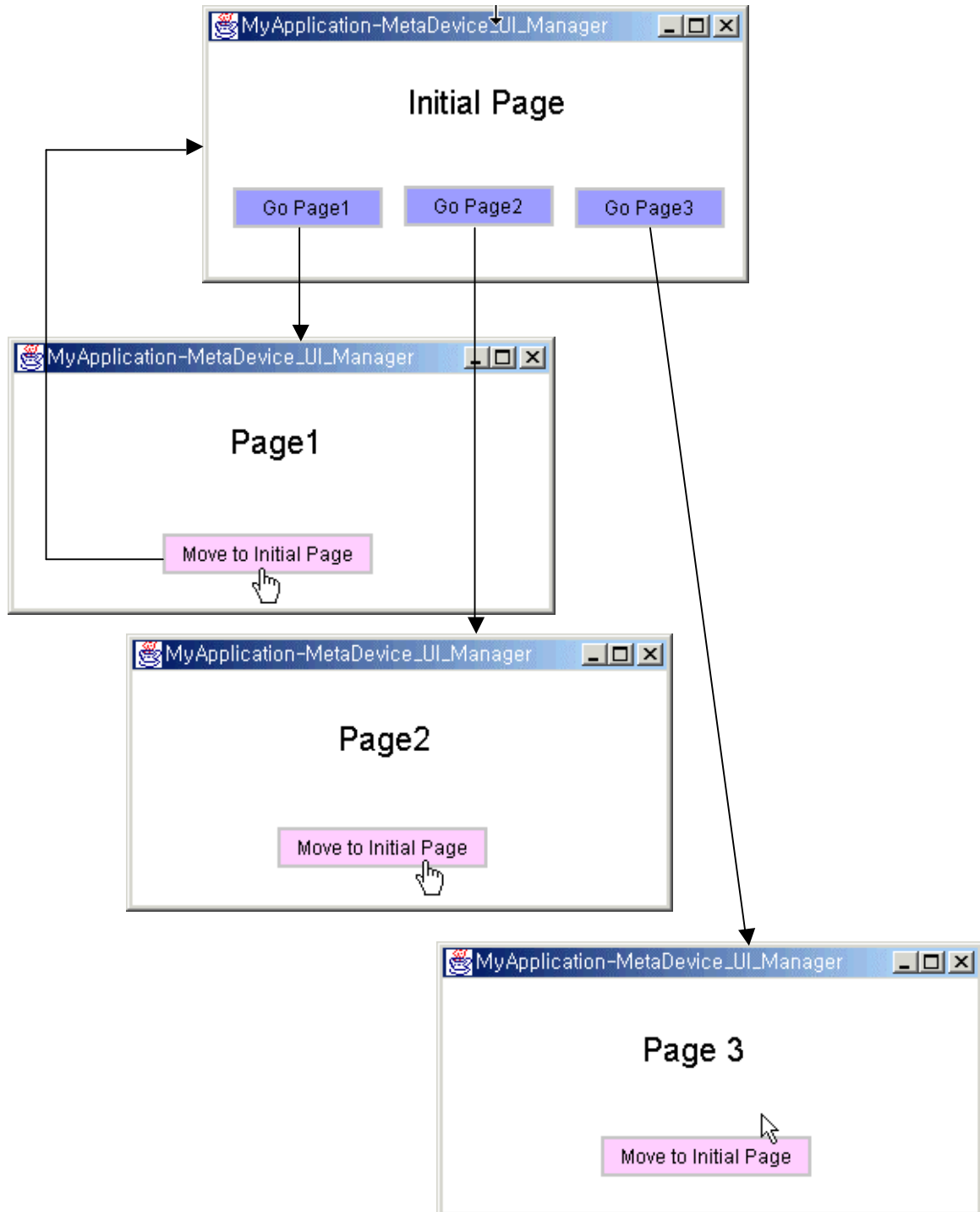


[Fig 3.9.2 Design Controls Component]

4.1 PageLinkButton

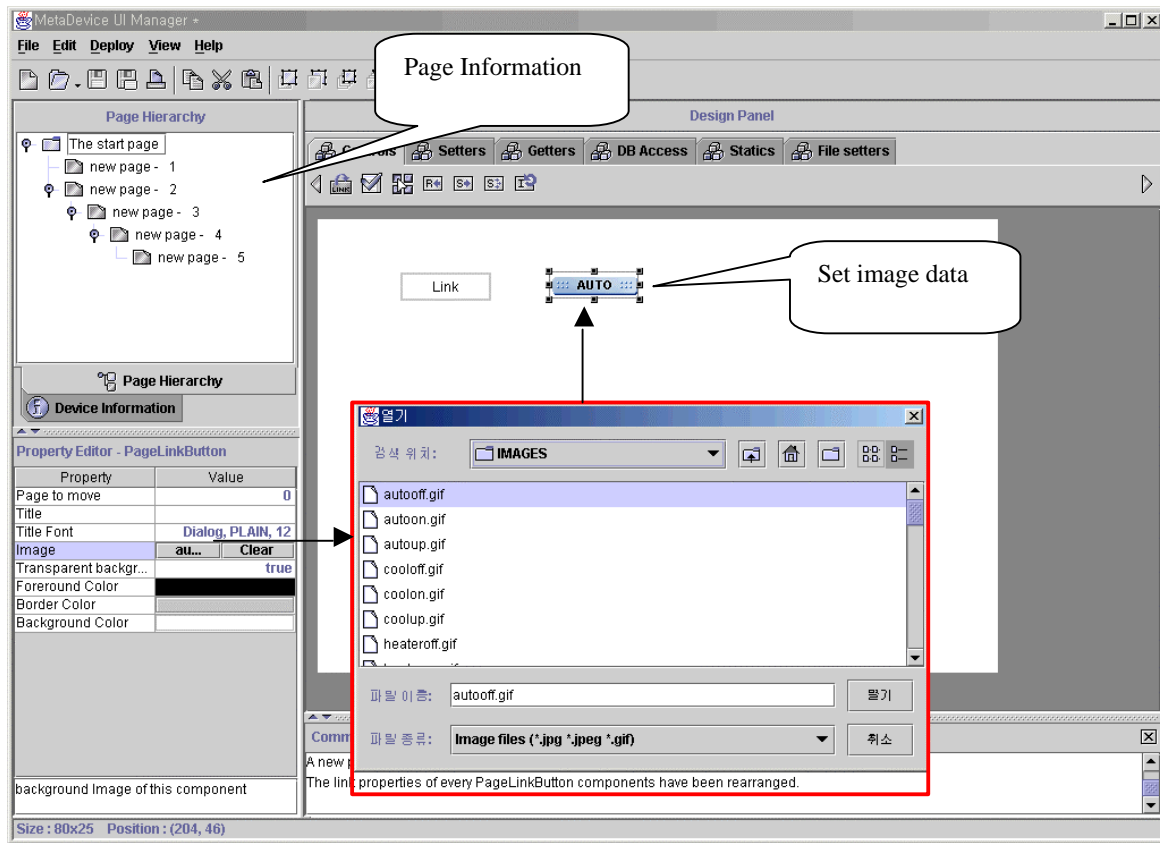
(1) Working at Run Time

User can move pages at Run Time

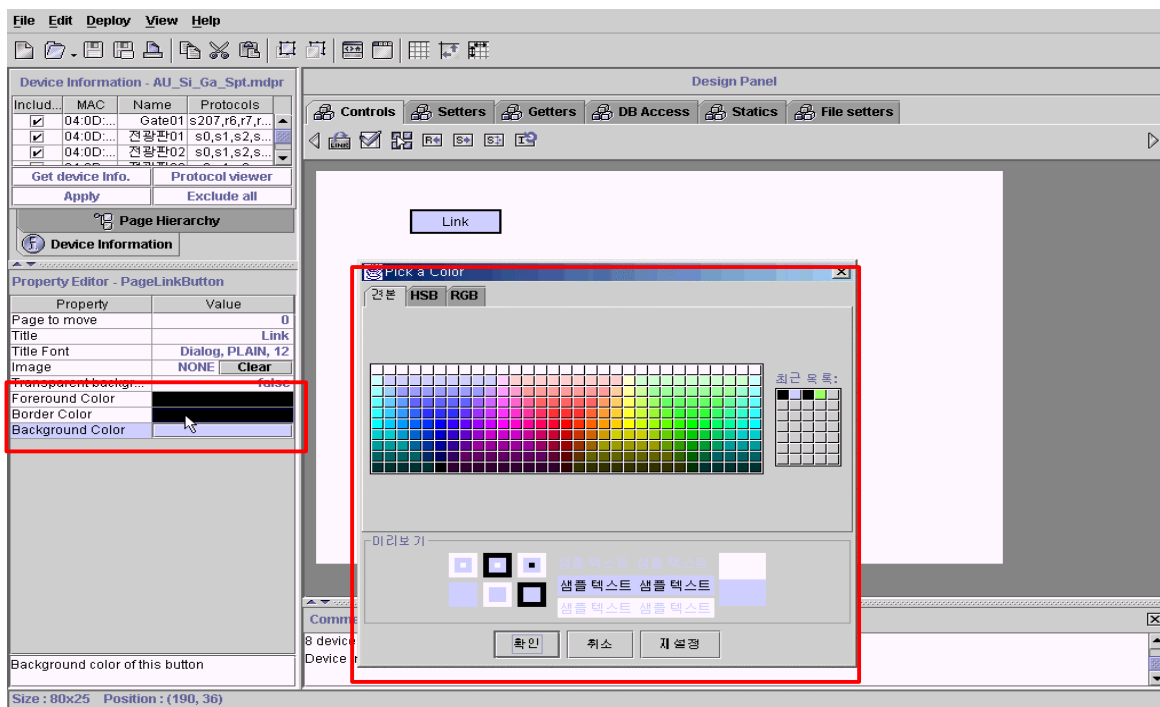


[Fig 3.9.3 PageLinkButton Example]

(2) Setting Property at Design Time



[Fig 3.9.4 Setting PageLinkButton Image Property]



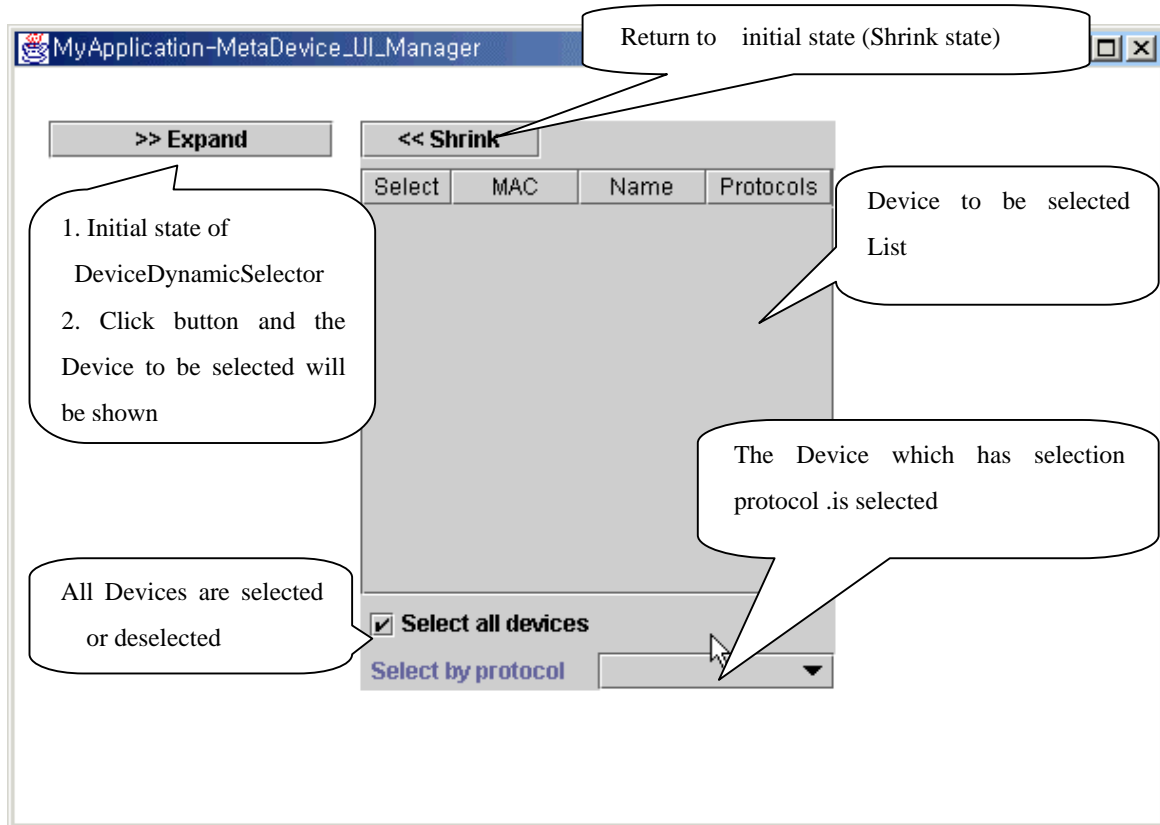
[Fig 3.9.5 Setting PageLinkButton Color Property]

- Page to Move : Input page number to be moved
- Title : Button Text
- Title Font : Title Font
- Image : You can set image file(.gif, .jpg, .jpeg) or not
- Transparent background : If set as True , the text is shown but the background is not shown
- Foreground Color : Text Color
- Border Color : Button Border Color
- Background Color : Button Background Color

4.2 DeviceDynamicSelector

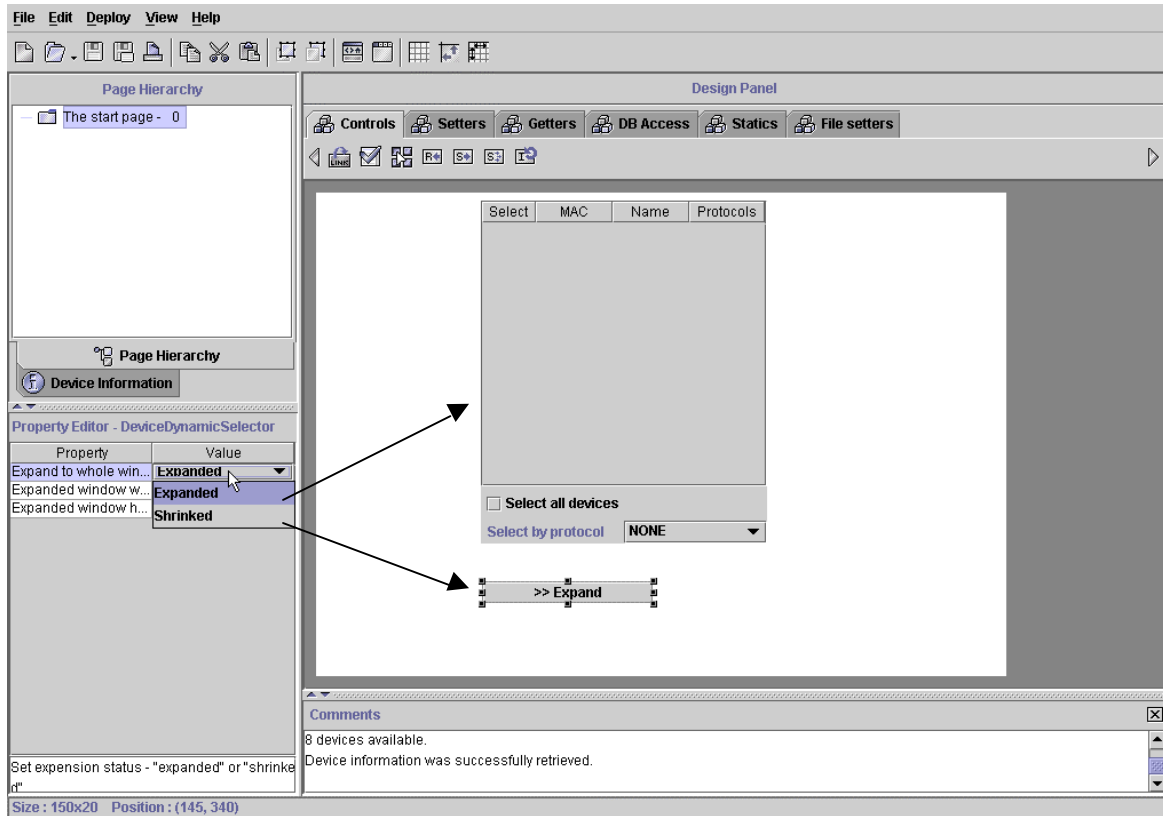
(1) Working at Run Time

The Device Selector List is opened by the User ID at Run Time and the user can select all devices at once



[Fig 3.10.1 DeviceDynamicSelector Example]

(2) Setting Property at Design Time



[Fig 3.10.2 Setting DeviceDynamicSelector Property]

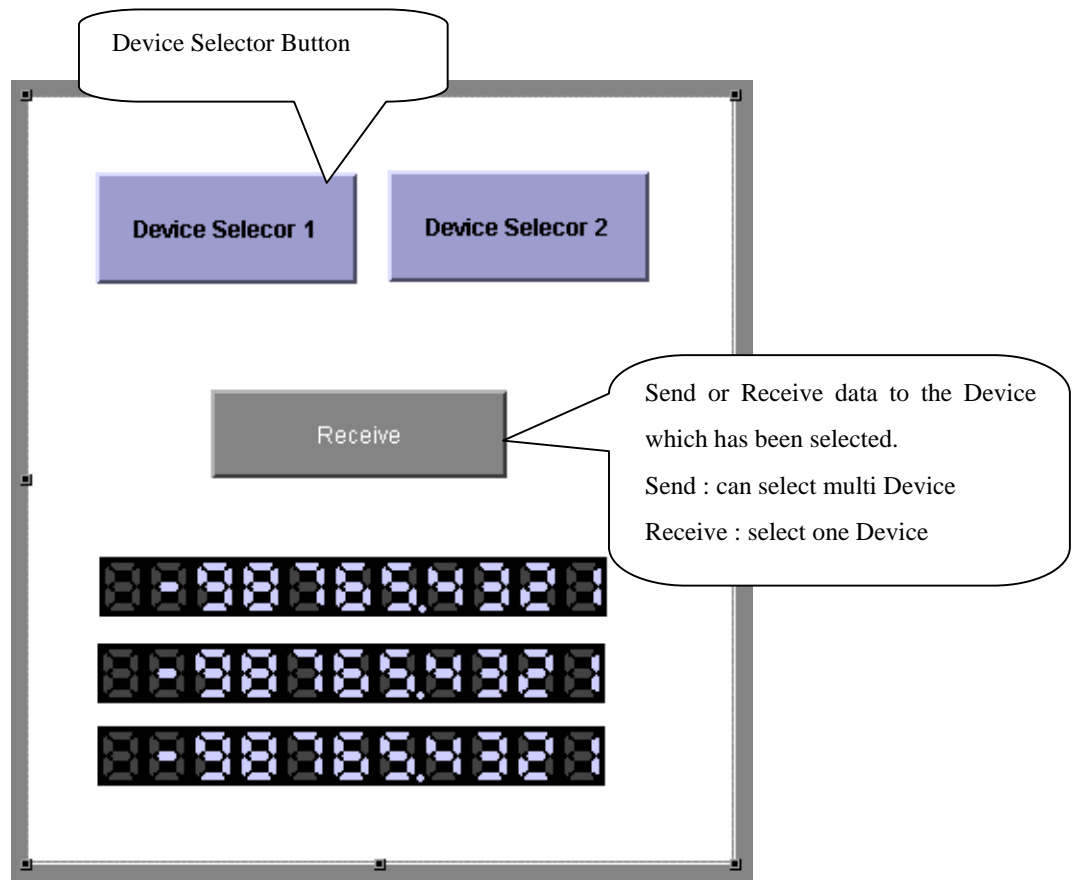
- Expand to Whole window : Displays state at Design Mode
- Get Device info. at login : Displays Device List that user can select at Run Time
- Expand to Window width : The width of Component
- Expand to Window height : The height of Component

4.3 DeviceSelector

(1) Working at Run Time

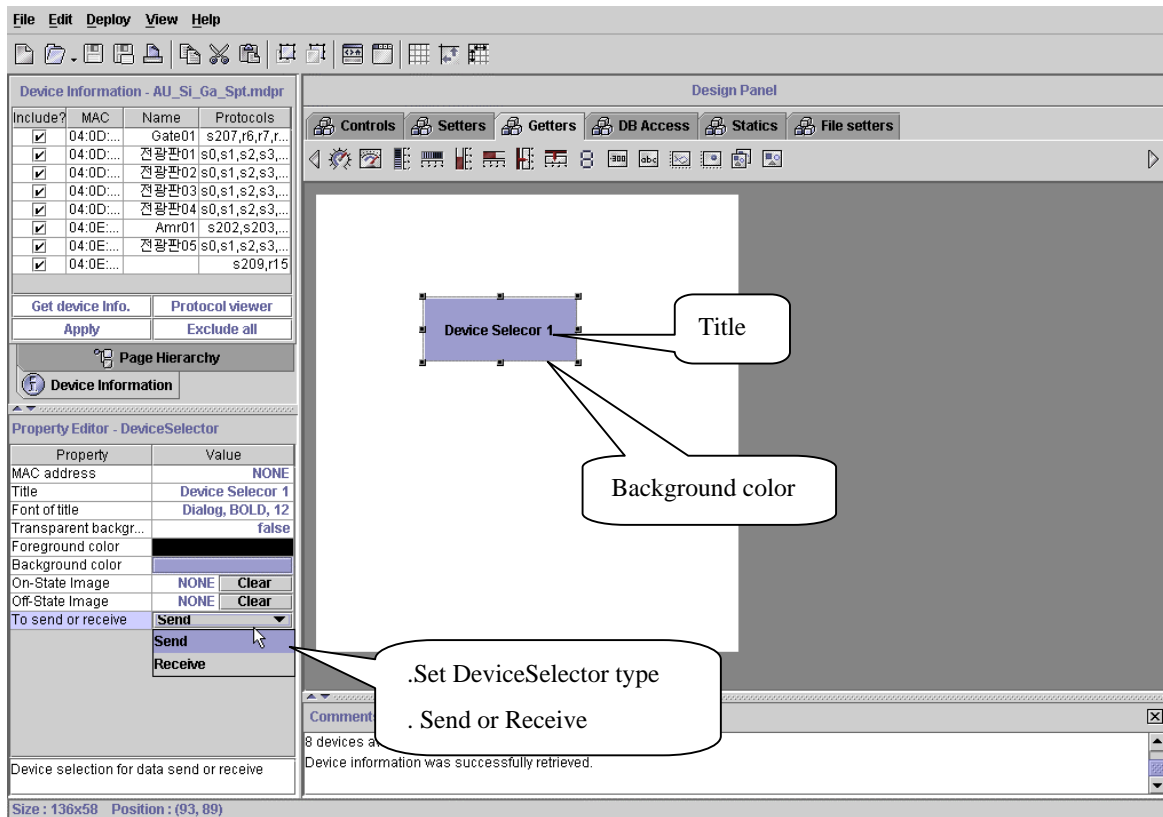
The User can select device to send or receive data.

Only the Device that is related with User ID can be selected.



[Fig 3.11.1 DeviceSelector Button Example]

(2) Setting Property at Design Time



[Fig 3.11.2 Setting DeviceSelector Property]

MAC address : The Mac Number registered in the Server and related with User in the Admin module. You do not have to input the Mac Number which is related to the User but if you input a Mac Number that does not have any relations with User, the Device can not be selected in the Run Time.

.Title : Text which is shown in the Button

.Font of title : Sets Title Font

.Transparent background :

true : The Color is not shown because the Button is transparent

false: The Color is shown because the Button is not transparent.

.Foreground color : Sets Title Text Color

.Background color : Sets Color of Button

.On State Image: The Image with the Button On. When the Image is not set, the Color is shown.

.Off State Image: The Image with the Button Off. When the Image is not set, the Color is shown.

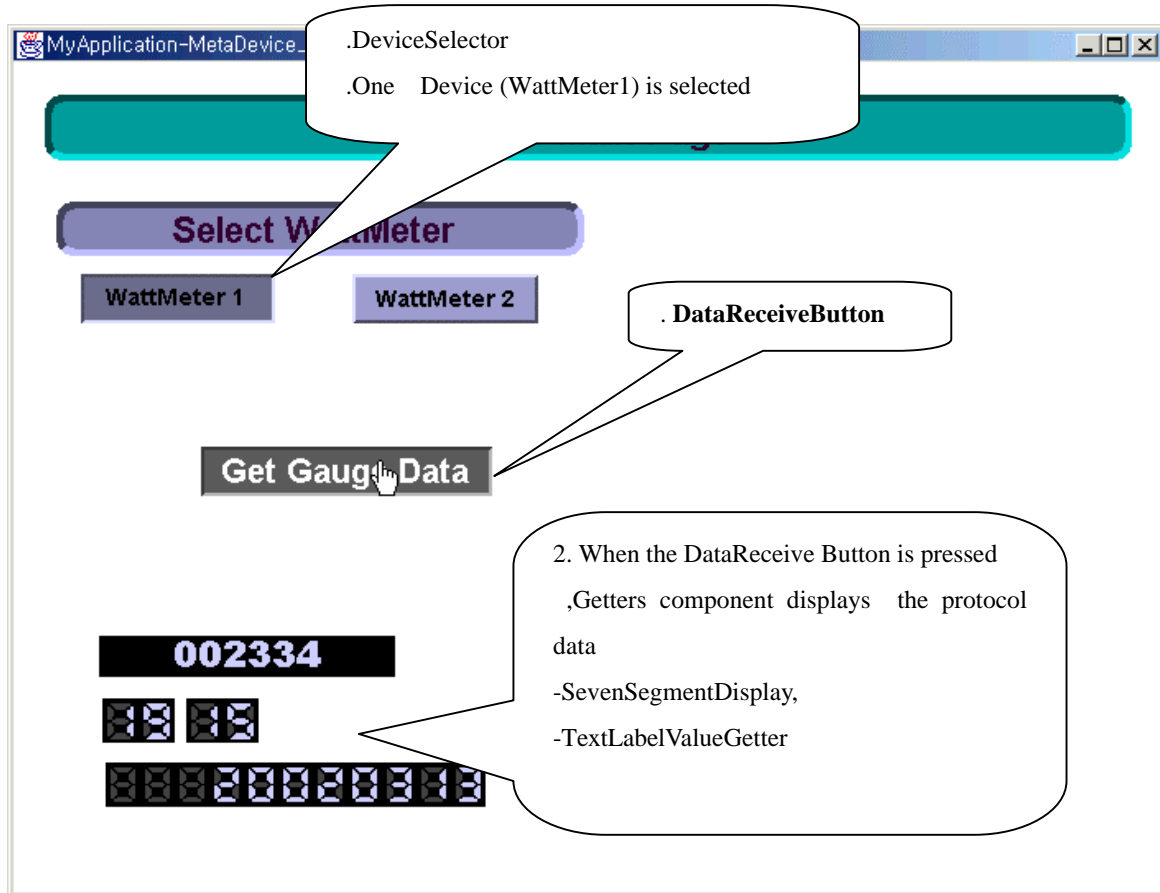
.To Send or Receive : When the Button is used for Send Mode , Select "Send", When the Button is used for Receive Mode, Select " Receive"

4.4 DataReceiveButton

(1) Working at Run Time

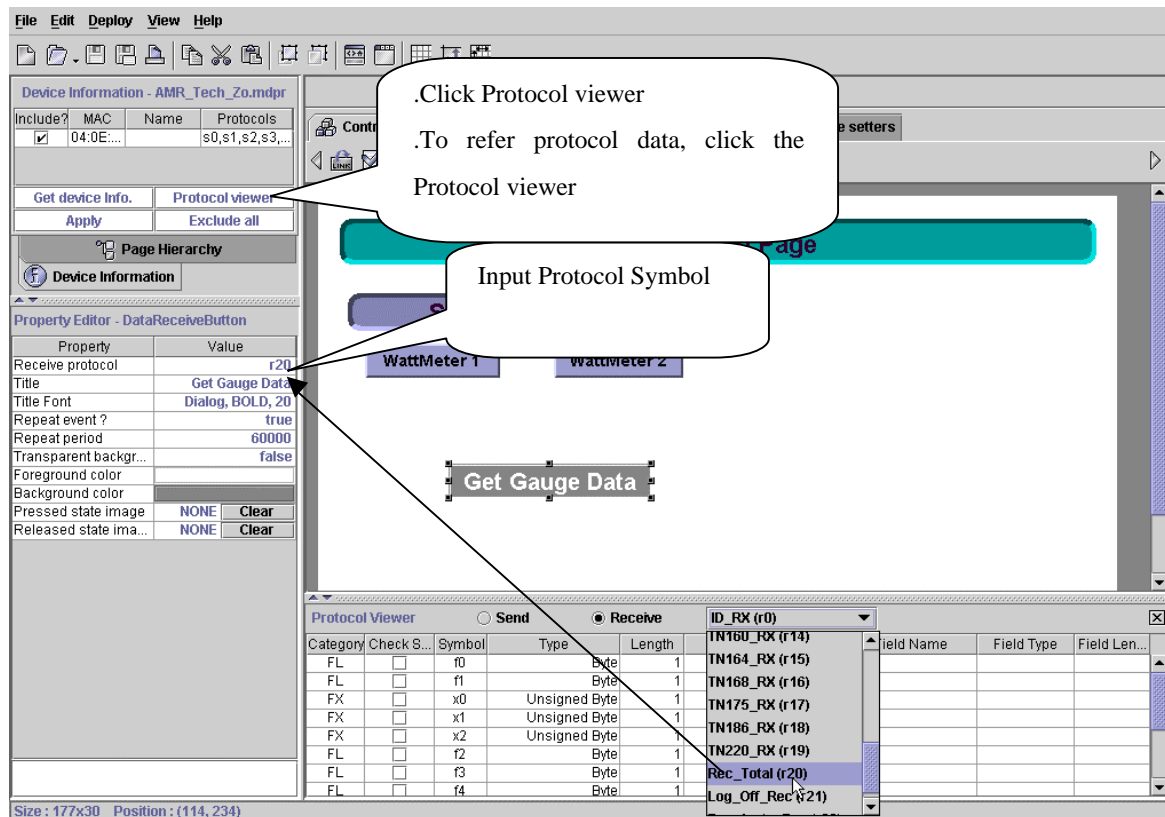
DataReceiveButton Component receives data from the Server

The method of receiving data from the Server depends on the Repeat Event Property.



[Fig 3.12.1 DataReceiveButton Example]

(2) Setting Property at Design Time



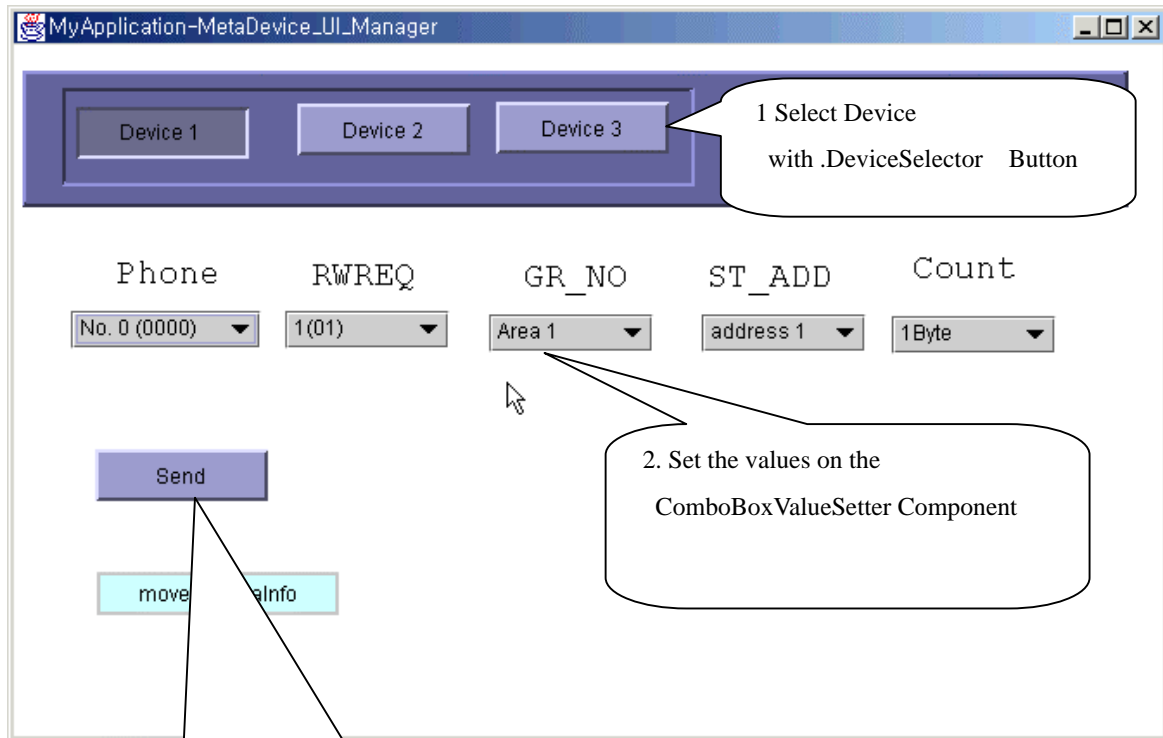
[Fig 3.12.2 Setting DataReceiveButton Property]

- Receive protocol : Input Receive Protocol Symbol
- Title : Button Text
- Repeat event : If set as True , receives data periodically (Repeat period : Value) from the Server
If set as false , receives data from the Server whenever the button is pressed
- Repeat period : When the Repeat event is True , sets the period interval time
(1000-> 1000 ms, 1 Sec)

4.5 DataSendButton

(1) Working at Run Time

DataSendButton sends one protocol variable values (m0,m1..or v0,v1..)-which is set on the Setters Component -to the Server

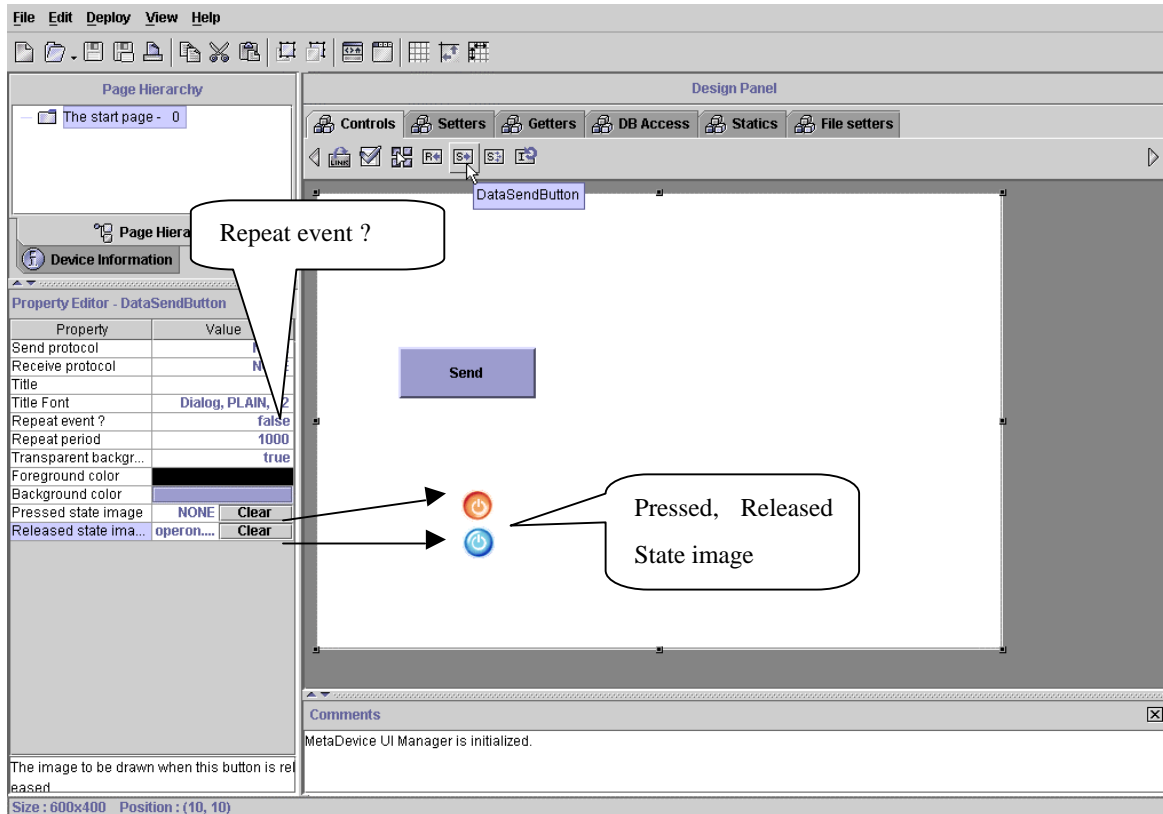


[Fig 3.13.1 DataSendButton Example]

3. DataSendButton

When DataSendButton is pressed, the protocol that is set on the DataSendButton is sent to the Server. The Server receives data from the client and sends data to the Device by following Semantics

(2) Setting Property at Design Time



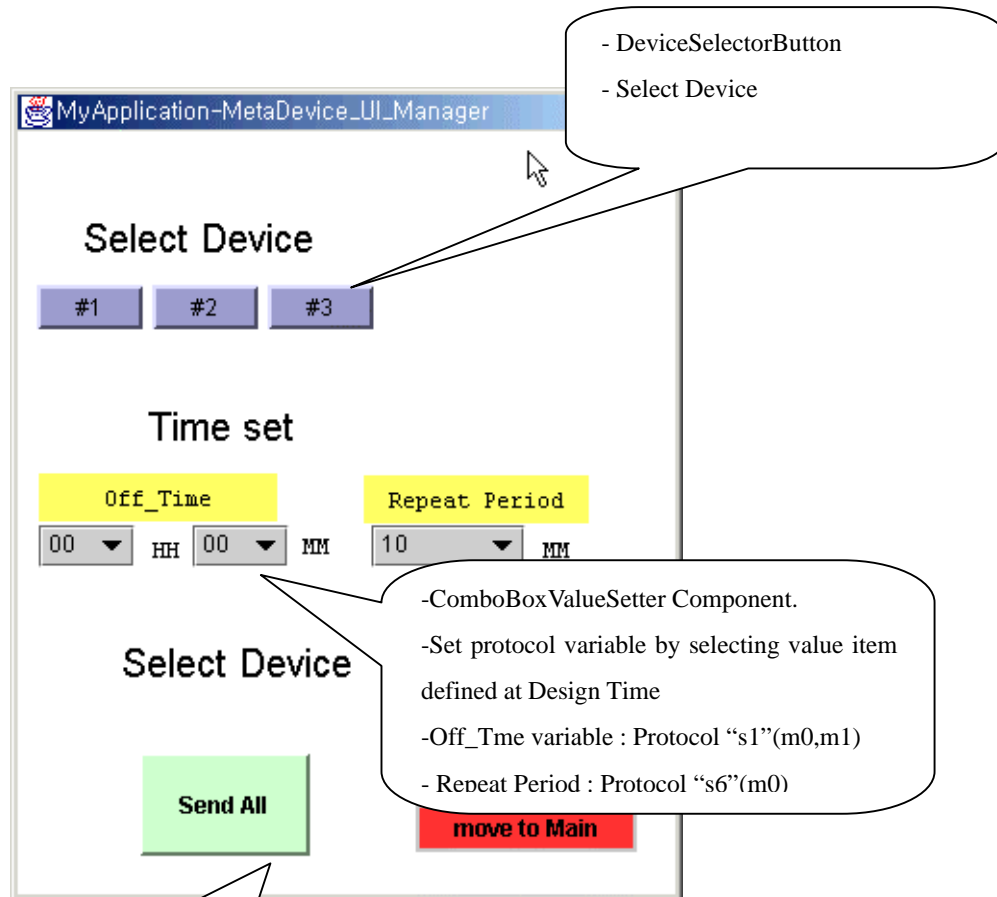
[Fig 3.13.2 Setting DataSendButton Property]

- Send protocol : Input Protocol Symbol (Only One)
- Receive protocol : Receive Protocol Symbol (Keep "None" mostly)
- Repeat event : If set as True , sends data periodically (Repeat period : Value) to the Server
If set as false , sends data to the Server whenever the button is pressed
- Repeat period : If the Repeat event is True , sets the period interval time
(1000-> 1000 ms, 1 Sec)
- Transparent background : If True , only the Text is shown.
- Foreground Color : Text Color
- Background Color : Button Color
- Pressed State Image : The image when the button is Pressed
- Released state Image : The image when the button is released

4.6 DataMultiSendButton

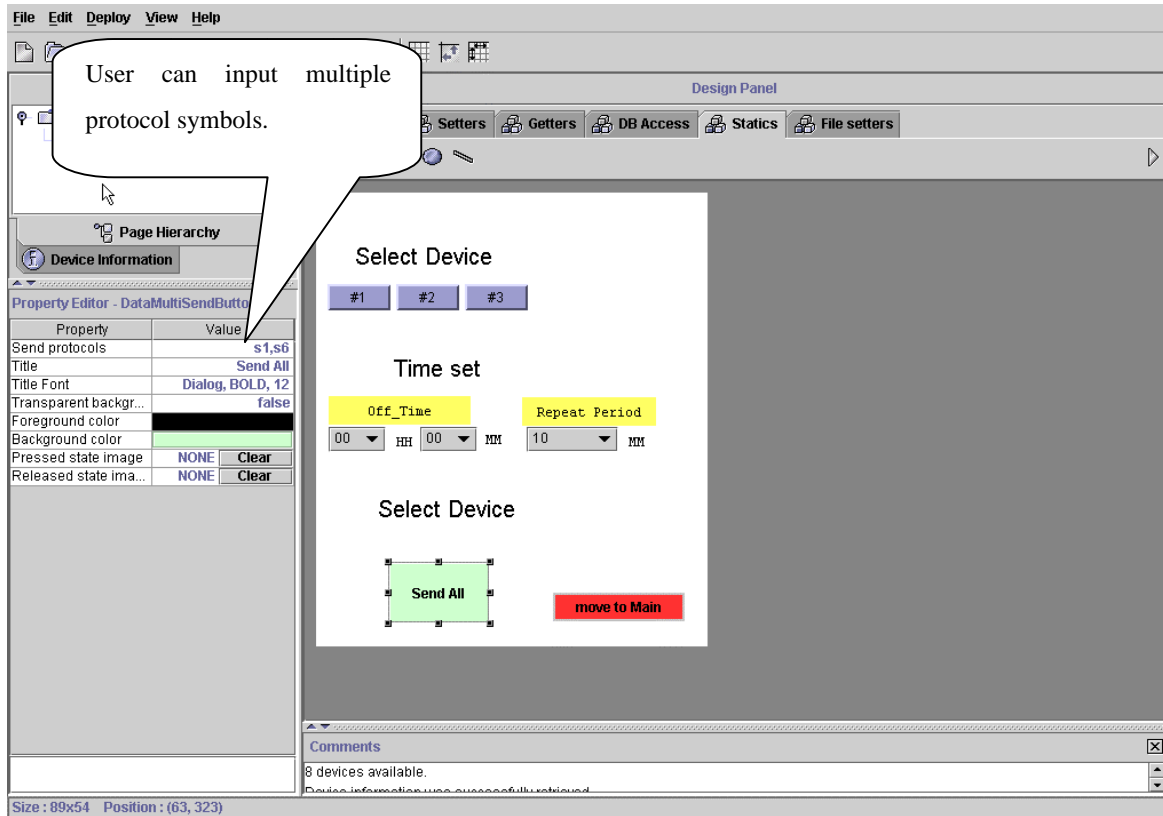
(1) Working

DataMultiSendButton sends multi protocol data to the Server to send set values to the selected Device



[Fig 3.14.1 DataMultiSendButton Example]

(2) Setting Property



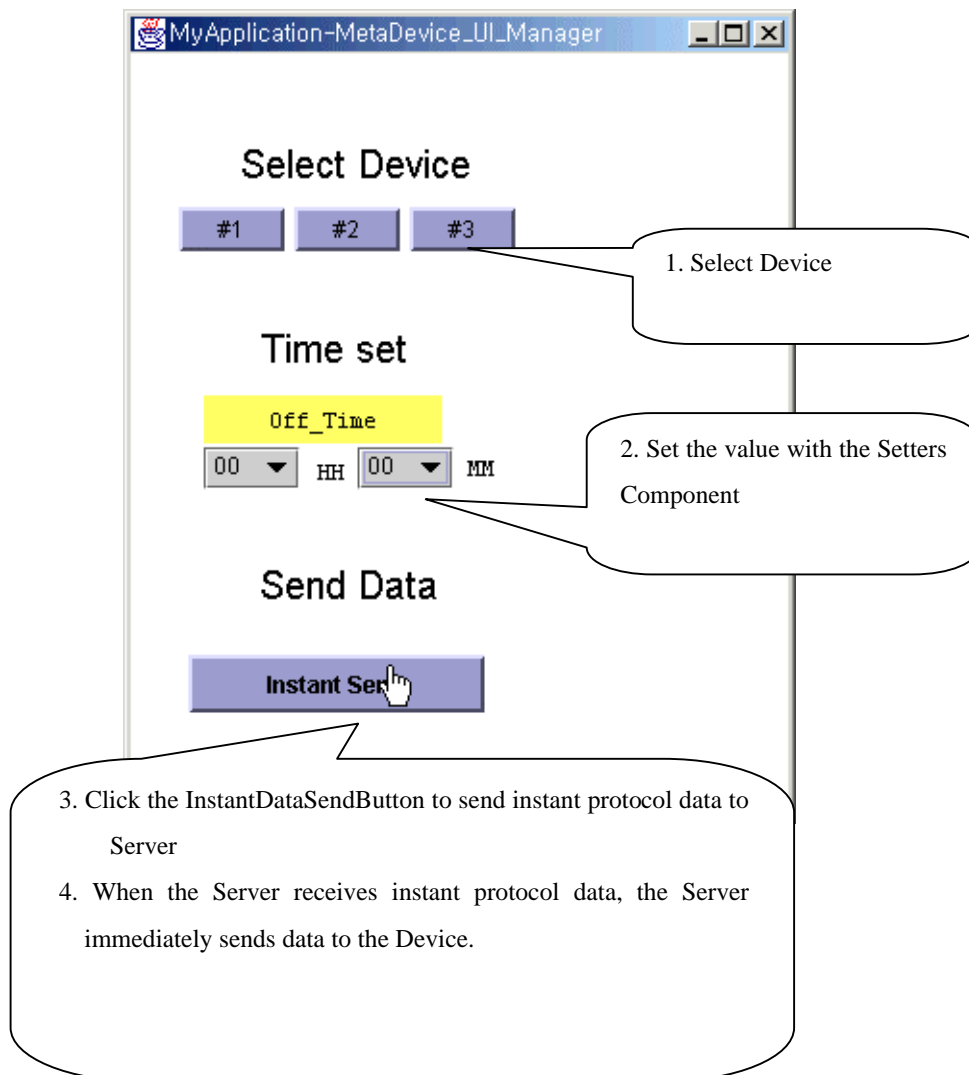
[Fig 3.14.2 Set DataMultiSendButton Property]

- Send protocol : Input Multiple Protocol Symbol.
- Pressed State Image : Image when the button is pressed
- Released state Image : Image when the button is released

4.7 InstantDataSendButton

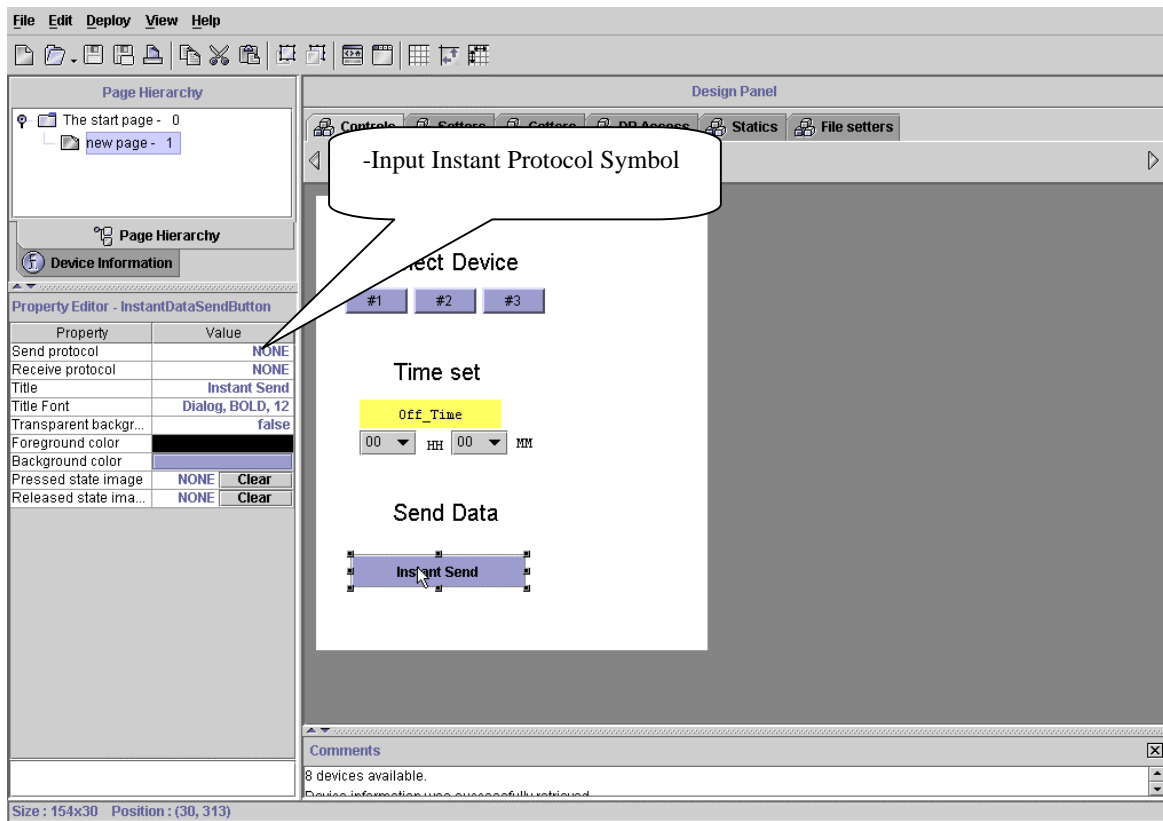
(1) Working at Run Time

The InstantDataSendButton sends one protocol to the Server so that the Server can send the protocol to the Device as soon as it receives protocol data from the client



[Fig 3.15.1 InstantDataSendButton Example]

(2) Setting Property at Design Time



[Fig 3.15.2 Setting InstantDataSendButton Property]

- Send protocol : Input Protocol Symbol
- Receive protocol : Receive Protocol Symbol
(Normally maintain "None" since ReceiveButton components are used for receiving data from the Server)
- Pressed State Image : Image when the button is pressed
- Released state Image : Image when the button is released

5. Setters Component Group

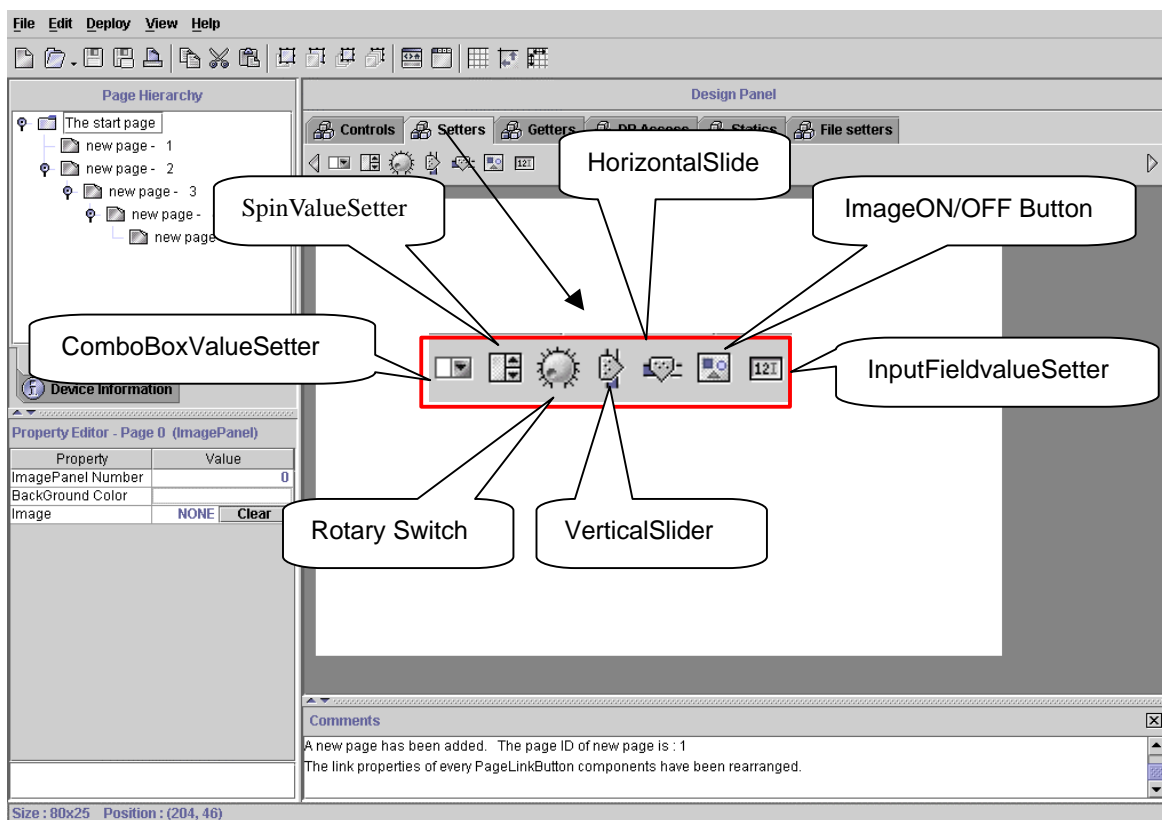
(1) What is **Setters Component**?

Setters Component is used for setting the value of the defined Run Time

Component variable type is numerical and Text.

(2) **Setters Components**

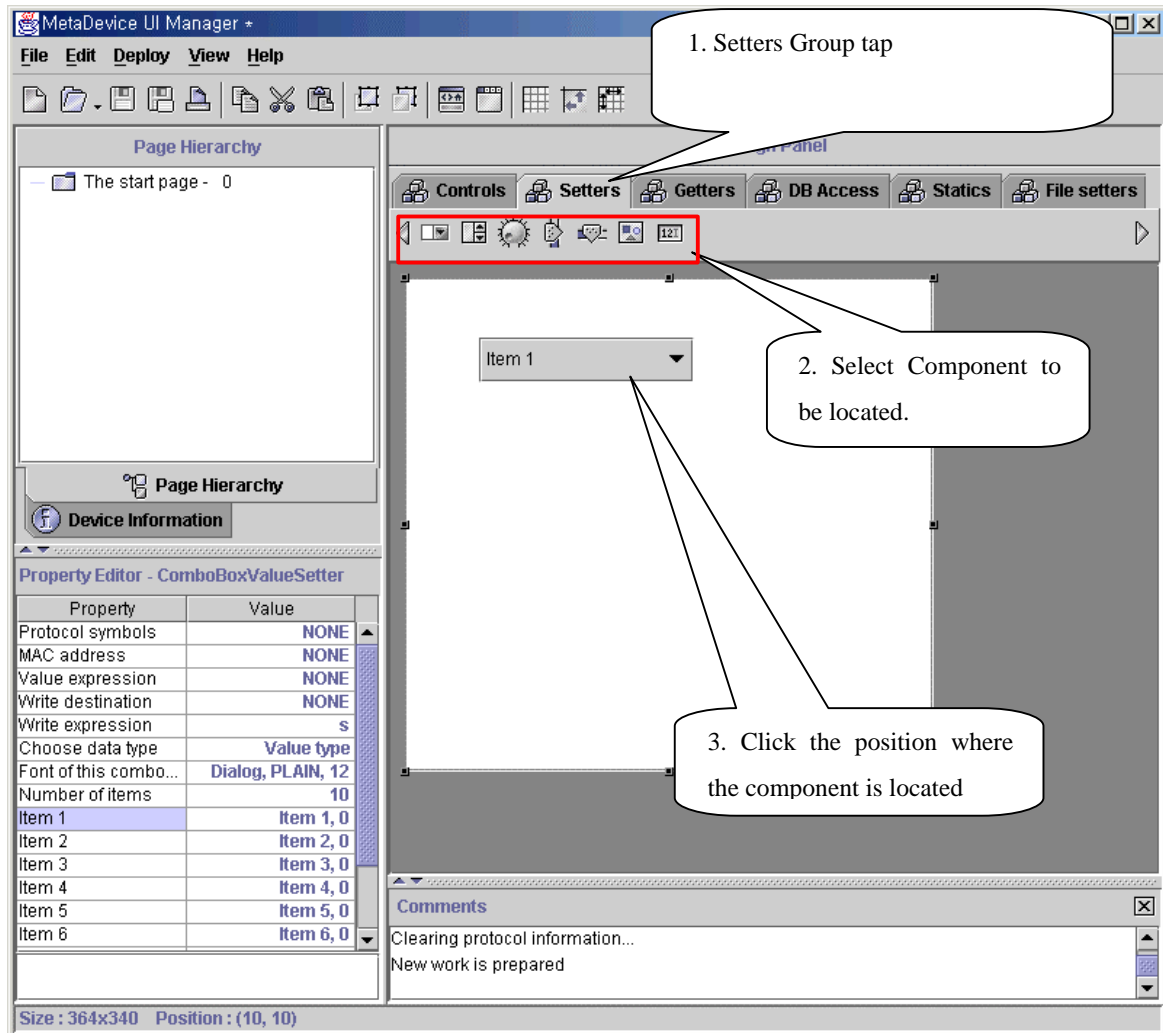
- ComboBoxValueSetter : The value item is pre-defined and the user can select one of items at Run Time
- SpinValueSetter : Set value using up/down key.
- Rotary Switch : Set value using rotary switch.
- VerticalSliderSwitch : Set value using vertical slider switch
- HorizontalSliderSwitch : Set value using horizontal slider switch
- ImageON/OFF Button : Set 0 or 1 for value using On/Off toggle image button.
- InputFieldvalueSetter : Set value using user input



[Fig 3.16.1 Setters Components]

(3) Setters Components

- Select Setters Component Group Tab.
- Click the position where the Setters Component should be located on the DesignPanel
- Set Property by double clicking component



[Fig 3.16.2 ComboBoxValueSetter]

5.1 ComboBoxValueSetter

(1) Working at Run Time

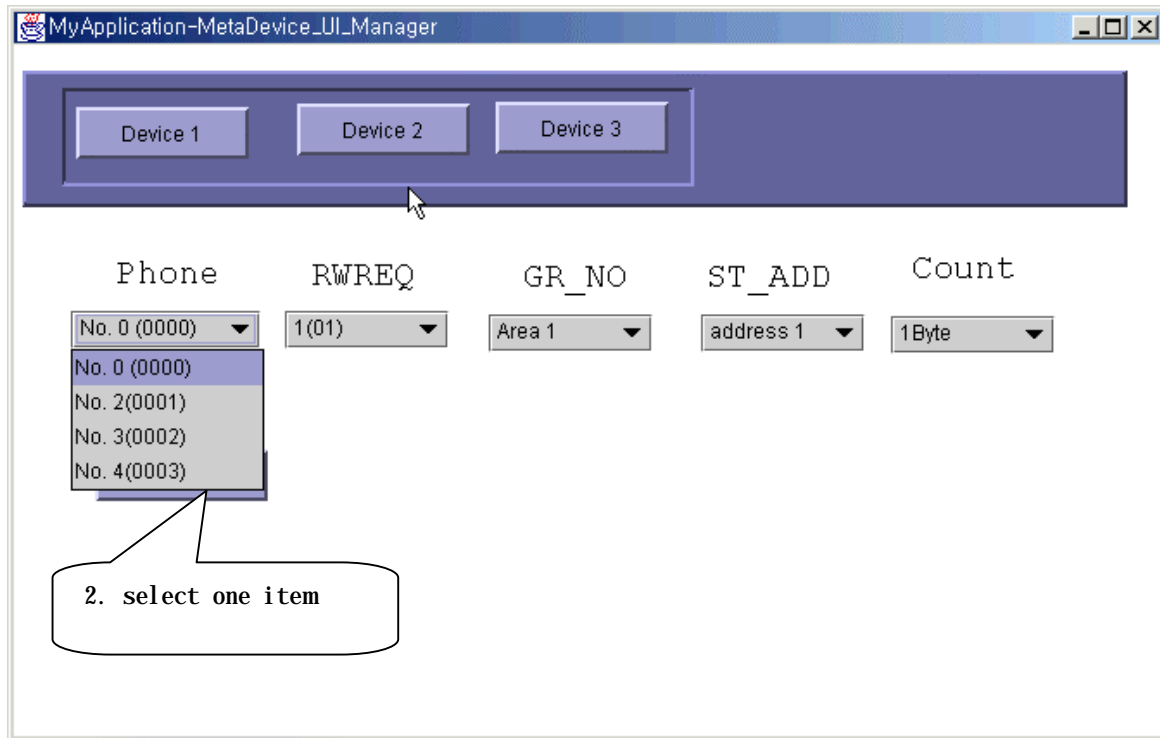
Choose Data Type as Text Type or Value Type

Text Type variable : v0,v1,....

Value Type variable : m0, m1, m2 ..

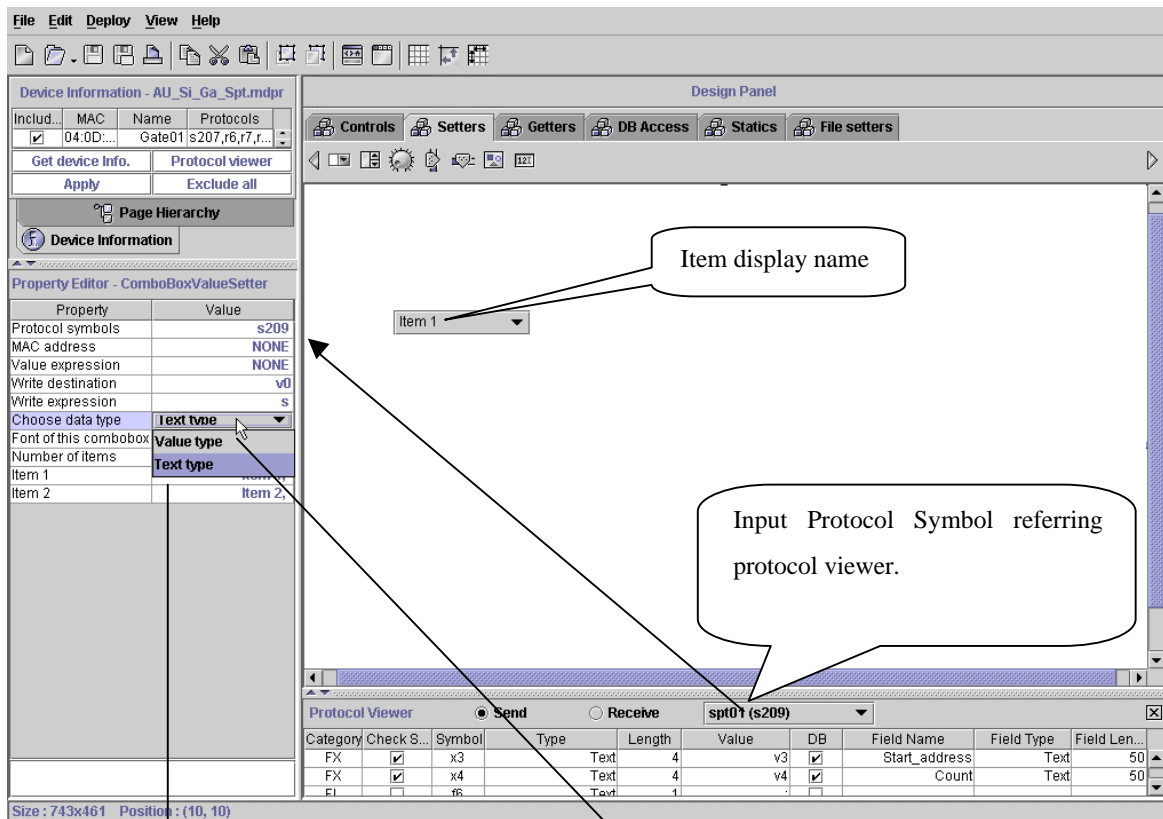
Value Type is numerical data such as integer, short, long, double.

ComboBoxValueSetter can select one item at Run Time

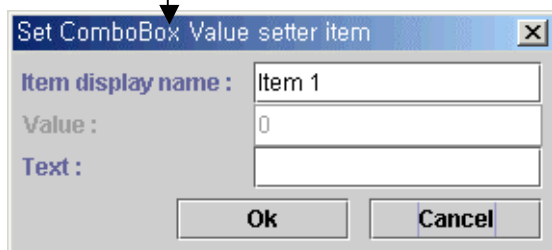


[Fig 3.17.1 ComboBoxValueSetter Example]

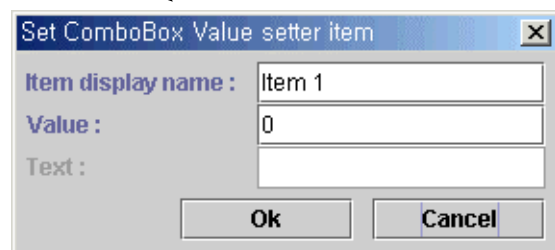
(2) Setting Property at Design Time



[Fig 3.17.2 Setting ComboBoxValueSetter Property]



[Fig 3.17.3 Text Type]



[Fig 3.17.4 Value Type]

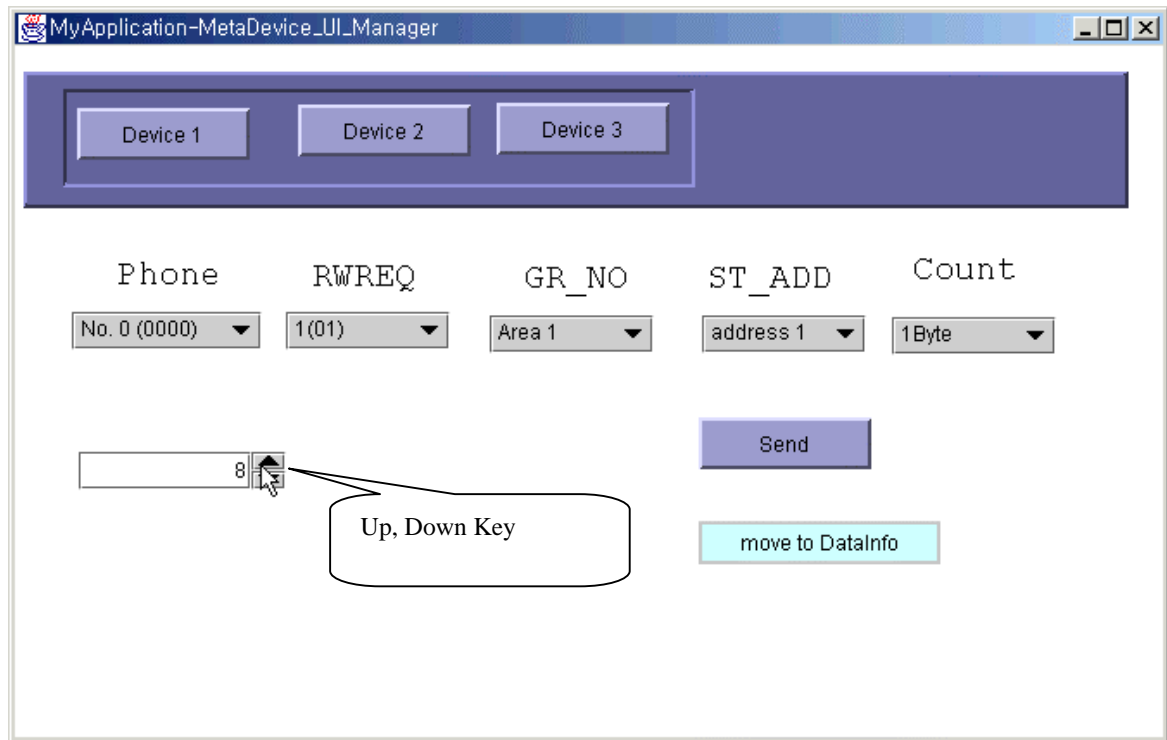
- Protocol symbols : Input the Protocol Symbol.
- MAC address : Input the Mac Number of Device where data is to be sent.
 - If NONE , the Mac number of the DeviceSelector will be applied.
- Value expression : Displays value of receive protocol.
- Write destination: Input user protocol variable.
- Write expression : S or !S
 - s : the value that user input
 - ex) s : m0, s*10: m0*10.
 - !S : “!” is logical operation NOT
 - (can not be applied to the Text Type)
- Choose data type : Set ComboBox data type.
- Font of this comboBoxvalueSetter : ComboBox value type
 - Text type : Input Text.
 - Value Type : Input Numerical data.
- Number of items : The number of items to be displayed in a ComboBox at Run Time
- item1.. : First item
 - Item type is Text type :[Fig 3.17.3] shows input Text
 - Item type is Value type : [Fig 3.17.4] shows input Value
- Item display name : Actually, the item display name is displayed at Run Time
 - But the value or Text is assigned to the variable
- Text : Text value
- Value : Numerical value

5.2 SpinValueSetter

(1) Working at Run Time

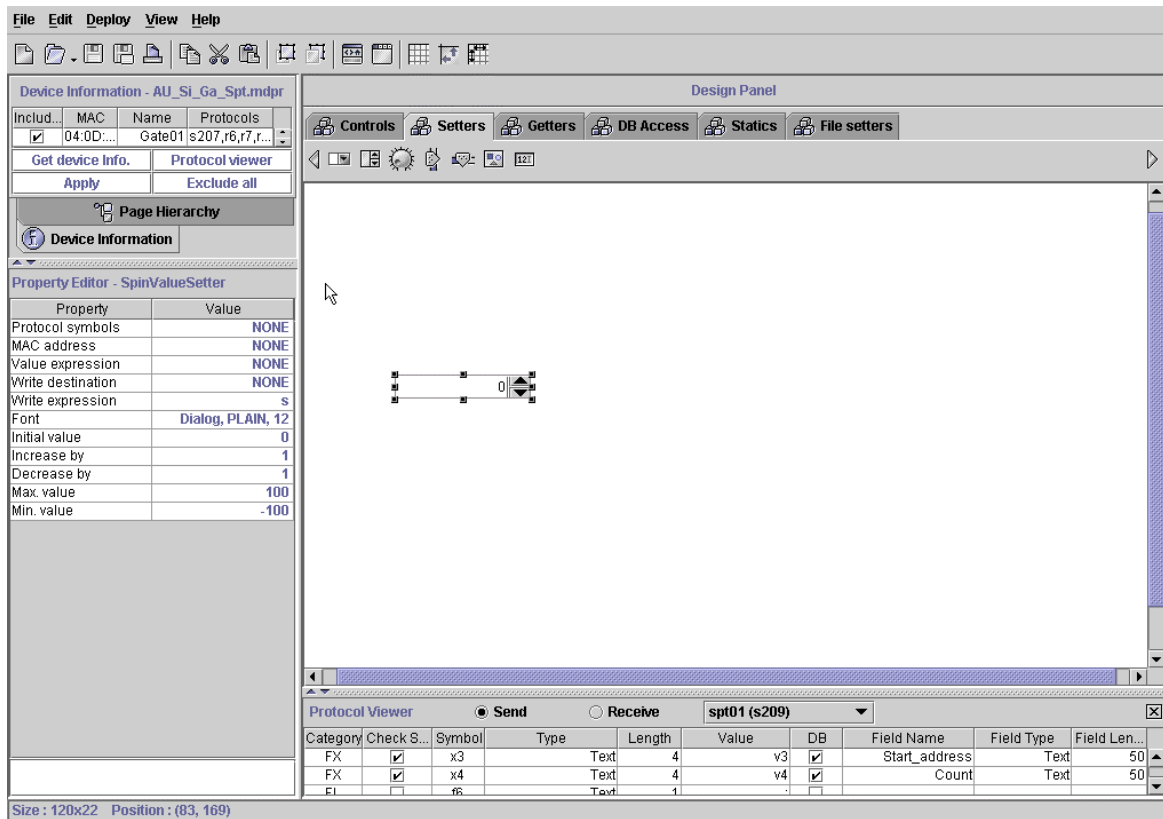
Set the value using Up / Down.Key

Protocol data type is numerical(value type). Set value between Min and Max



[Fig 3.18.1 SpinValueSetter]

(2) Setting Property at Design Time



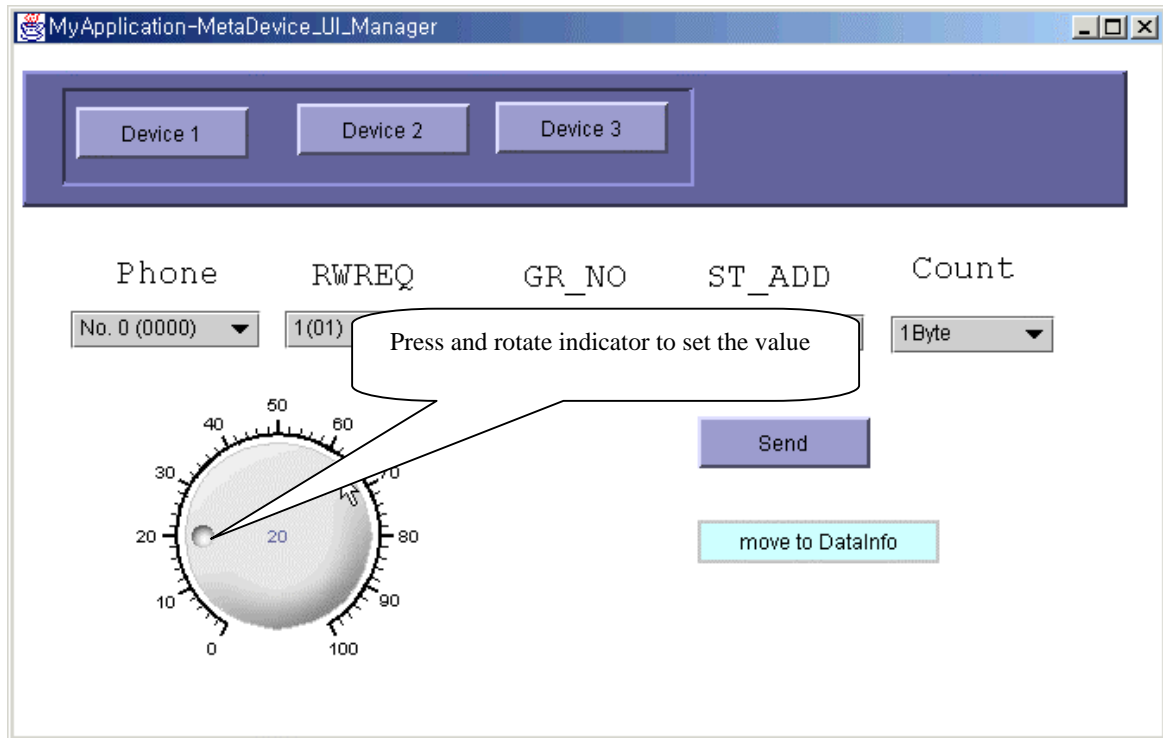
[Fig 3.18.2 Setting SpinValueSetter Property]

- Font : Text Font
- Initial Value : Set initial value
- Increase by : Increasing value
- Decrease by : Decreasing value
- Max value : Maximum value
- Min value : Minimum value

5.3 RotarySwitch

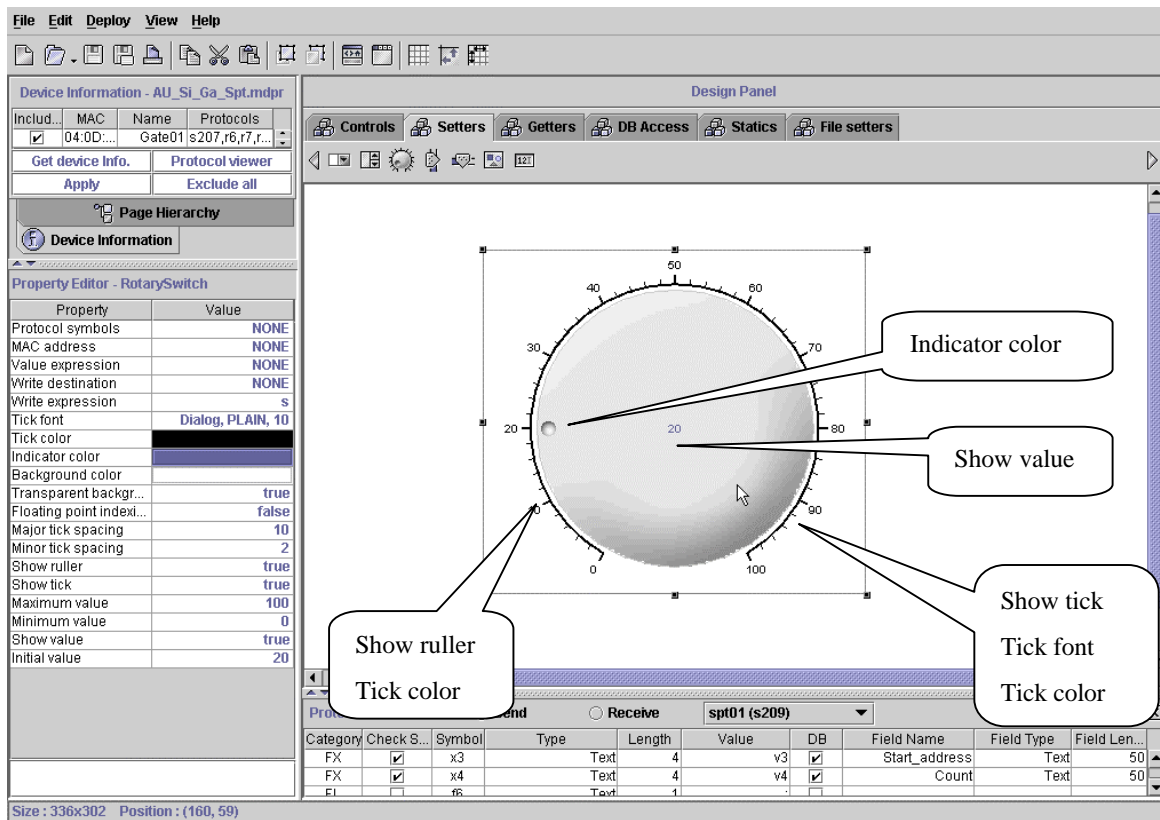
(1) Working at Run Time

Set value by rotating switch.



[Fig 3.19.1 RotarySwitch Working]

(2) Setting Property at Design Time



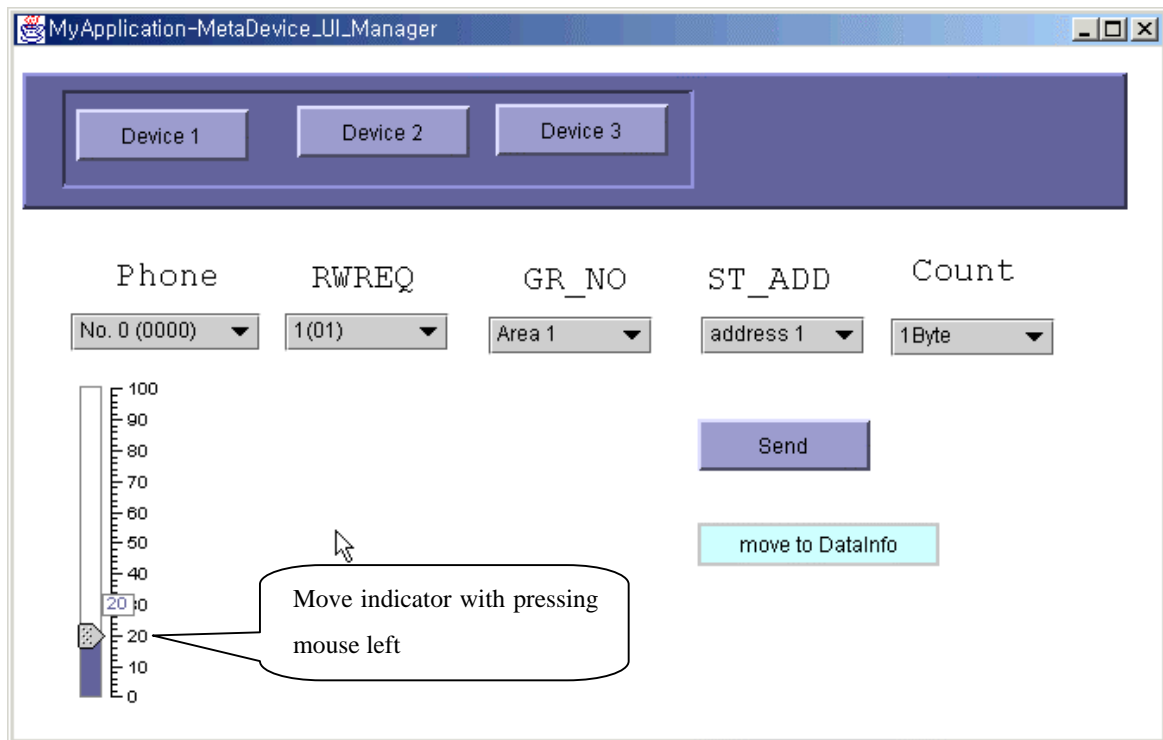
[Fig 3.19.2 Setting RotarySwitch Property]

- Floating point indexing : If True , the value is displayed as real value
- Major tick spacing : the interval of large graduations
- Minor tick spacing : the interval of small graduations
- show ruler : If True, the ruler is shown.
- show tick : If True, the number selected in Major tick Spacing is displayed
- Maximum value : maximum value that this gauge can display
- Minimum value : minimum value that this gauge can display
- show value : If set as True, displays current value
- Initial value : sets initial value

5.4 VerticalSliderSwitch

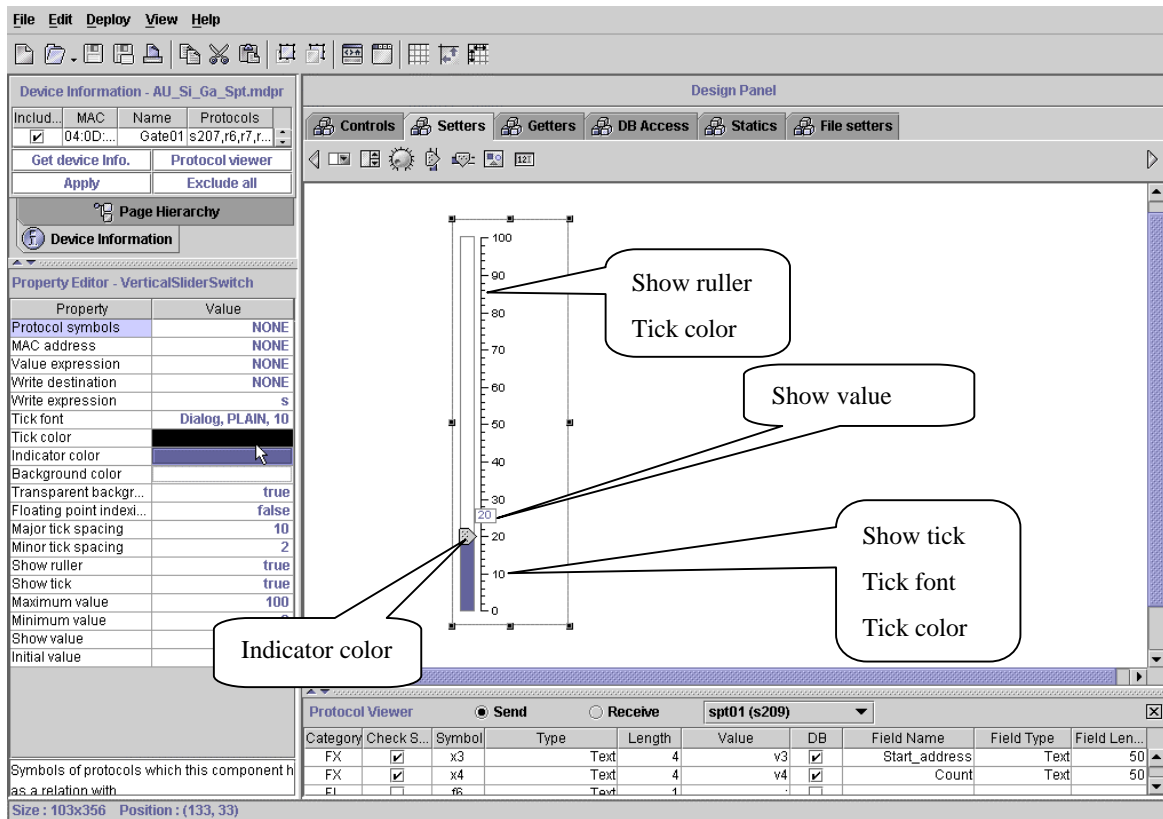
(1) Working at Run Time

Sets value by having the indicator move up or down



[Fig 3.20.1 VerticalSliderswitch]

(2) Setting Property at Design Time



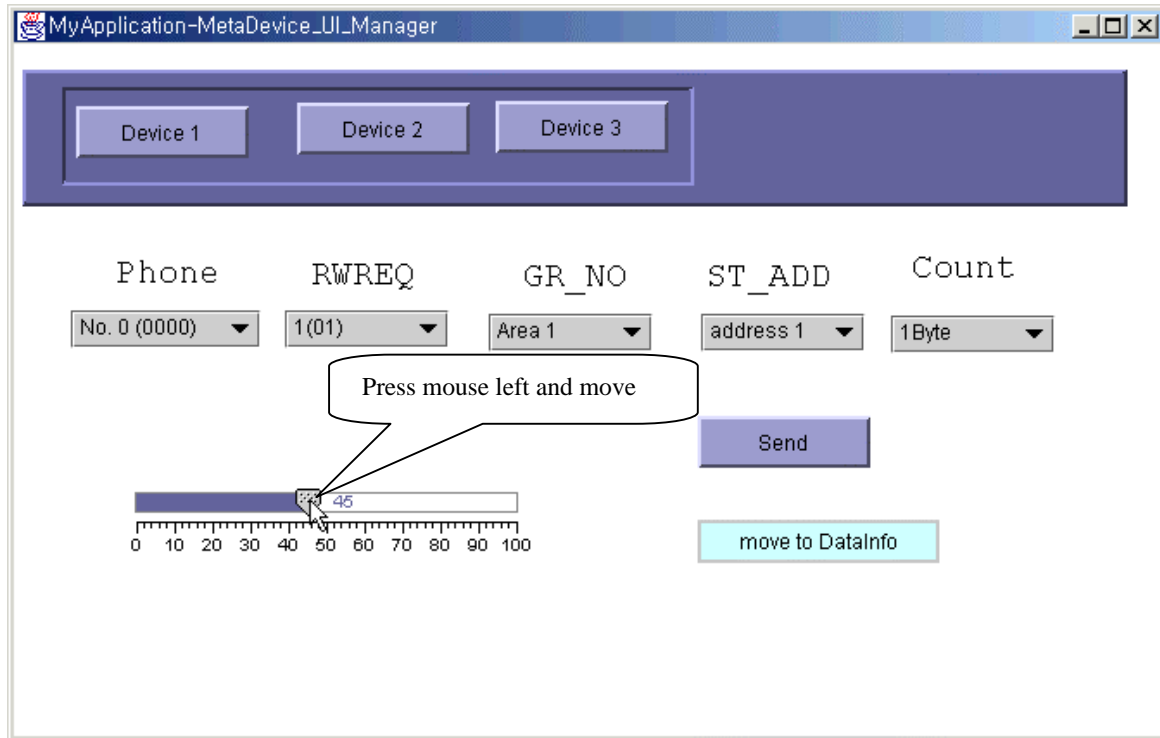
[Fig 3.20.2 Setting VerticalSliderSwitch]

- Floating point indexing : If True , the value is displayed as real value
- show ruler : If True, the ruler is shown.
- show tick : If True, tick with numbers is shown.
- Maximum value : maximum value that gauge can display
- Minimum value : minimum value that gauge can display
- show value : If set as True, displays current value.
- Initial value : Sets initial value.

5.5 HorizontalSliderSwitch

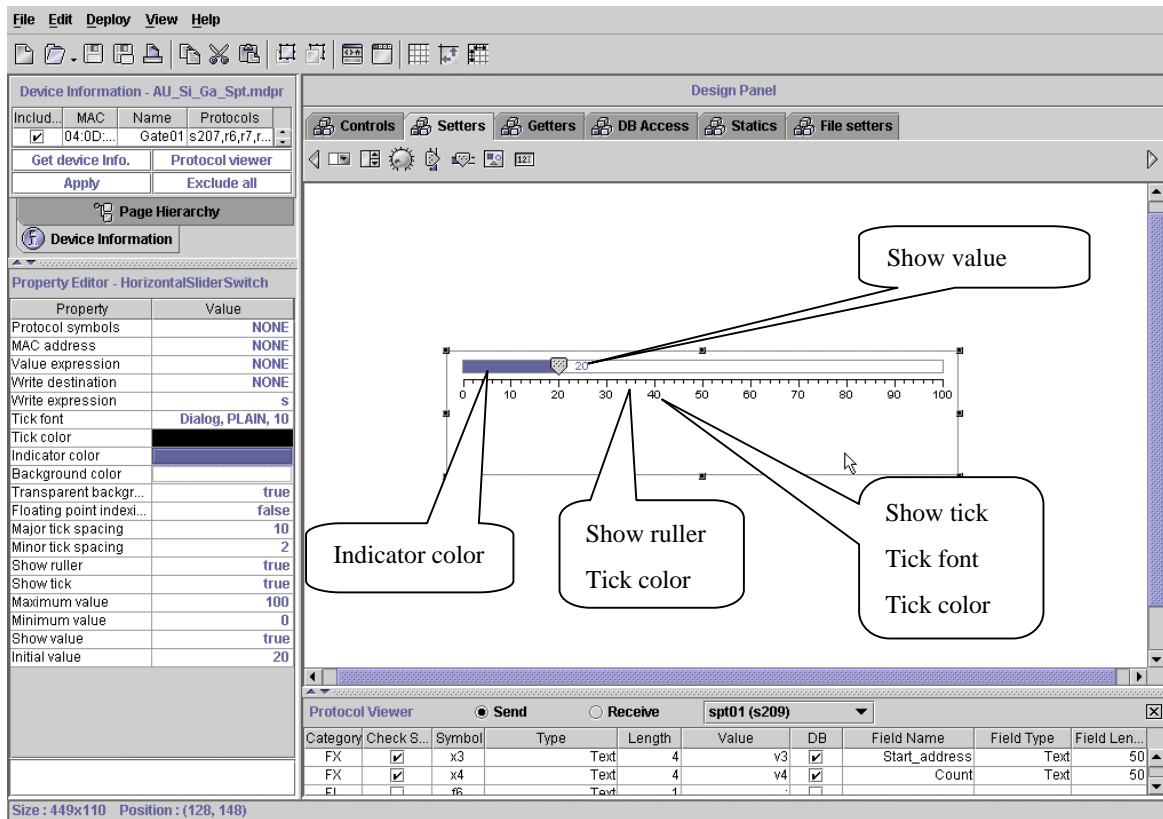
(1) Working at Run Time

Sets value by having the indicator move left or right.



[Fig 3.21.1 HorizontalSliderSwitch]

(2) Setting Property at Design Time



[Fig 3.21.2 Setting HorizontalSliderSwitch Property]

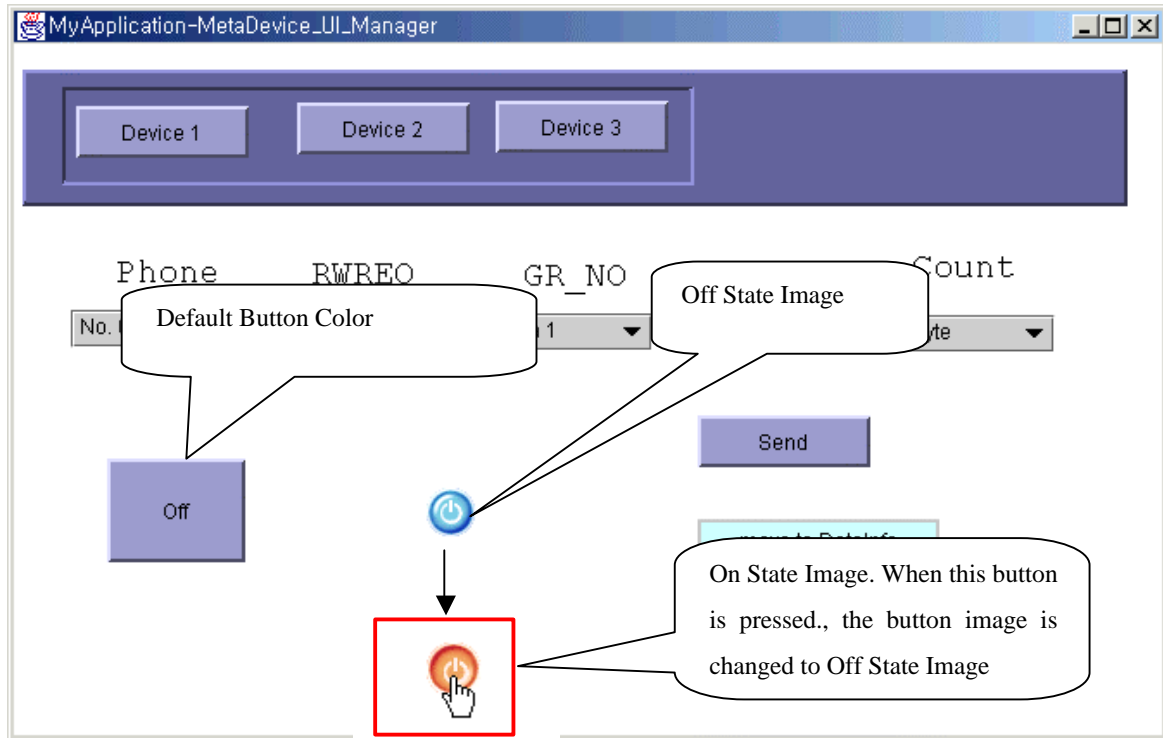
- Floating point indexing : If True , the value is displayed as real value
- Major tick spacing : the interval of large graduations
- Minor tick spacing : the interval of small graduations
- show ruler : If True, the ruler is shown.
- show tick : If True, tick with number is shown.
- Maximum value : maximum value that gauge can display
- Minimum value : minimum value that gauge can display
- show value : If set as True, displays current value
- Initial value : sets initial value.

5.6. ImageOnOffButton

(1) Working at Run Time

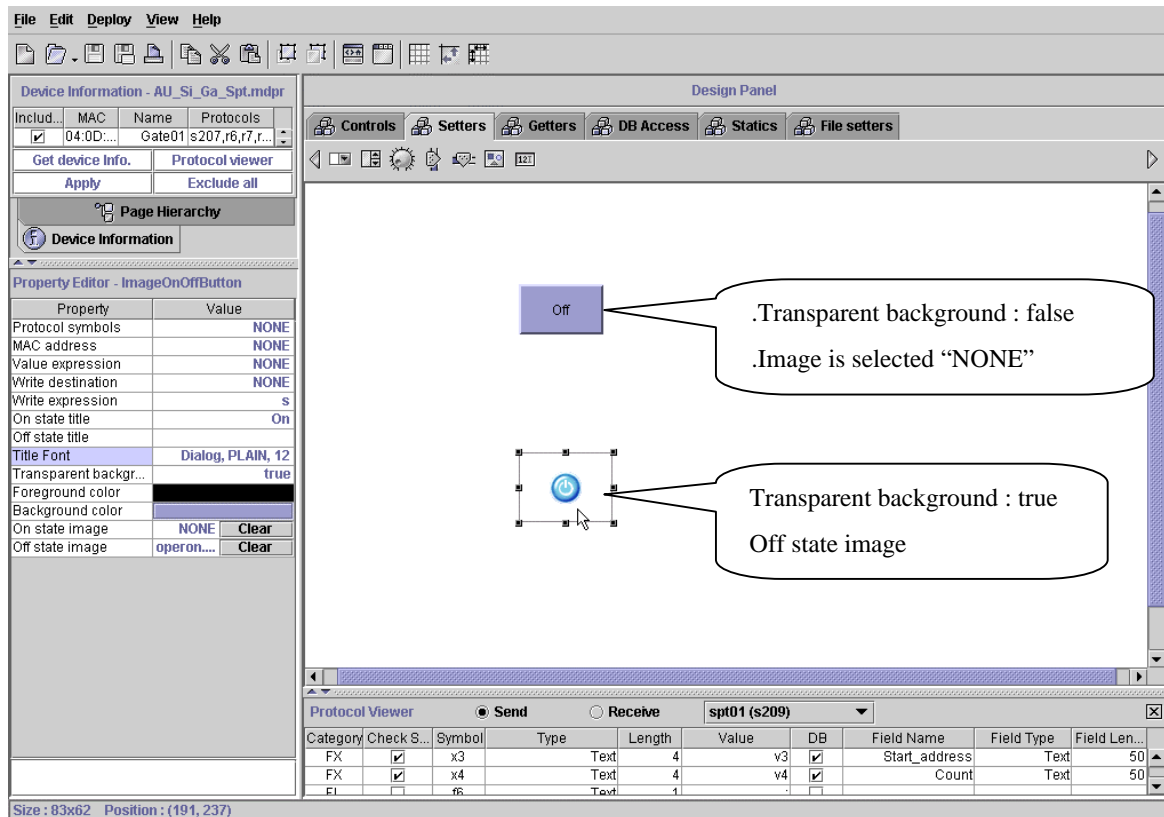
Whenever you press this button, the value will be toggled as 0, 1

On/Off State value can be defined by the user.



[Fig 3.22.1 ImageOnOffButton]

(2) Setting Property at Design Time



[Fig 3.22.2 Setting ImageOnOffButton Property]

- On State Title : On State Button Text
- Off State Title : Off State Button Text
- Title Font : Title Text Font
- Foreground color : Text color
- Background color : Button color
- On state image : On State Image
- Off state image : Off State Image

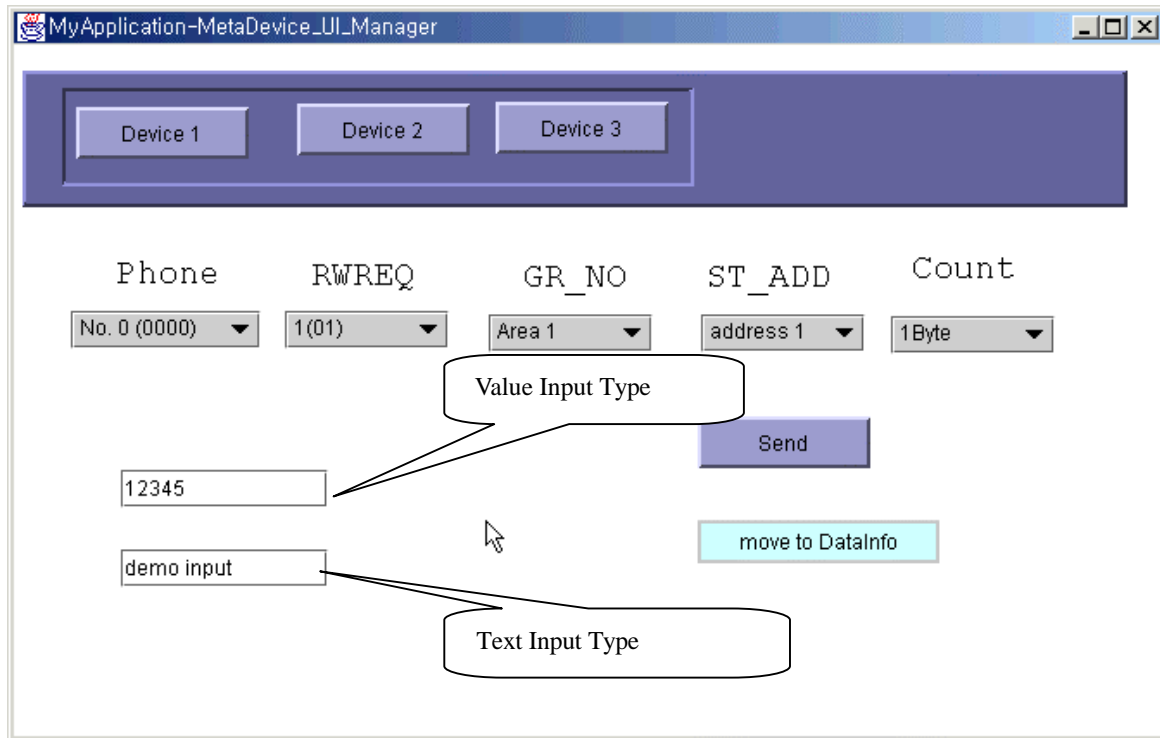
5.7 InputFieldValueSetter

(1) Working at Run Time

Sets value by receiving input data from the user.

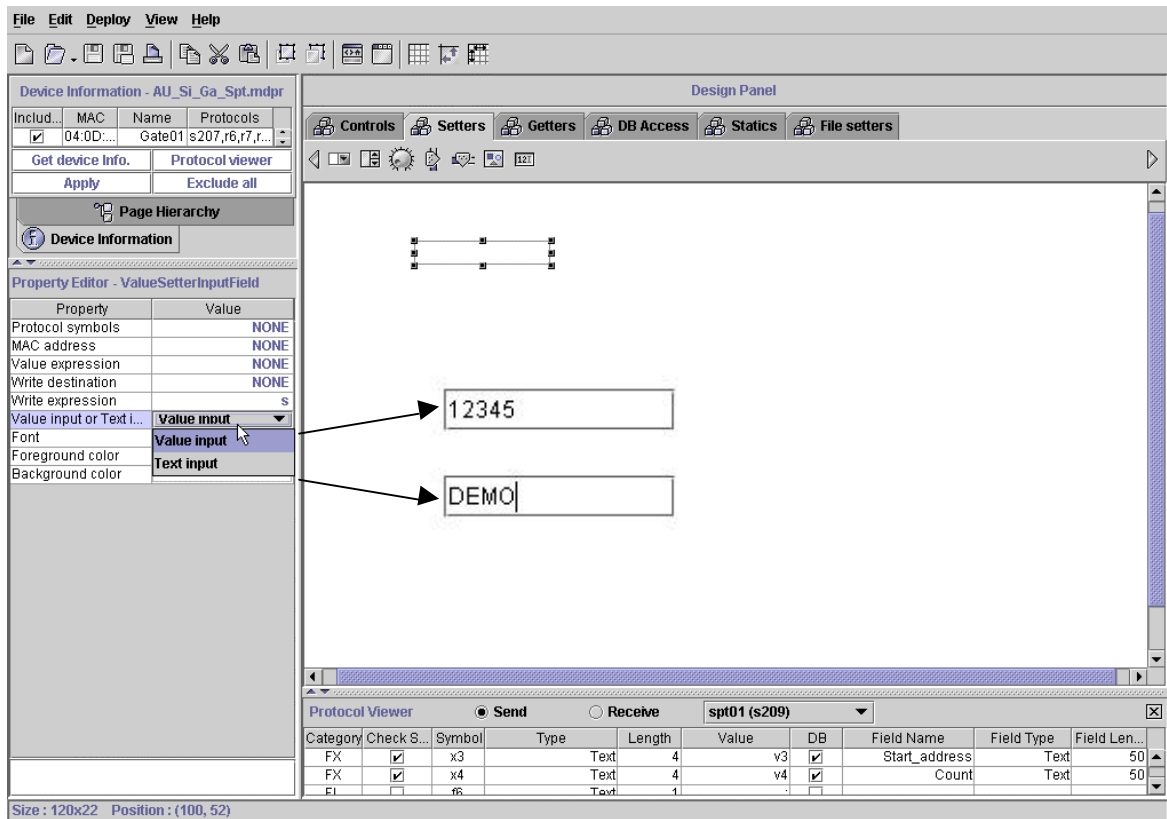
Data Type is Text or Value input

Text variables : v0, v1 , value variables : m0, m1.



[Fig 3.23.1 InputFieldValueSetter]

(2) Setting Property at Design Time



[Fig 3.23.2 Setting InputFieldValueSetter Property]

- Value input or Text input:

Value input : user can type numerical number

Text input : user can type text

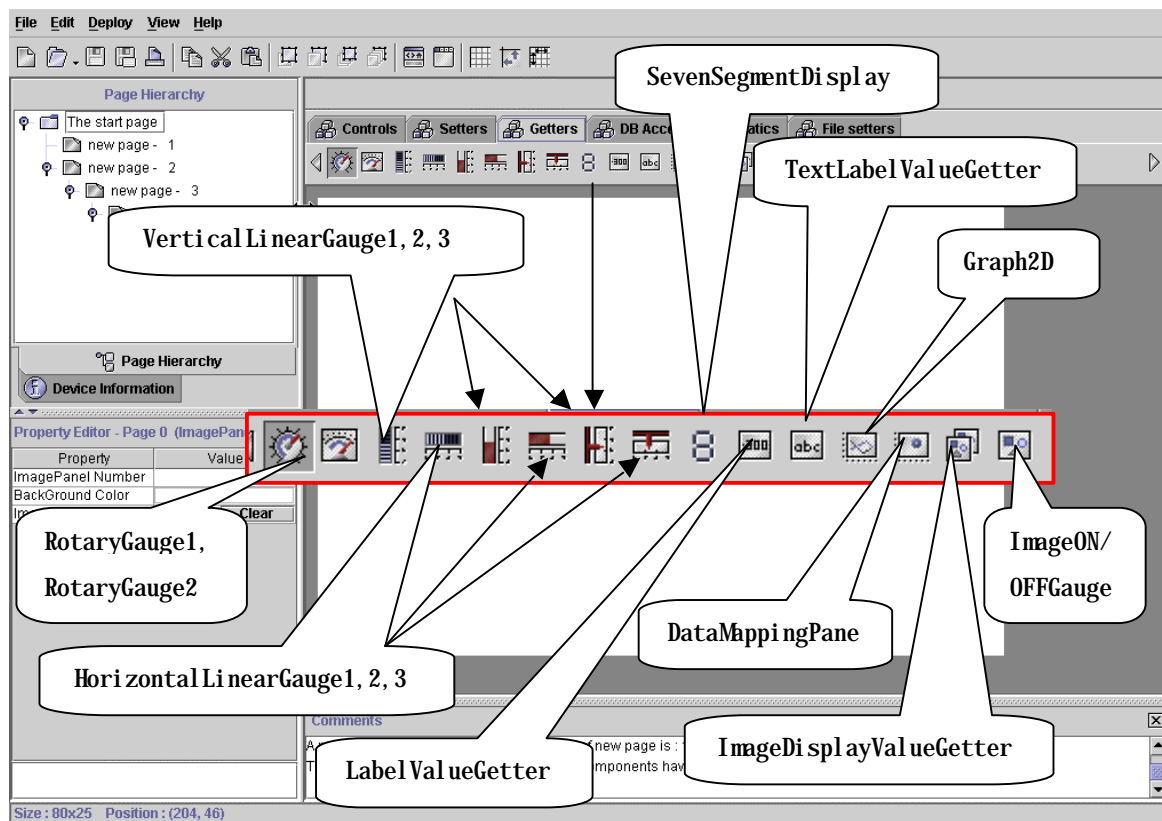
6. Getters Component Group

(1) What is **Getters Component**?

Getters Component displays variable data within the Receive Protocol.

(2) **Getters Component**

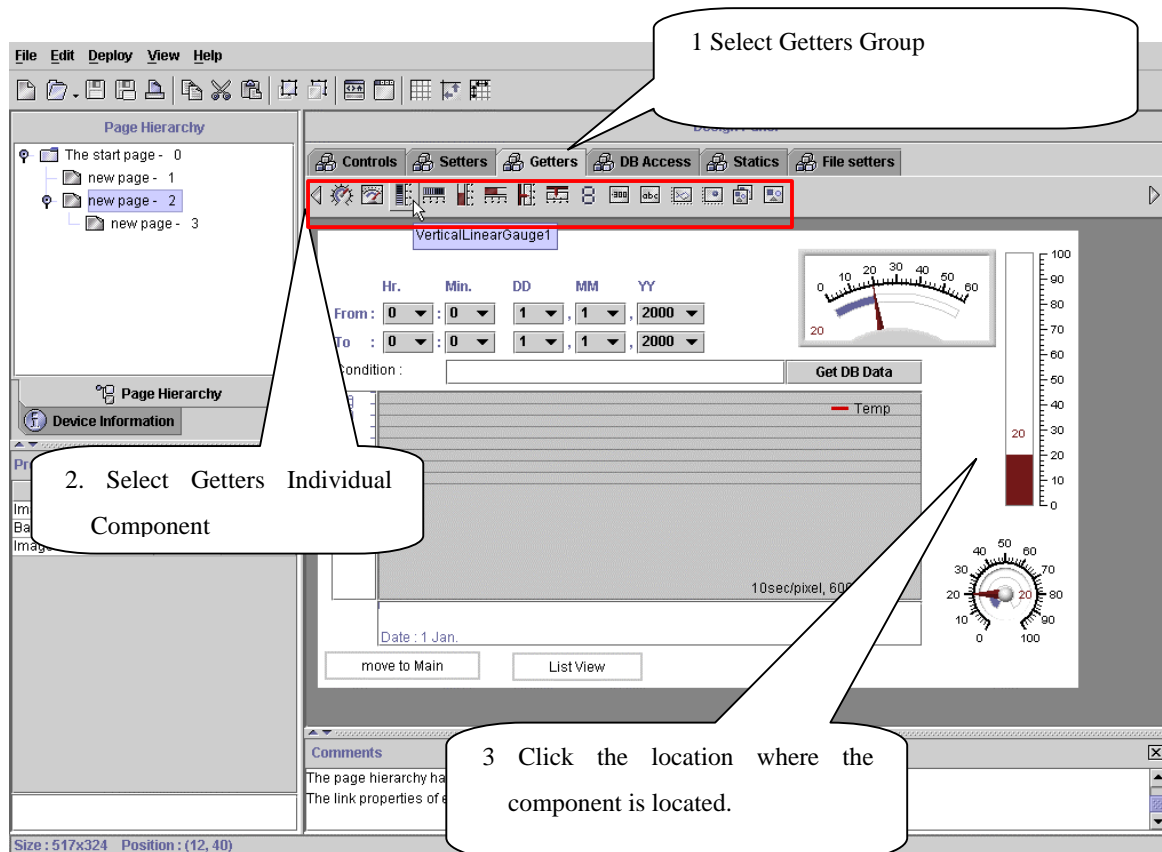
- RotaryGauge1, RotaryGauge2 : Displays value in the rotary gauge.
- VerticalLinearGauge1,2,3 : Displays value in the vertical gauge.
- HorizontalLinearGauge1,2,3 : Displays value in the horizontal gauge..
- SevenSegmentDisplay : Displays value in the seven segment displayer..
- LabelValueGetter : Displays value in the label..
- TextLabelValueGetter: Displays Text value in the label..
- Graph2D : Displays value in the 2D Graph.
- DataMappingPanel : Displays value in the 2D position map
- ImageDisplayValueGetter : Displays value with image.
- ImageON/OFFGauge : Displays On/Off value with image.



[Fig 3.24.1 Getters Components]

(3) Getters Component

- Select Getters from the Design Panel Tab.
- Click the position where the component should be located
- Edit Getters Component Property.



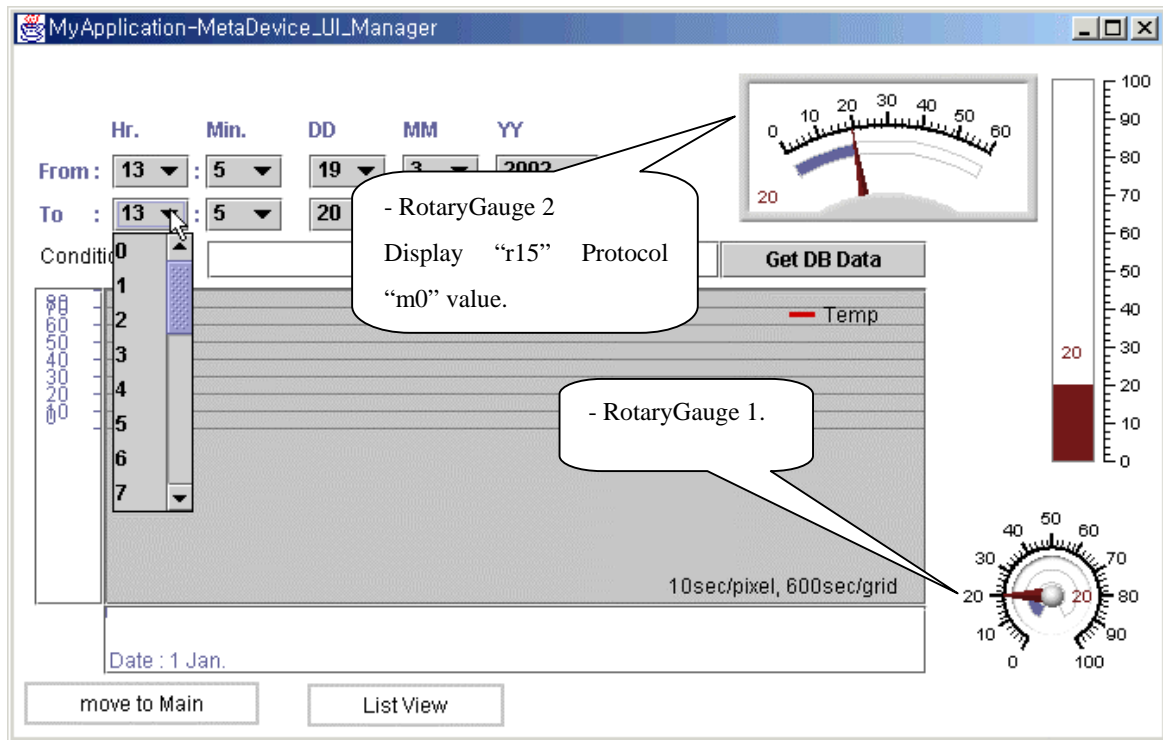
[Fig 3.24.2 RotaryGaug1,2 Example]

6.1 RotaryGauge1, RotaryGauge2

(1) Working at Run Time

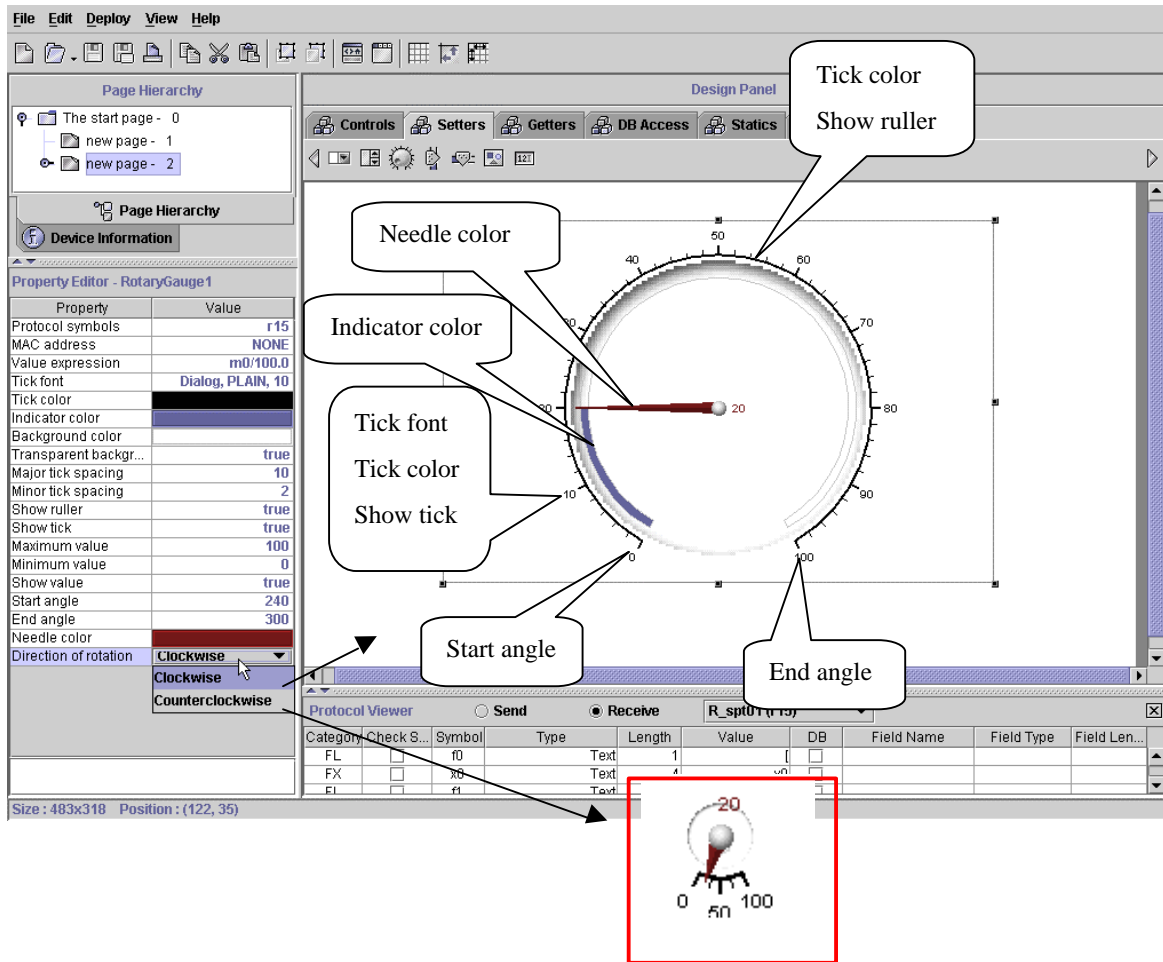
Display value received from the Server in the rotary gauge

The value is Receive Protocol (r0, r1, r3..) variable (m0, m1, m2..)



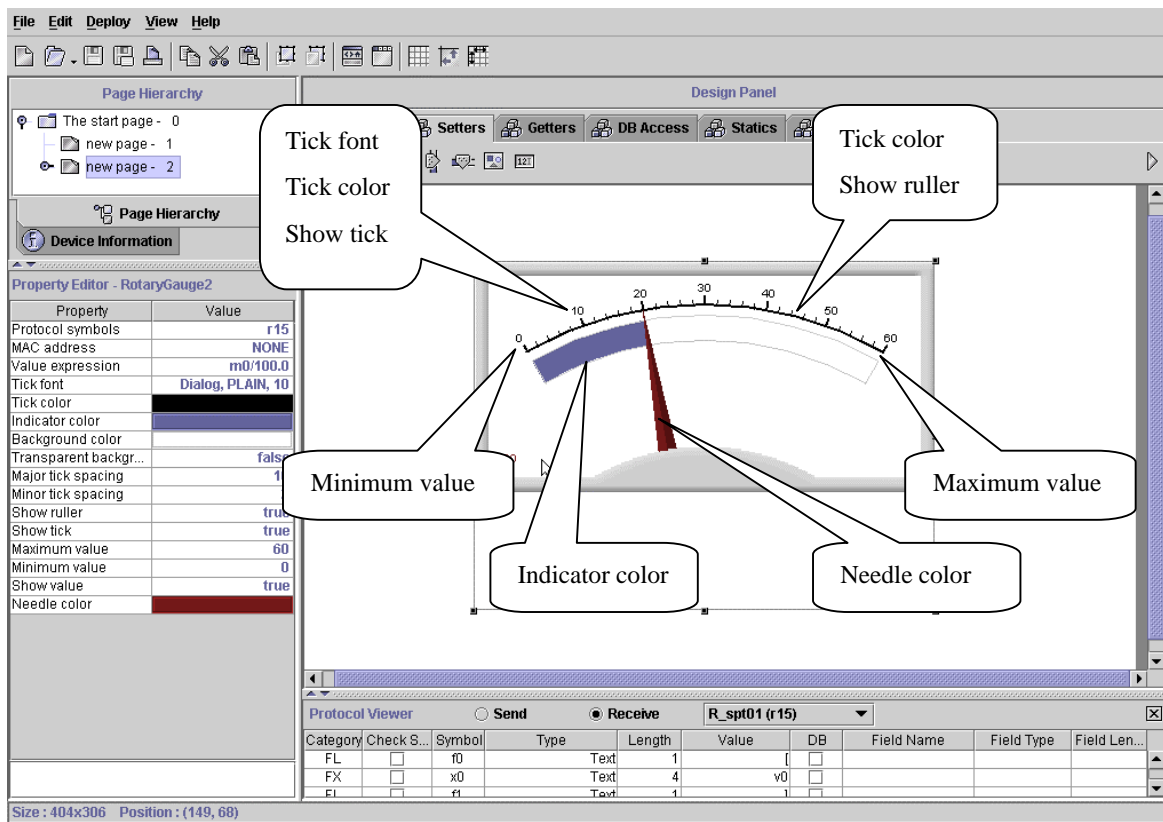
[Fig 3.25.1 Working RotaryGauge1,2 Example]

(2) Setting Property at Design Time



[Fig 3.25.2 Setting RotaryGauge1 Property]

- Value expression : Displays number of matching value and its operation
- Major tick spacing : The interval of large graduations
- Minor tick spacing : The interval of small graduations
- Show ruler : Displays ruler
- Show tick :Displays number
- Maximum value : Maximum value that gauge can display
- Minimum value : Minimum value that gauge can display
- Show value : Shows display value numerically at the center of this component.
- Needle color : Needle Color
- Direction of destination : Clockwise/counterclockwise numbering order



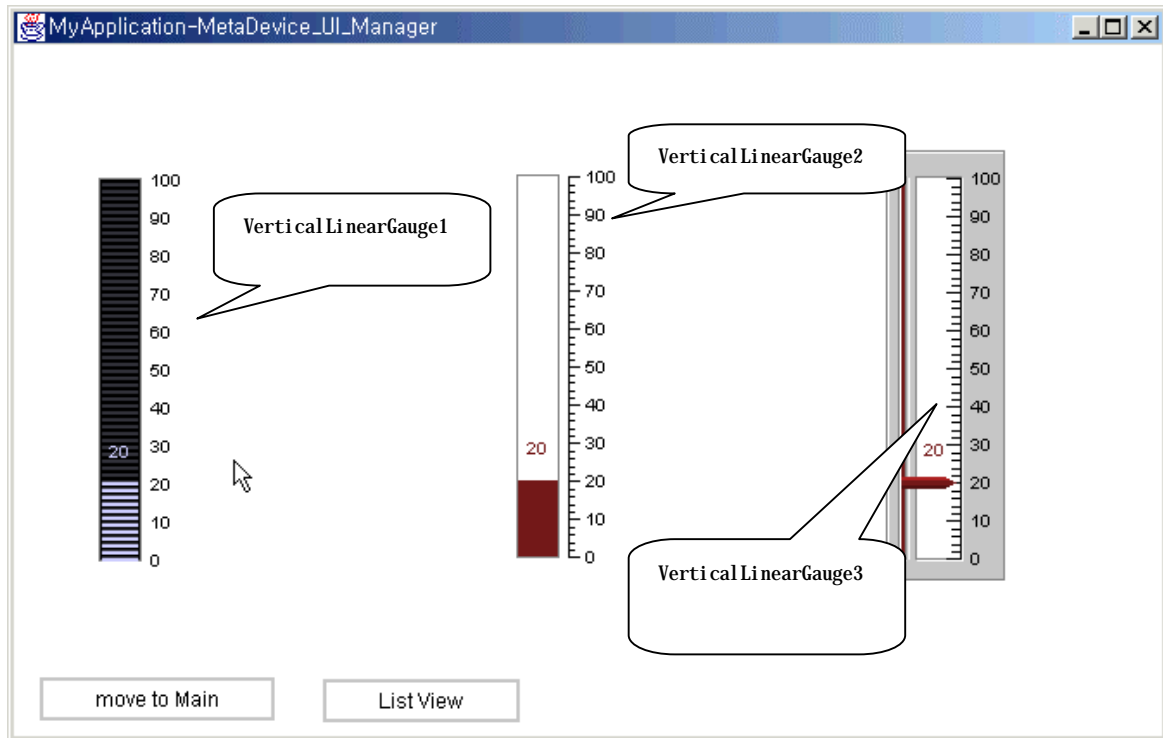
[Fig 3.25.3 Setting RotaryGauge2 Property]

6.2 VerticalLinearGauge1, VerticalLinearGauge2, VerticalLinearGauge3

(1) Working at Run Time

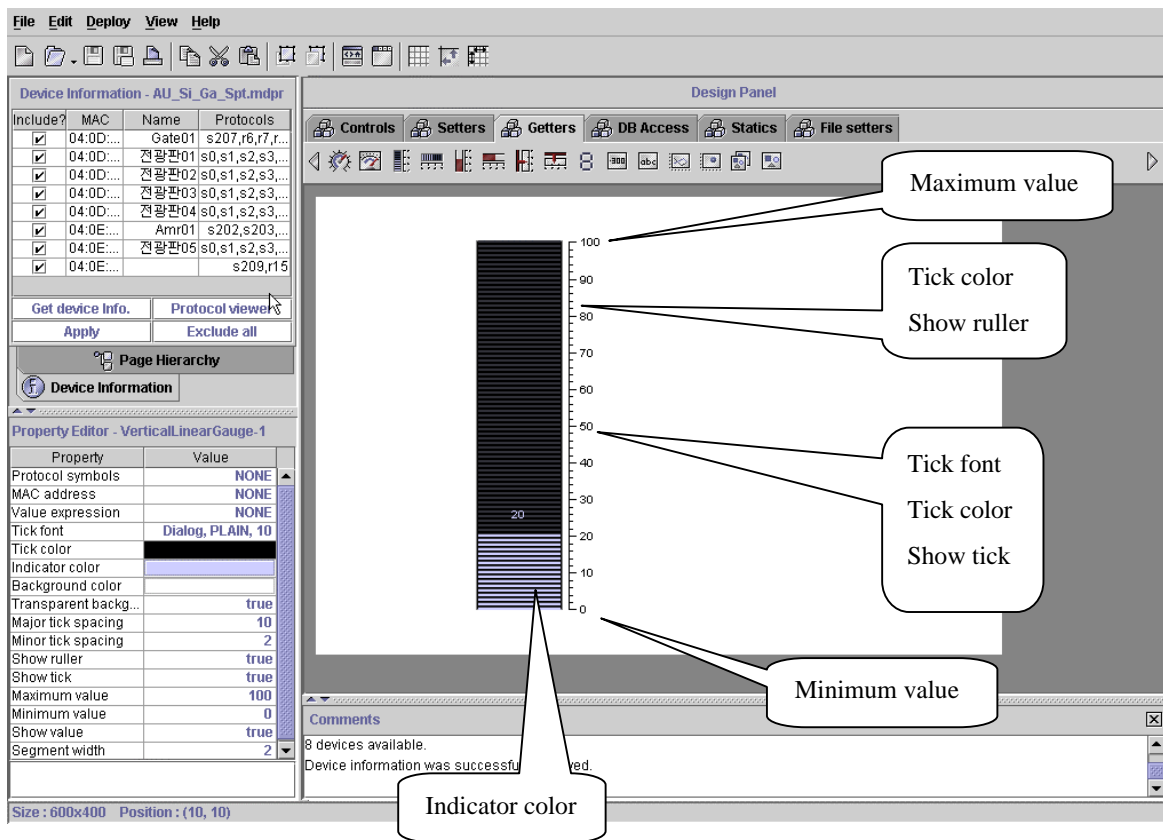
Displays value received from the Server on the vertical gauge.

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..).



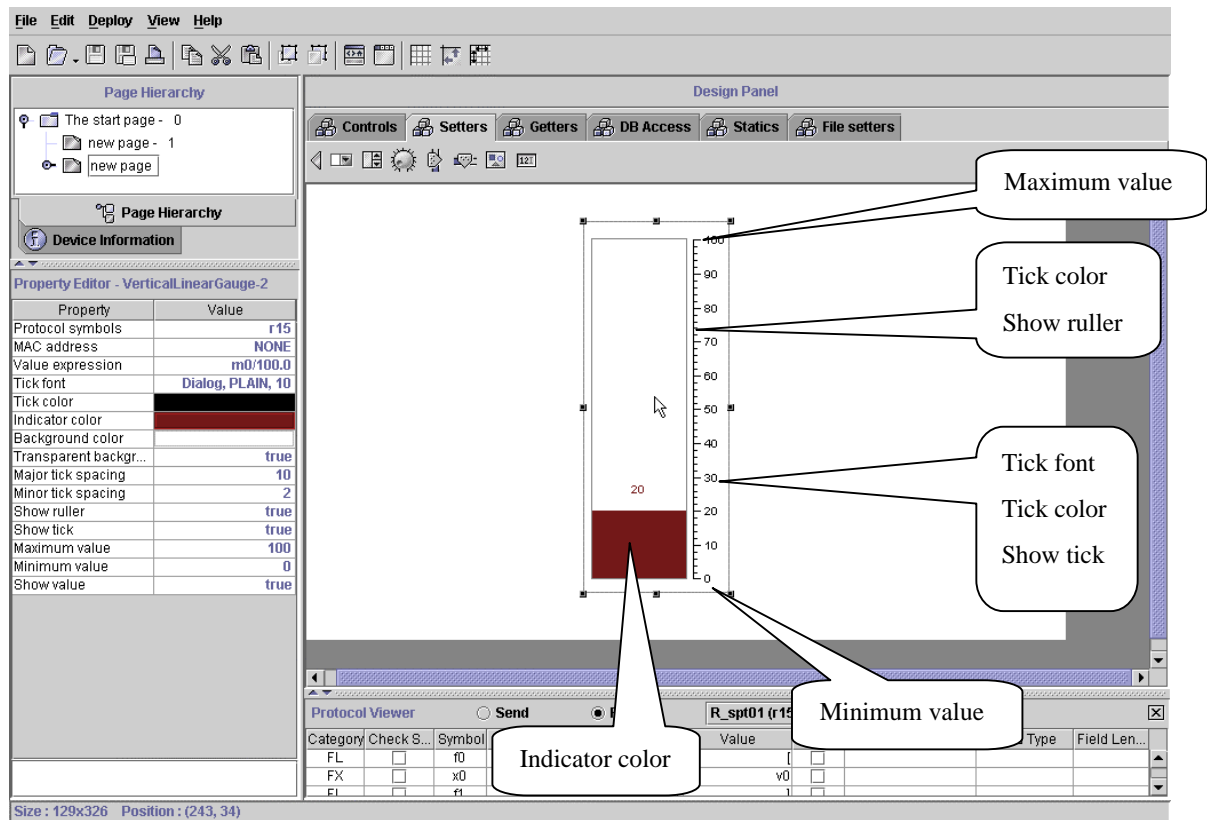
[Fig 3.26.1 VerticalLinearGauge1, VerticalLinearGauge2,
VerticalLinearGauge3]

(3) Setting Property at Design Time



[Fig 3.26.2 Set VerticalLinearGauge1 Property]

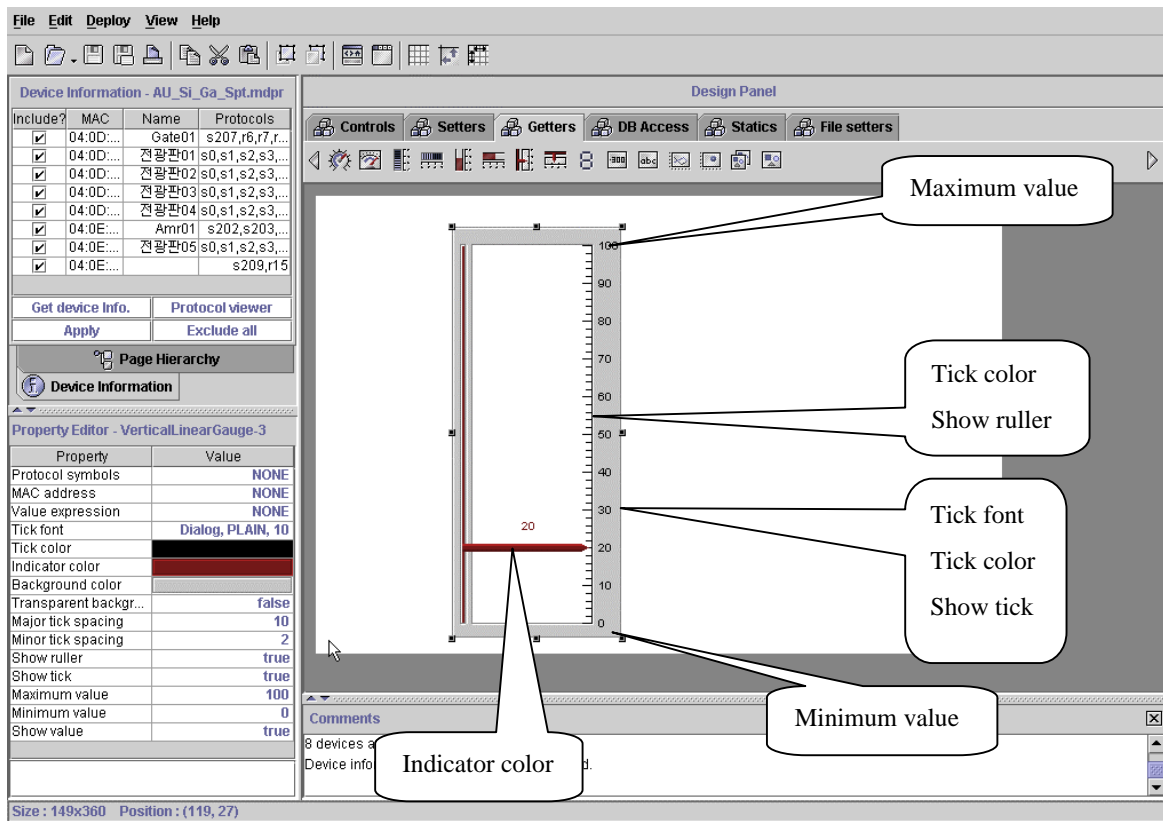
- Value expression : Input the value to be displayed in the protocol. Arithmetic operations are possible for displayed values.
- Transparent background : If True, only Component is seen and background is not displayed.
- Major tick spacing : The interval of large graduations
- Minor tick spacing : The interval of small graduations
- Show ruler : Displays ruler
- Show tick :Displays number
- Maximum value : Maximum value gauge can display
- Minimum value : Minimum value gauge can display
- Show value : If set as True, displays numerical data
- Segment width: the width of each ruler



[Fig 3.26.3 Set VerticalLinearGauge2 Property]

- Value expression : Input the desired variable
(Value type : m0,m1,...).
- Tick font : The font of tick number
- Tick color : Sets Tick color
- Indicator color : Sets Indicator color
- Background color : Sets component background color
- Transparent background : The color of this component background
If True, only the Component is shown.
- Major tick spacing : The interval of large graduations
- Minor tick spacing : The interval of small graduations
- Show ruler : Display ruler
- Show tick :Display number
- Maximum value : Maximum value gauge can display

- Minimum value : Minimum value gauge can display
- Show value : If set as True, displays numerical data



[Fig 3.26.4 Set VerticalLinearGauge3 Property]

- Value expression : Input the receive protocol variable
(Value type : m0,m1,...)..
- Tick font : The font of tick number
- Tick color : Tick color
- Indicator color : Indicator color
- Background color : The component background color
- Transparent background : The color of this component's background
If True, only the Component is shown.
- Major tick spacing : The interval of large graduations
- Minor tick spacing : The interval of small graduations
- Show ruler : Displays ruler
- Show tick : Displays number
- Maximum value : Maximum value that gauge can display

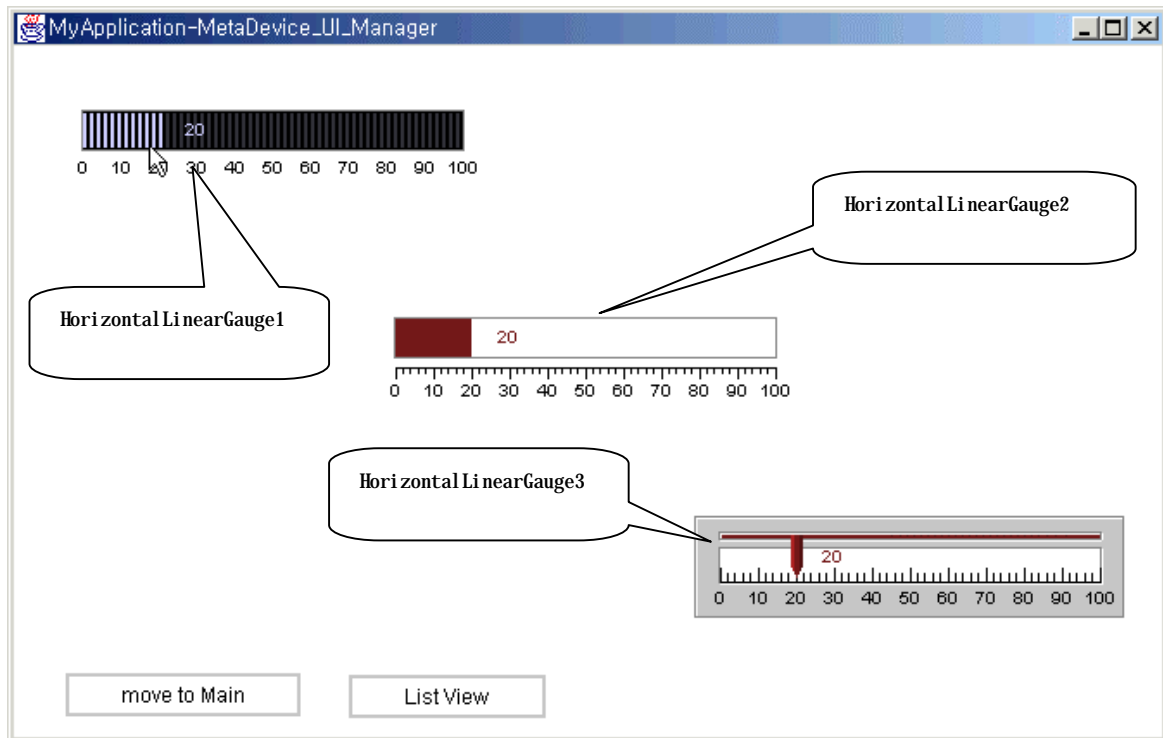
- Show value : If set as True, displays numerical data

6.3 HorizontalLinearGauge1, HorizontalLinearGauge2, HorizontalLinearGauge3

(1) Working at Run Time

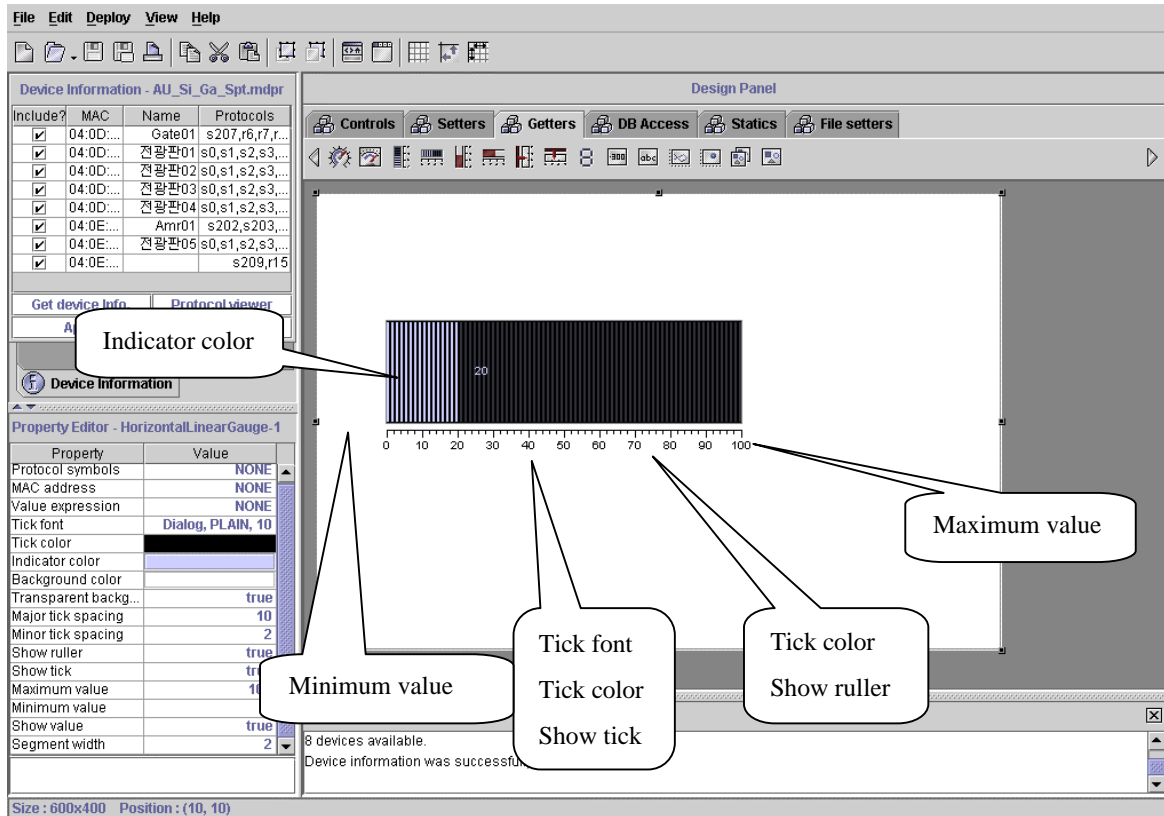
Displays value received from the Server on the horizontal gauge

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..).



[Fig 3.27.1 HorizontalLinearGauge1, HorizontalLinearGauge2, HorizontalLinearGauge3]

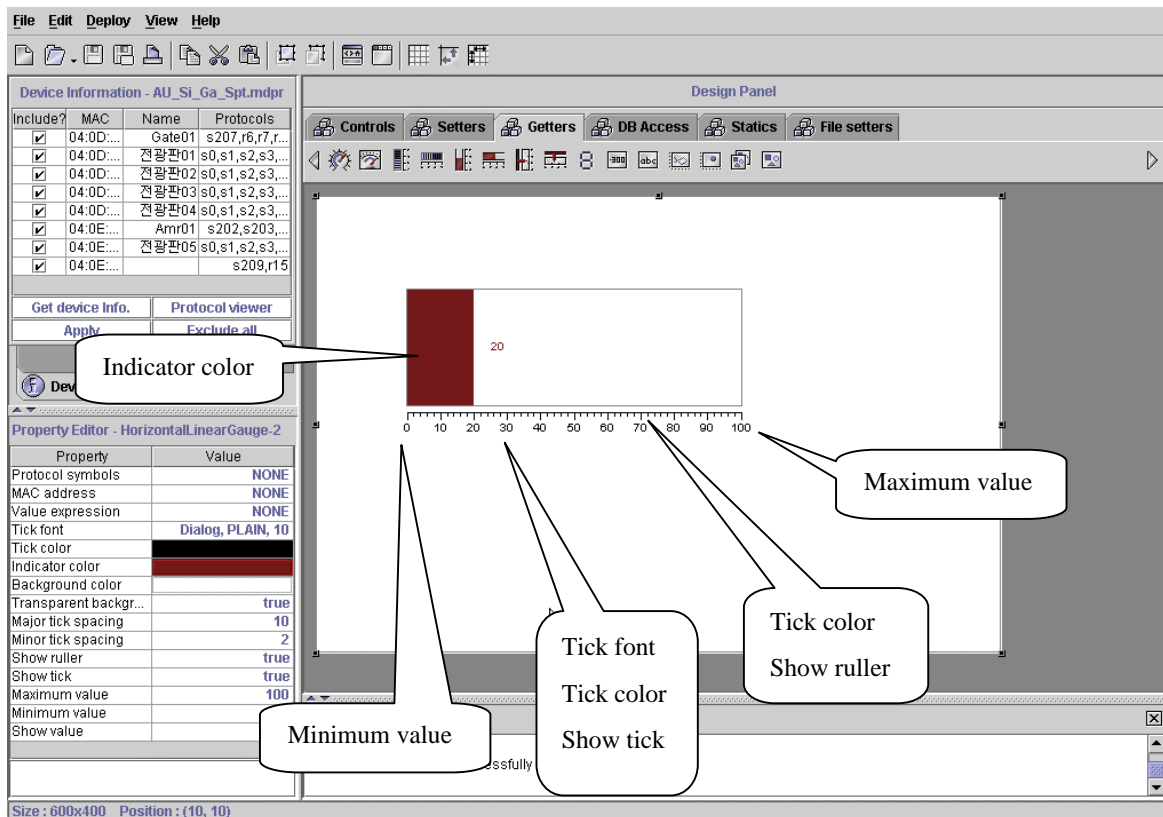
(2) Setting Property



[Fig 3.27.2 Setting HorizontalLinearGauge1 Property]

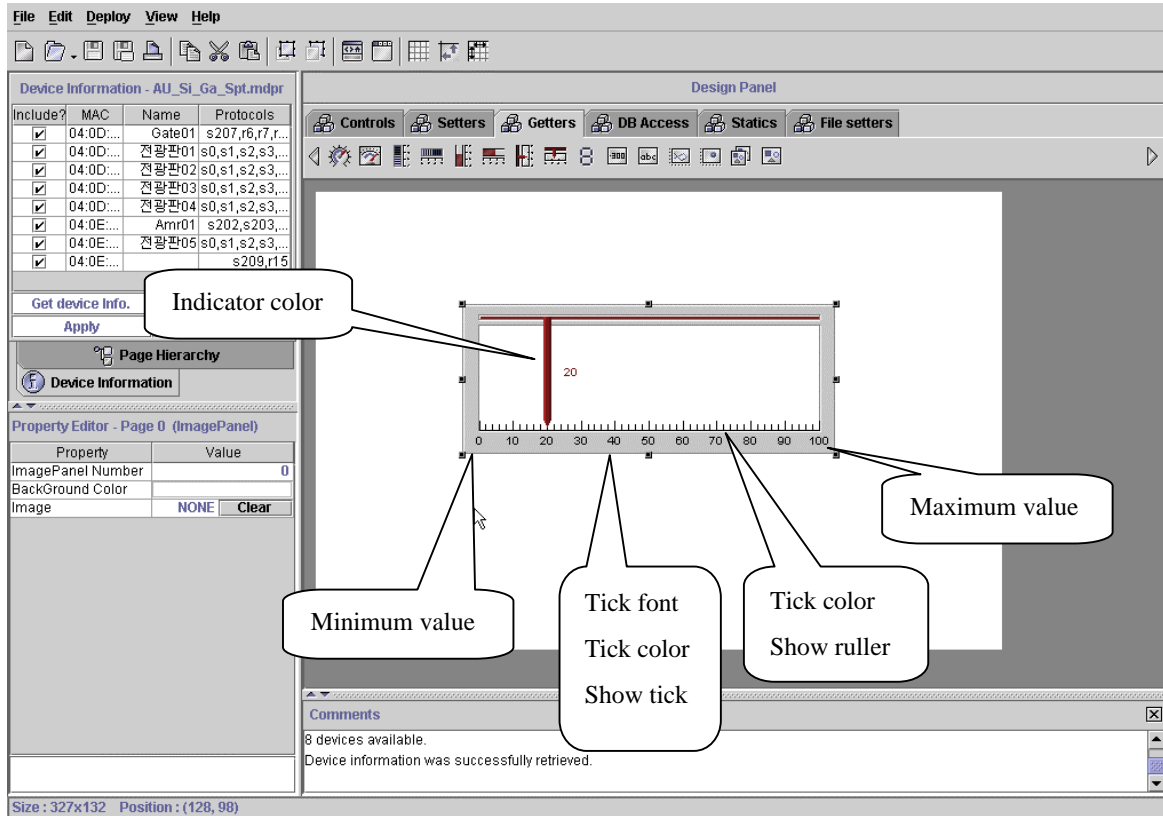
- Value expression : Input the variable of the receive protocol
- Tick font : The font of numbers
- Tick color : The color of numbers, graduations.
- Indicator color : Indicator color
- Background color : Component background color
- Transparent background : The color of component background
If True, only the Component is shown.
- Major tick spacing : the interval of large graduations
- Minor tick spacing : the interval of small graduations
- Show ruler : Displays ruler
- Show tick : Displays number
- Maximum value : Maximum value that gauge can display
- Minimum value : Minimum value that gauge can display

- Show value : If set as True, displays numerical data.
- Segment width: The width of one segment



[Fig3.27.3 Setting HorizontalLinearGauge2 Property]

- Value expression : Input the variable of the receive protocol
- Tick font : The font of numbers.
- Tick color : The color of numbers, graduations.
- Indicator color : Indicator color
- Background color : Component background color
- Transparent background : The color of this component's background
If True, Component is shown only.
- Major tick spacing : the interval of large graduations
- Minor tick spacing : the interval of small graduations
- Show ruler : Displays ruler
- Show tick : Displays number
- Maximum value : Maximum value that gauge can display
- Minimum value : Minimum value that gauge can display
- Show value : If set as True, displays numerical data



[Fig 3.27.4 Setting HorizontalLinearGauge2 Property]

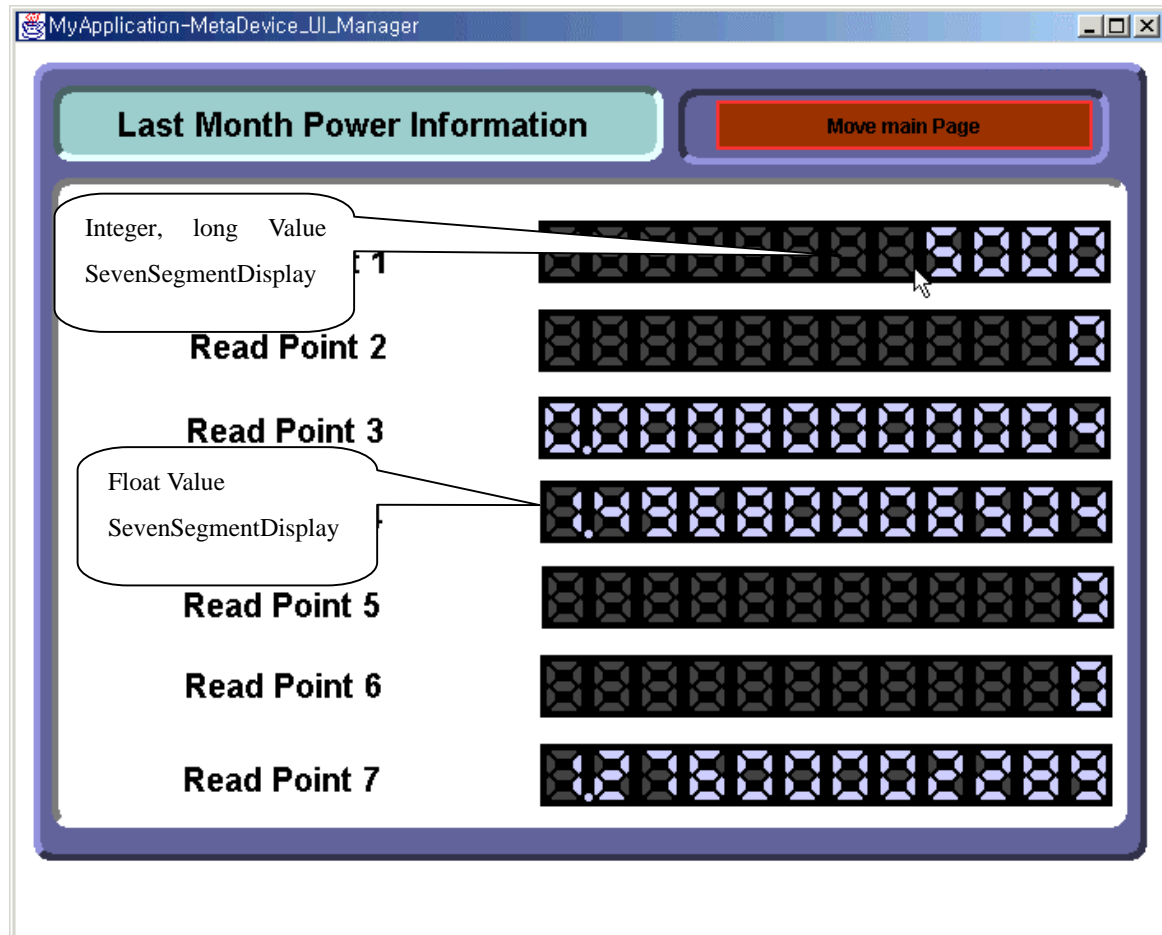
- Value expression : Input the variable of the receive protocol
- Tick font : The font of number
- Tick color : The color of number.
- Indicator color : Indicator color
- Background color : Component background color
- Transparent background : The color of this component's background
If True, only the Component is shown.
- Major tick spacing : the interval of large graduations
- Minor tick spacing : the interval of small graduations
- Show ruler : Displays ruler
- Show tick :Displays number
- Maximum value : Maximum value that gauge can display
- Minimum value :Minimum value that gauge can display
- Show value : If set as True, displays numerical data

6.4 SevenSegmentDisplay

(1) Working at Run Time

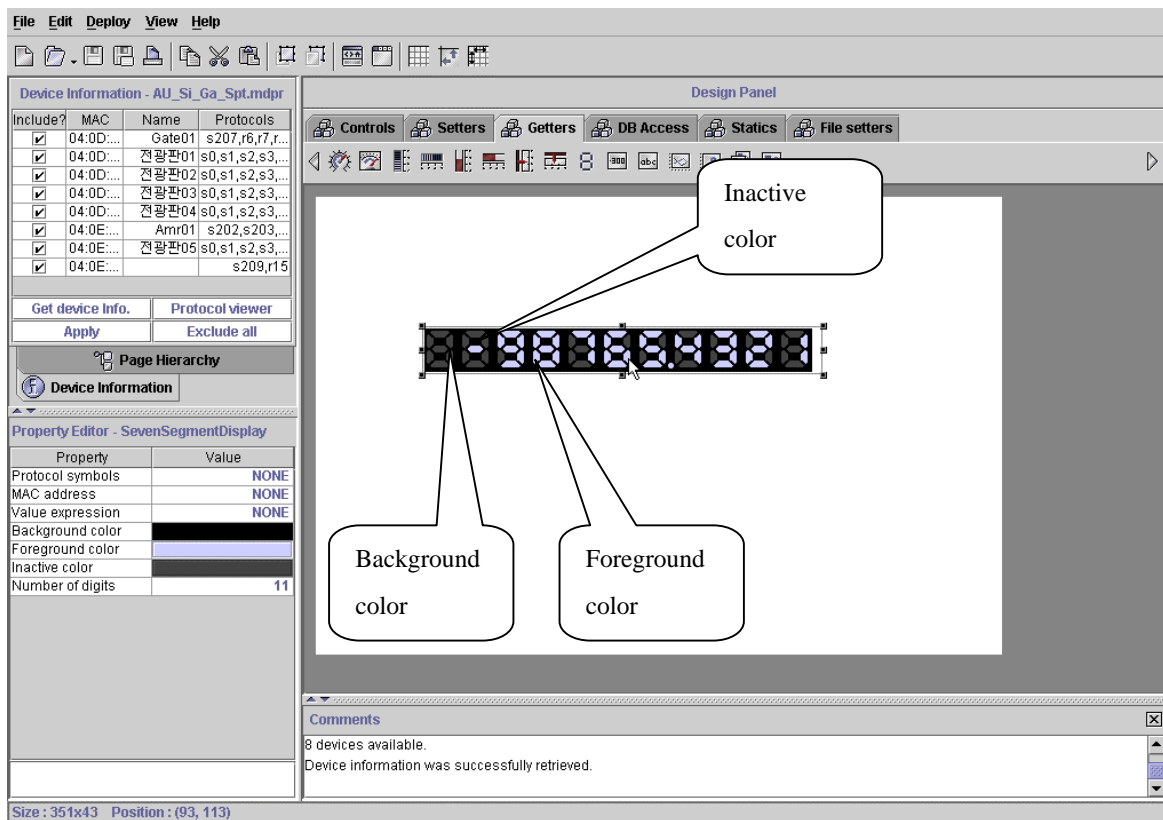
Displays value received from the Server on the Seven Segment Displayer

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..).



[Fig 3.28.1 SevenSegmentDisplay]

(2) Setting Property at Design Time



[Fig 3.28.2 Setting SevenSegmentDisplay Property]

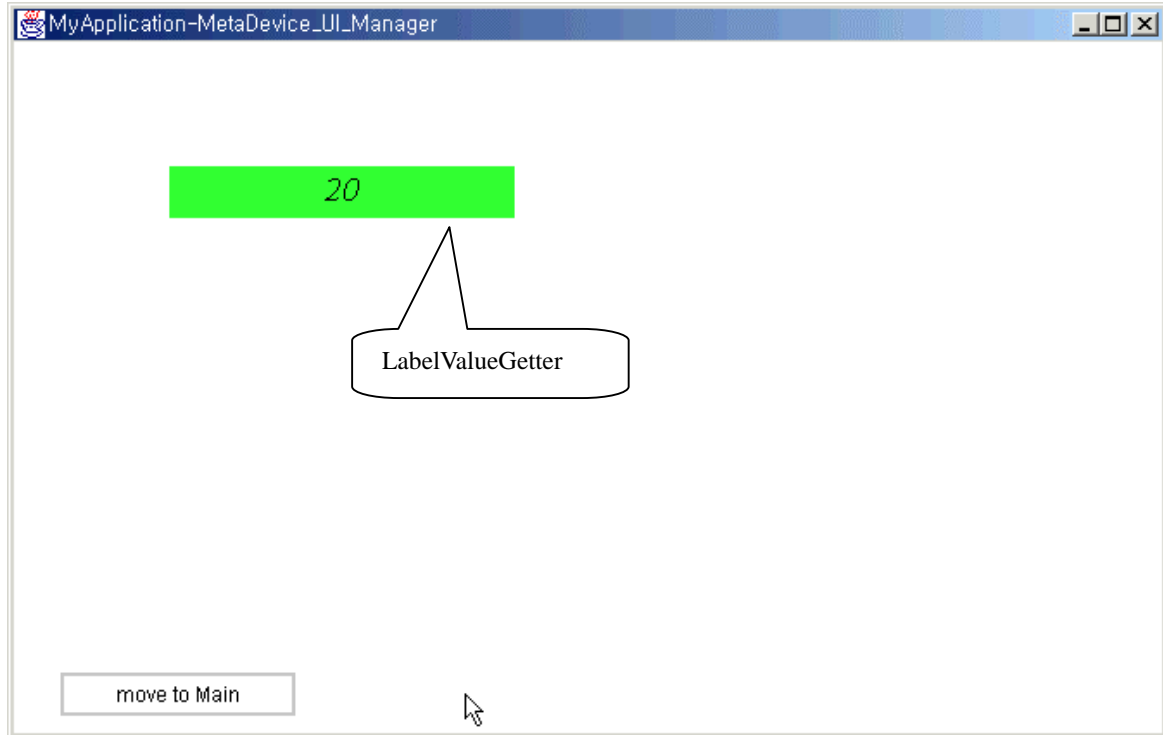
- Value expression : Input the variable of receive protocol.
- Background color : Component background color
- Foreground color : Value display color
- Inactive color : Inactive color
- Number of digits : The displayer size of number

6.5 LabelValueGetter

(1) Working at Run Time

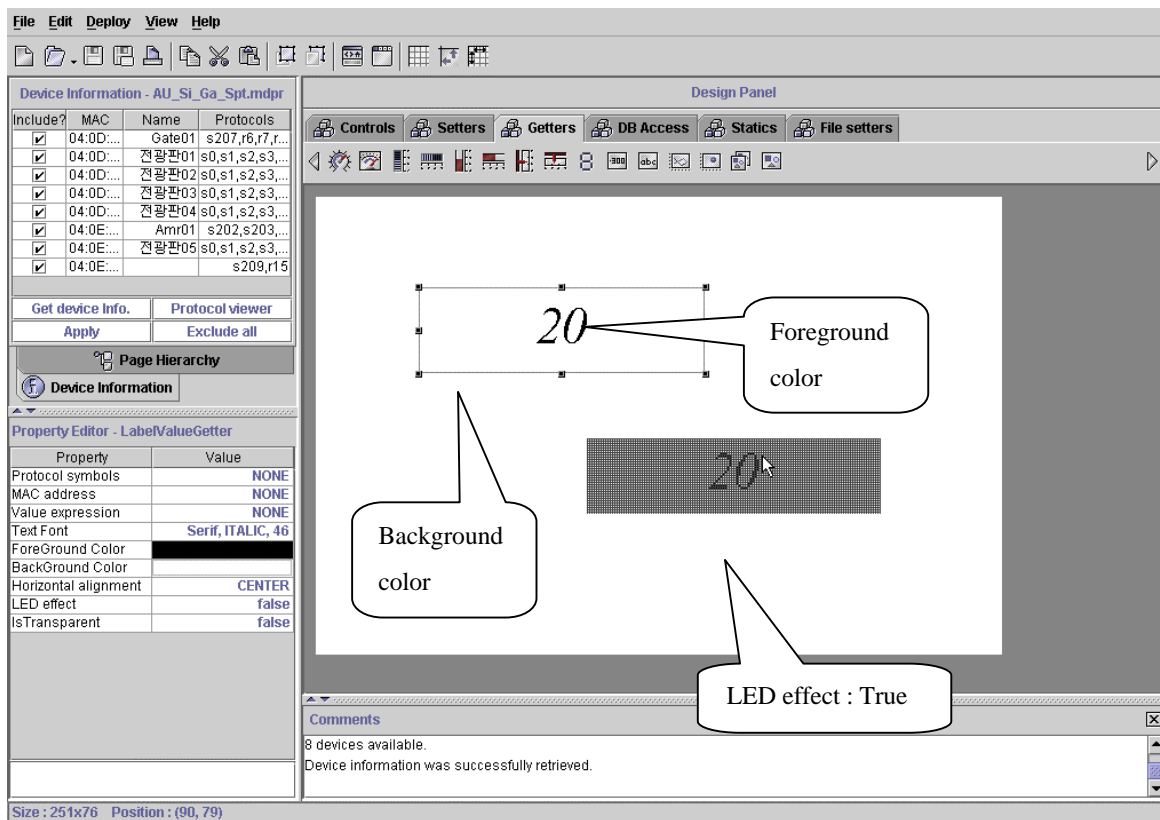
Displays value received from the Server on the label.

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..) ..



[Fig 3.29.1 LabelValueGetter]

(2) Setting Property at Design Time



[Fig 3.29.2 Setting LabelValueGetter Property]

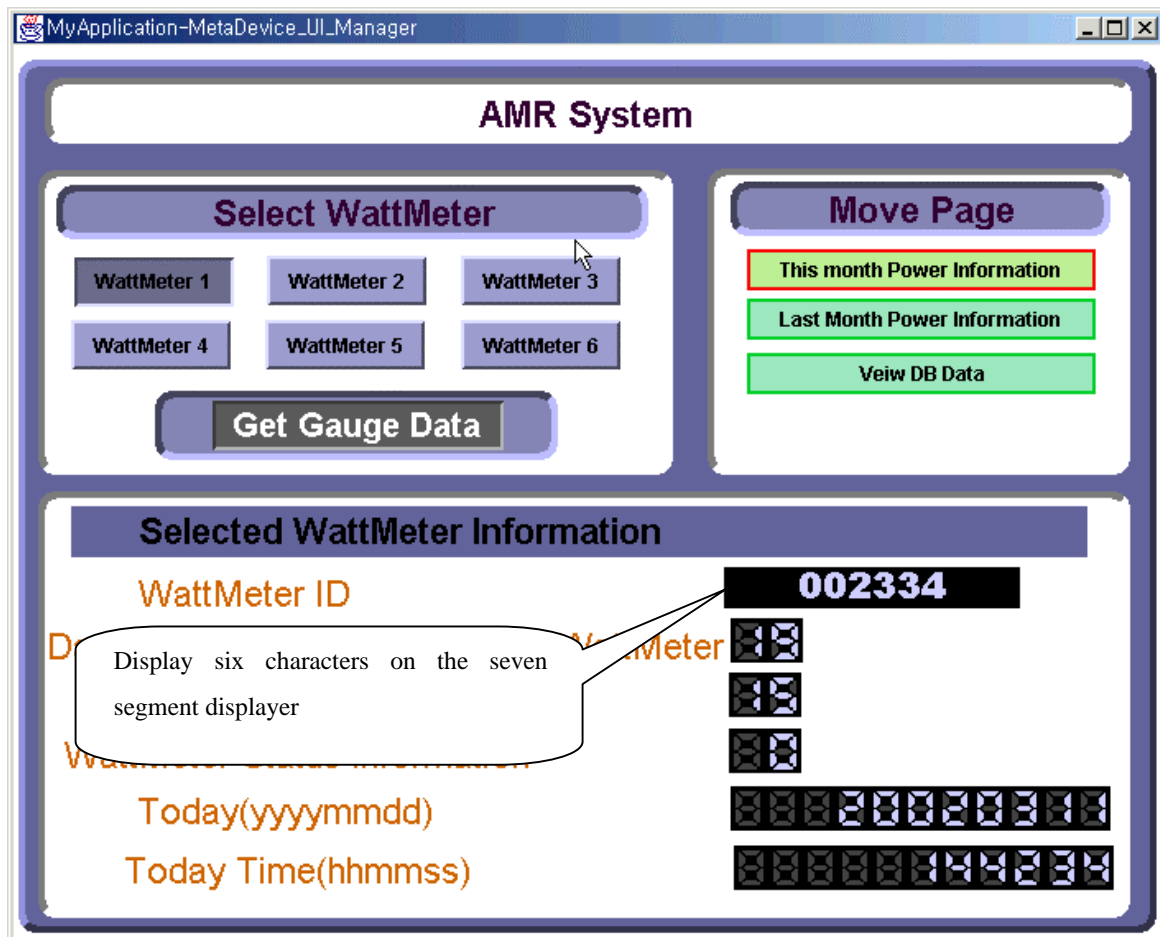
- Value expression : Input the receive protocol variable.
- ForeGround Color : The color of the number to be displayed
- BackGround Color : Component background color
- Horizontal alignment : The position of the number (Left, Center, Right)
- LED effect : If True, the component background is displayed like LED
- IsTransparent : If True , the background of component is not displayed.

6.6 TextLabelValueGetter

(1) Working at Run Time

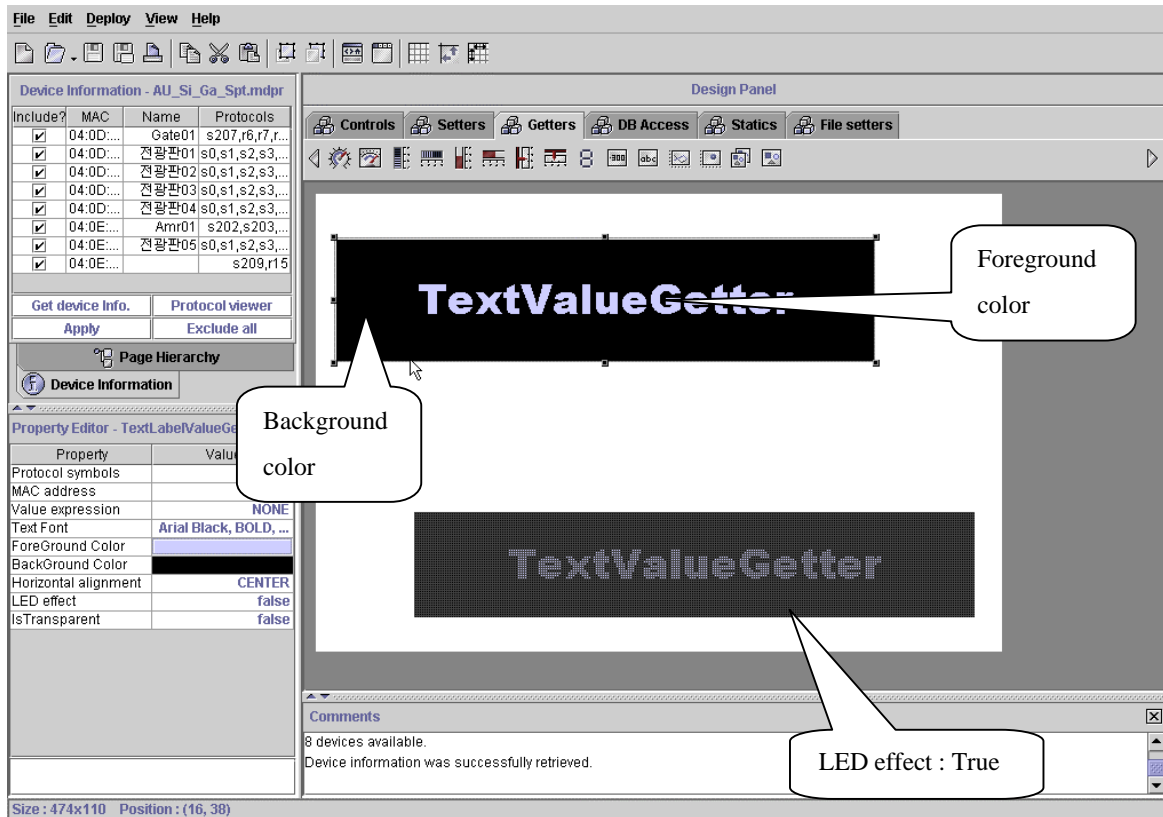
Displays text value received from the Server on the label.

The text value is Receive Protocol(r0, r1, r3..) text variable (v0, v1, v2...)...



[Fig 3.30.1 TextLabelValueGetter]

(2) Setting Property at Design Time



[Fig 3.30.2 Setting TextLabelValueGetter Property]

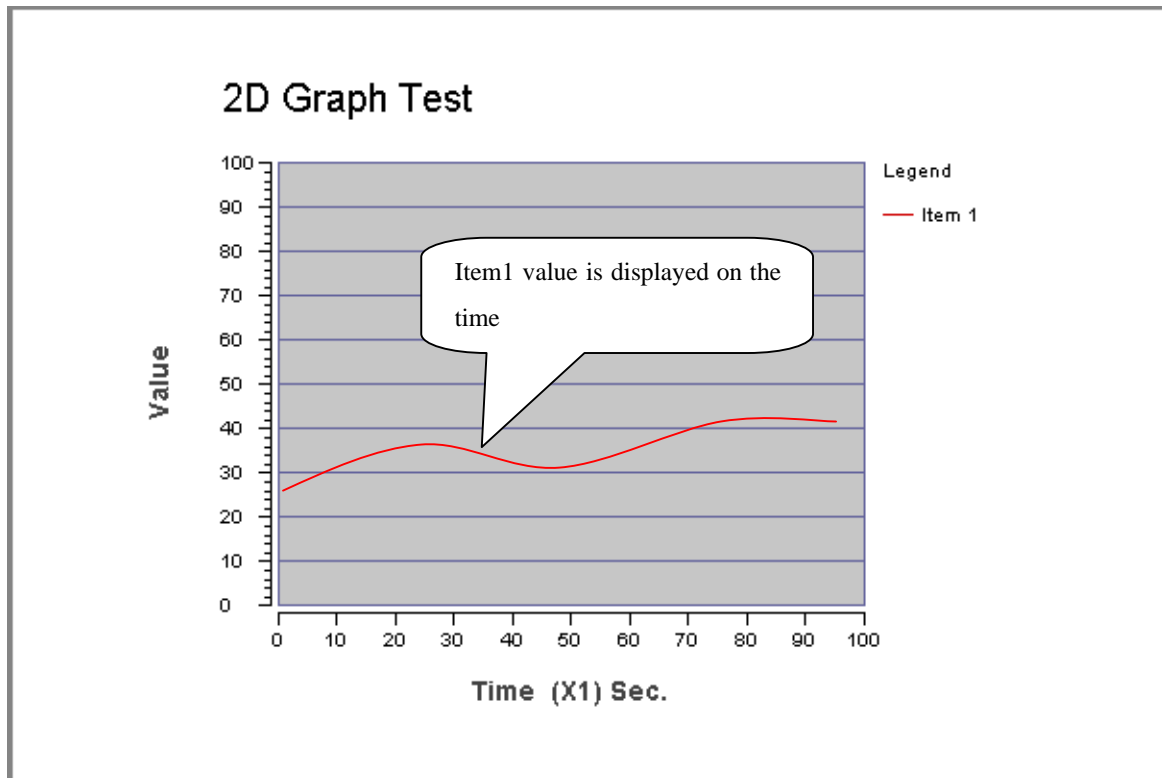
- Value expression : Input the text variable
- Text Font : Text font to be displayed
- ForeGround Color : The color of text
- BackGround Color : Component background color
- Horizontal alignment : The position of text value(Left, Center, Right)
- LED effect : If True, the component background is displayed like LED

6.7 Graph2D

(1) Working at Run Time

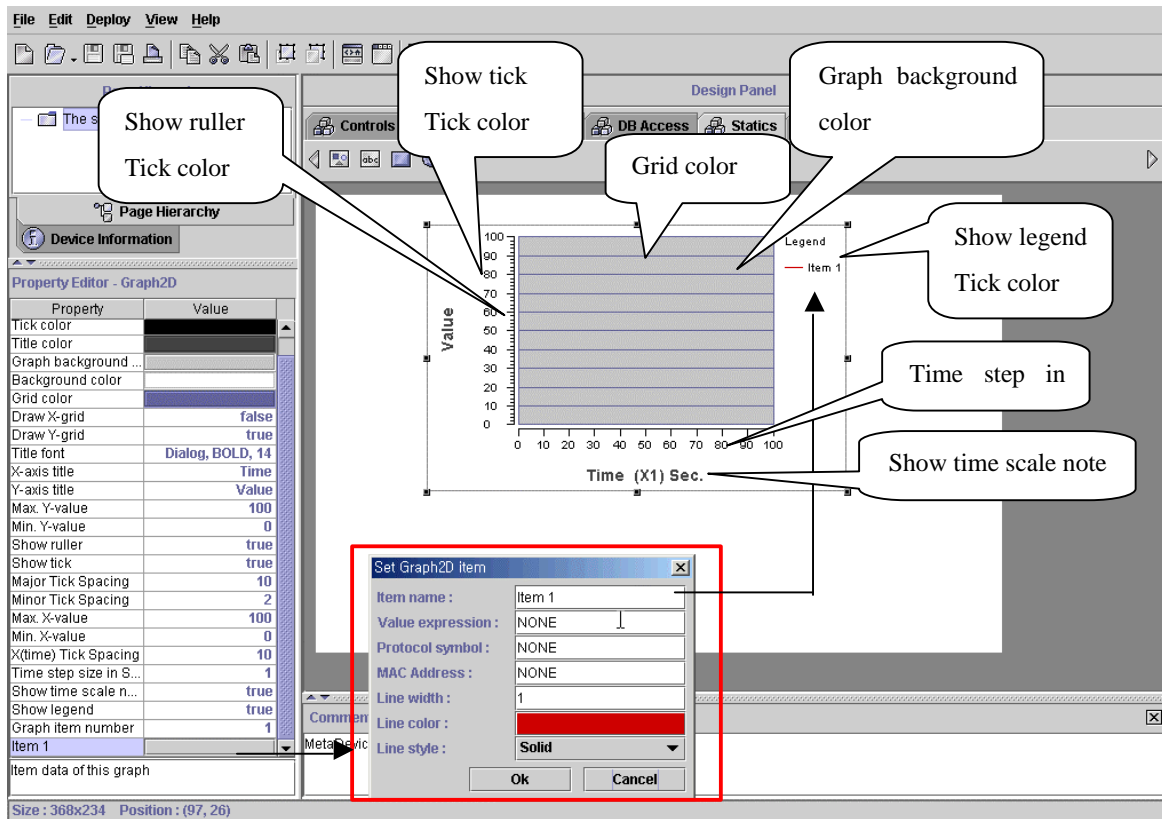
Displays value received from the Server on the 2D Graph.

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..) ..



[Fig 3.31.1 Graph2D]

(2) Setting Property at Design Time



[Fig 3.31.2 Setting Graph2D Property]

- Background color : Component background color
- Draw X grid : If True, display grid with X Major tick spacing
- Draw Y grid : If True, display grid with Y Major tick spacing
- X-axis title : X Axis Title(Text)
- Y-axis title : Y Axis Title(Text)
- Max Y value : Maximum value of Y axis
- Min Y value : Minimum value of Y axis
- Major tick spacing : the interval of large graduations
- Miner tick spacing : the interval of small graduations
- Max X value : X axis Maximum value
- Min X value : X axis Minimum value
- Time step size in sec : The interval of time
- Graph item number : The number of items.
- item1 : The first item set by the Graph item number

Click the item field to edit the item property

< item property >

- item name : item name
- Value expression : Input the receive protocol value
- Protocol symbols : Input the receive protocol symbol.
- MAC address : Input the Device Mac Number.

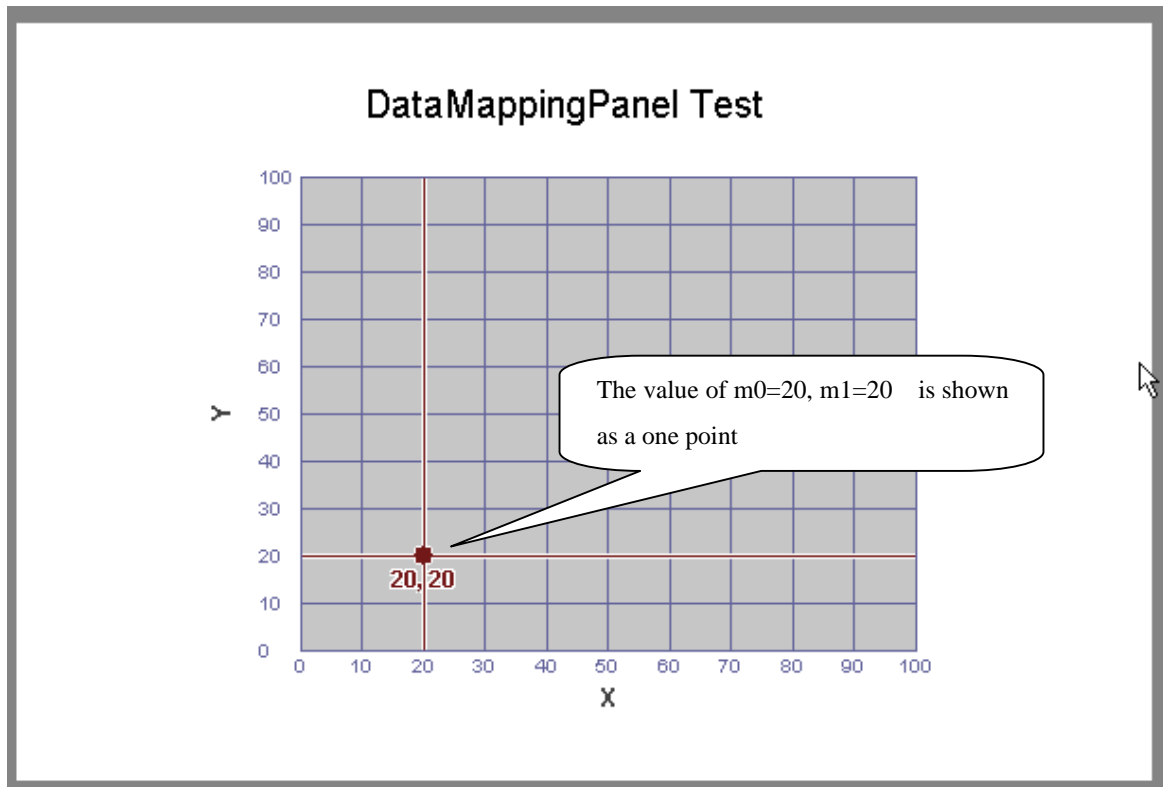
If NONE , DeviceSelector Mac number is applied to this component

- Line width : The line width of value displayer
- Line color : Line color
- Line Style : Dashed, Solid

6.8 DataMappingPanel

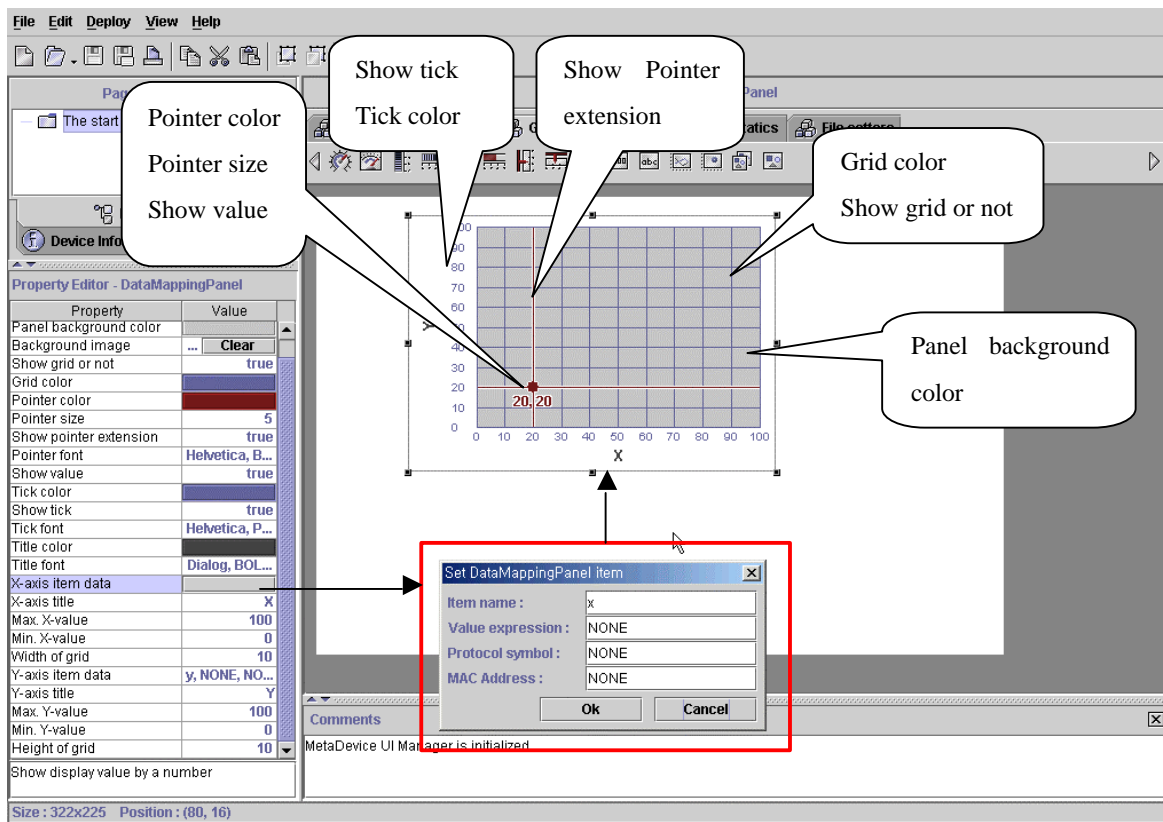
(1) Working at Run Time

Two values are displayed in the X position and Y position of the DataMappingPanel



[Fig 3.32.1 DataMappingPanel Working]

(2) Setting Property at Design Time



[Fig 3.32.2 Setting DataMappingPanel Property]

- Background Image : Component background image
- Show grid or not : If True , grid line is shown
- Pointer size : The value pointer size
- Show pointer extension : If True, the extension line is shown .
- Show value : If True, the value of pointer is shown
- X-axis item data: Input the variable to be shown on the X coordination
- X-axis Title : X axis title name
- Max.X value : X axis maximum value
- Min.X value : X axis minimum value
- Width of grid : The grid graduations
- Y-axis Title : Y axis title name
- Max.Y value : Y axis maximum value
- Min.Y value : Y axis minimum value
- Height of grid : The grid vertical graduations

< set DataMappingPanel axis data item >

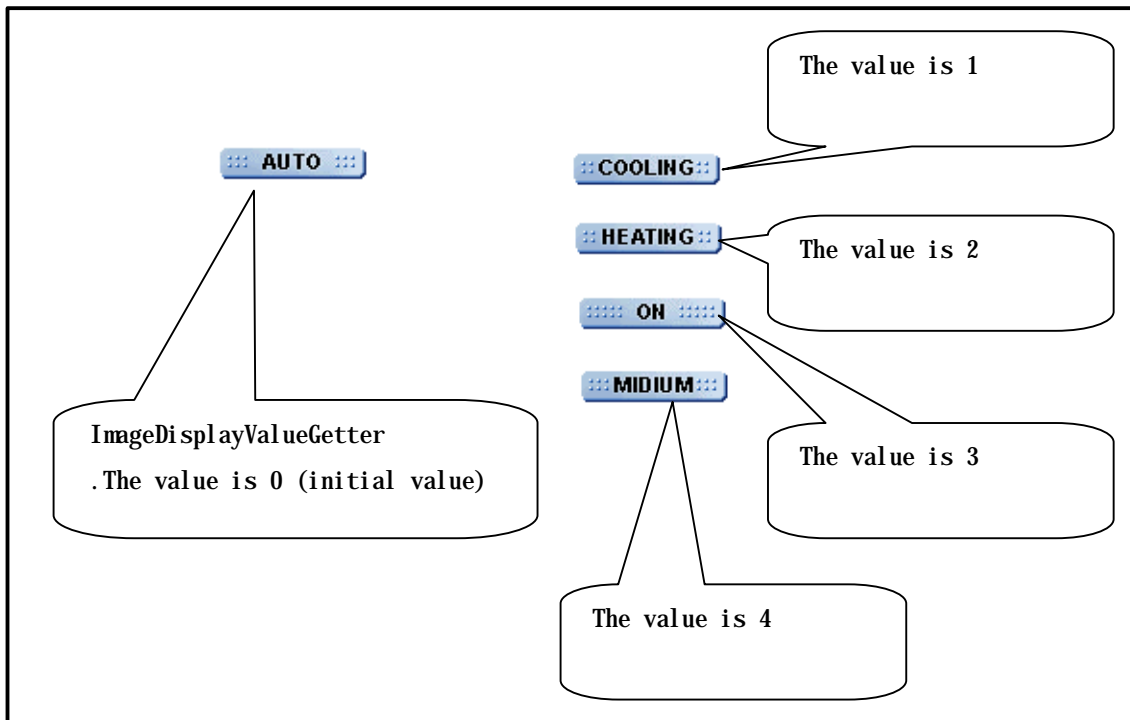
- Item name : Item name
- value expression: Input the value which will be shown on the X(Y) Coordination.
- Protocol symbol : Input Protocol Symbol.

6.9 ImageDisplayValueGetter

(1) Working at Run Time

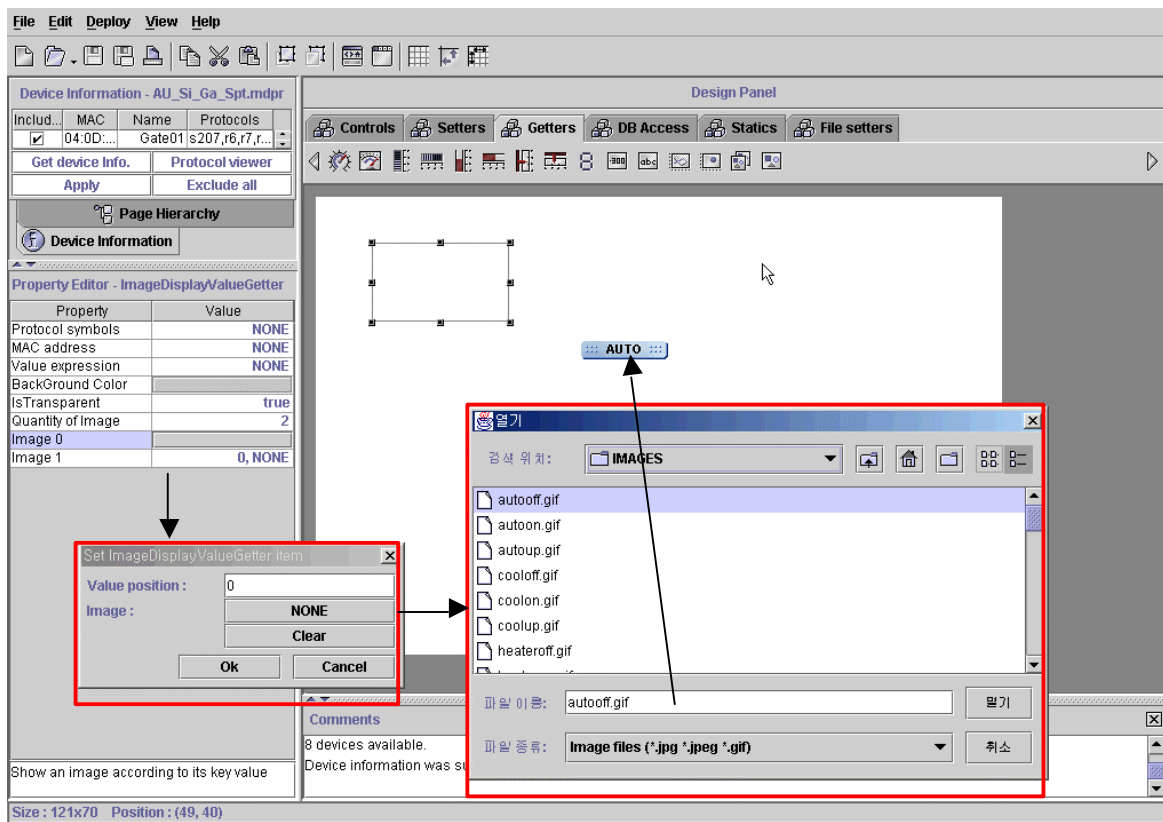
Displays value received from the Server with image

The value is Receive Protocol(r0, r1, r3..) variable (m0, m1, m2..) ..



[Fig 3.33.1 ImageDisplayValueGetter]

(2) Setting Property at Design Time



[Fig 3.33.2 Set ImageDisplayValueGetter Property]

- Value expression : Input the variable of receive protocol.
- Background color : Component background color
- IsTransparent : If True, the background is transparent
- Quantity of image : The number of image to be displayed
- Image 0 : Click to set ImageDisplayValueGetter value and image

Input the actual value in the Value Position.

Click NONE to select image

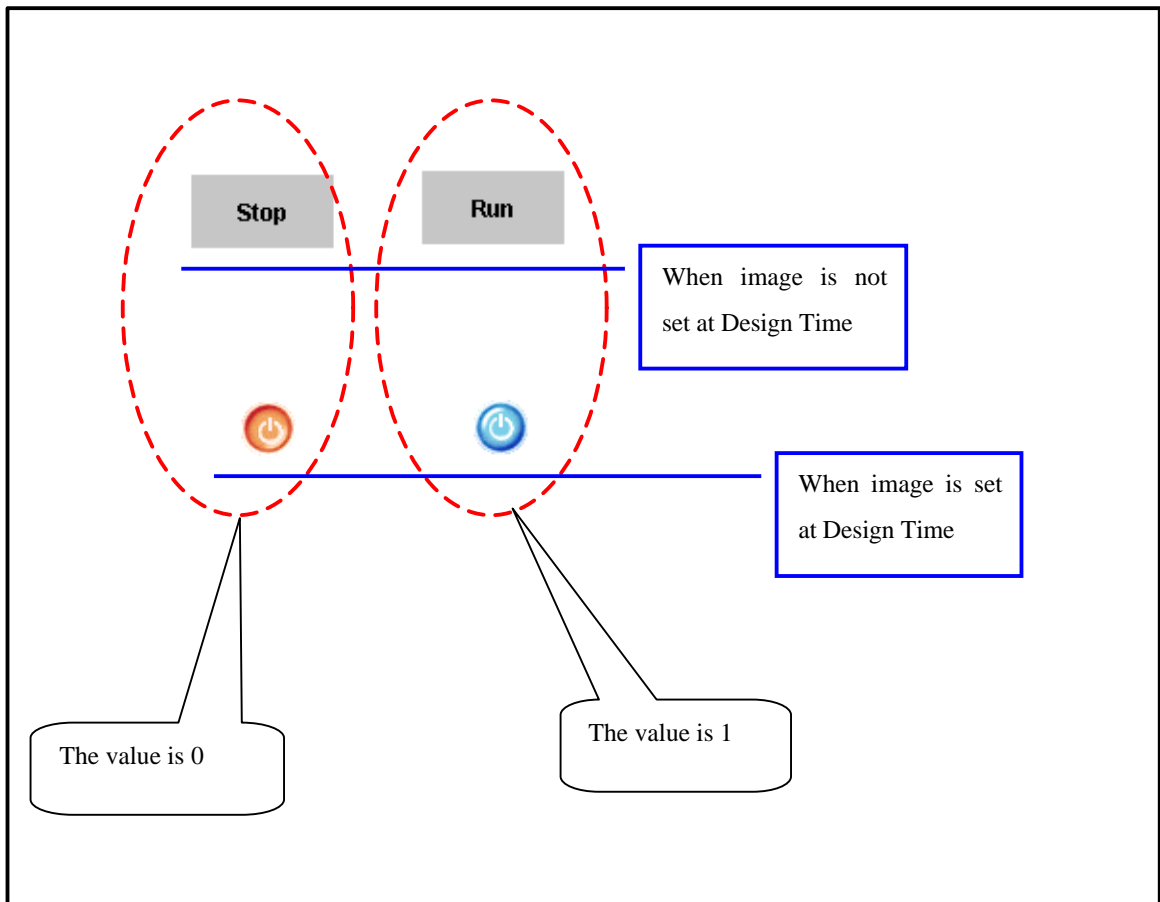
If the variable value of Value expression equals Value Position, the selected image is shown

6.10 ImageOnOffGauge

(1) Working at Run Time

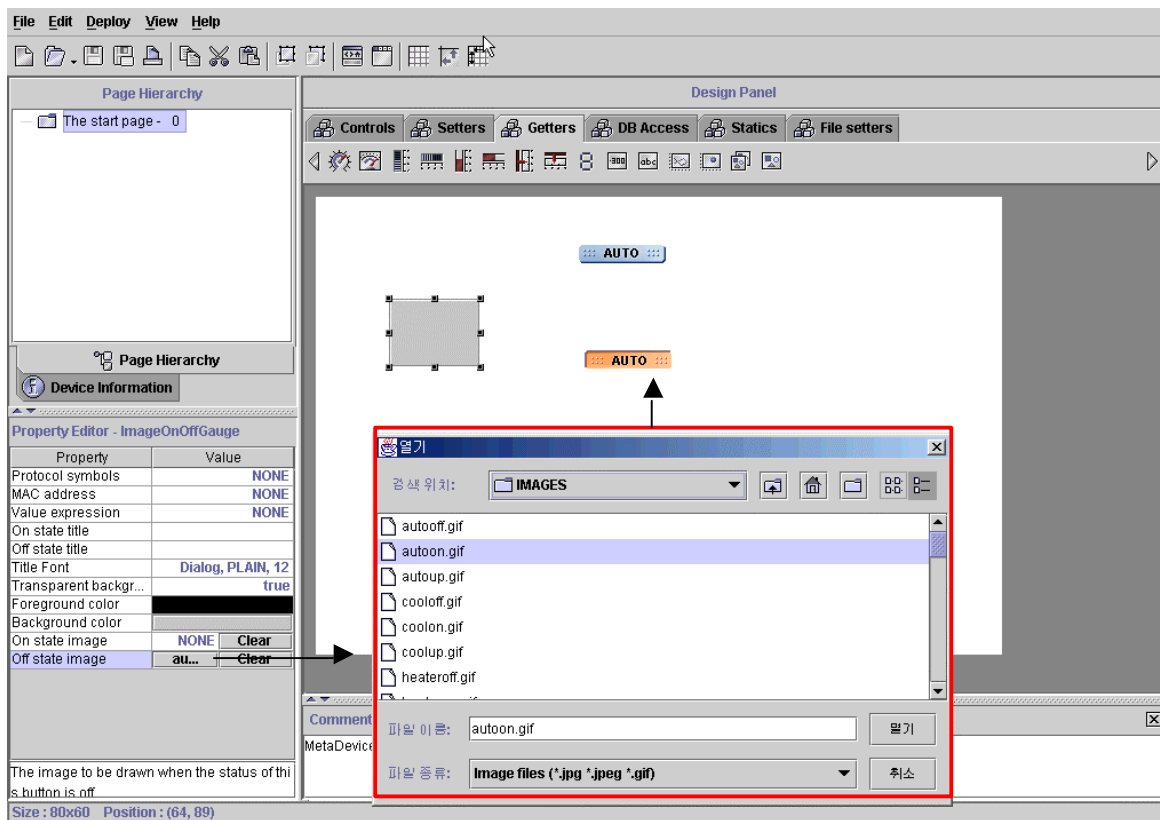
Displays image differently depending on whether the Receive data is 0 or 1.

User can change and set the toggle data



[Fig 3.34.1 ImageOnOffGauge]

(2) SettingProperty at Design Time



[Fig 3.34.2 Set ImageOnOffGauge Property]

- Value expression : Input the variable of receive protocol.
- On state title : If the value is 1, this title will be shown
- Off state title : If the value is 0, this title will be shown
- Foreground color : Text color
- Background color : Background color
- On state image : If the value is 1, this image will be shown
- Off state image : If the value is 0, this image will be shown

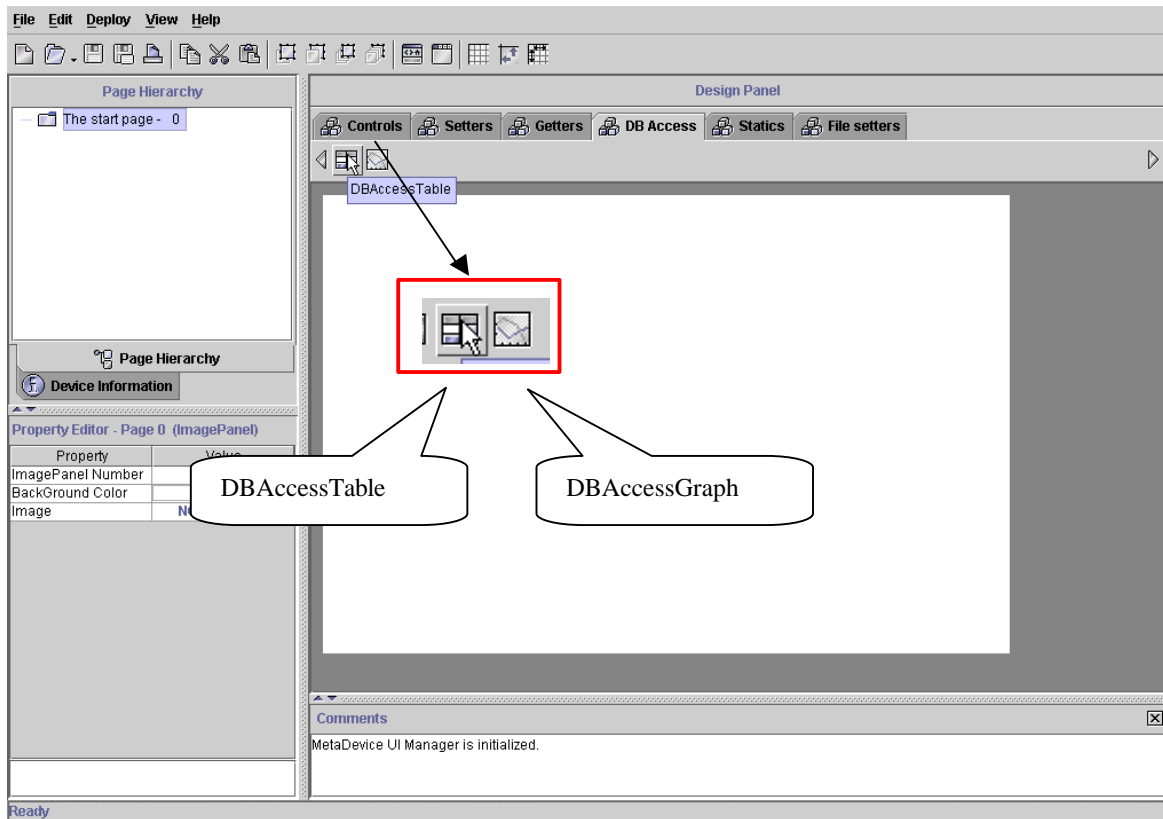
7. DB Access Component Group

(1) What is DB Access Component ?

The DB Access Component displays protocol data in the Database

(2) DB Access Component

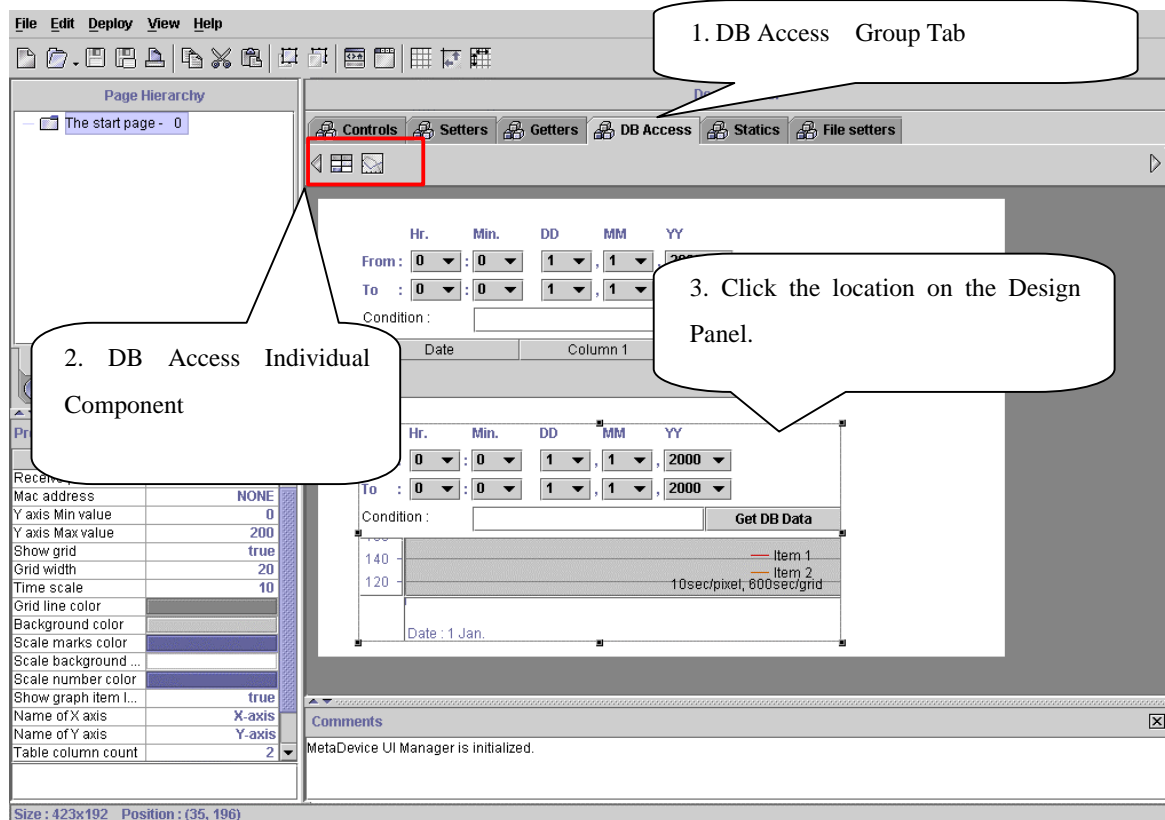
- DBAccessTable : The data is displayed as spreadsheet type
- DBAccess Graph : The data is displayed as graph type



[Fig 3.35.1 DB Access Component]

(3) DB Access Component Editing

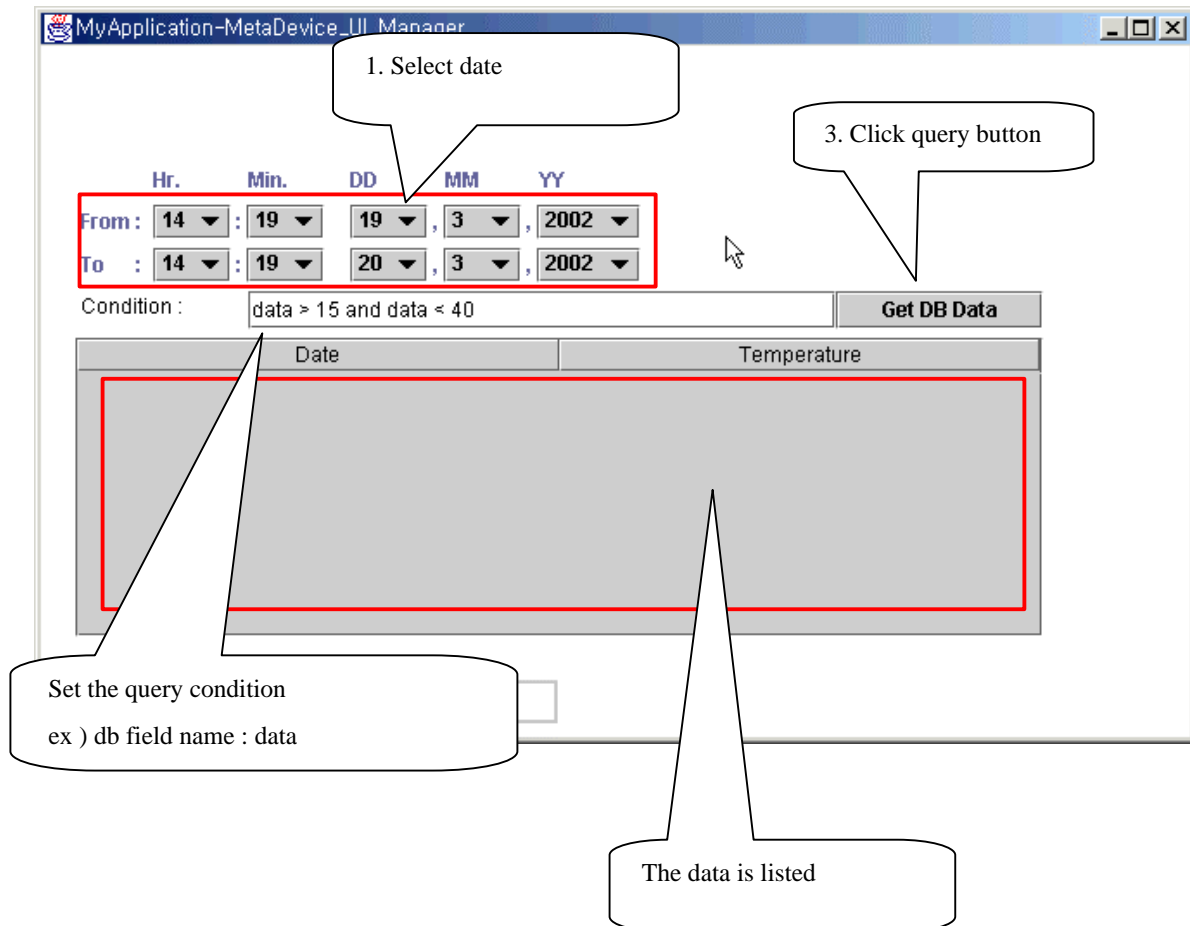
- Click the DB Access Group Tab from the Design Panel.
- Select the DB Access Group individual component and click the position where the component should be located.
- Edit Property.

**[Fig 3.35.2 DB Access Component Editing]**

7.1 DB Access Table

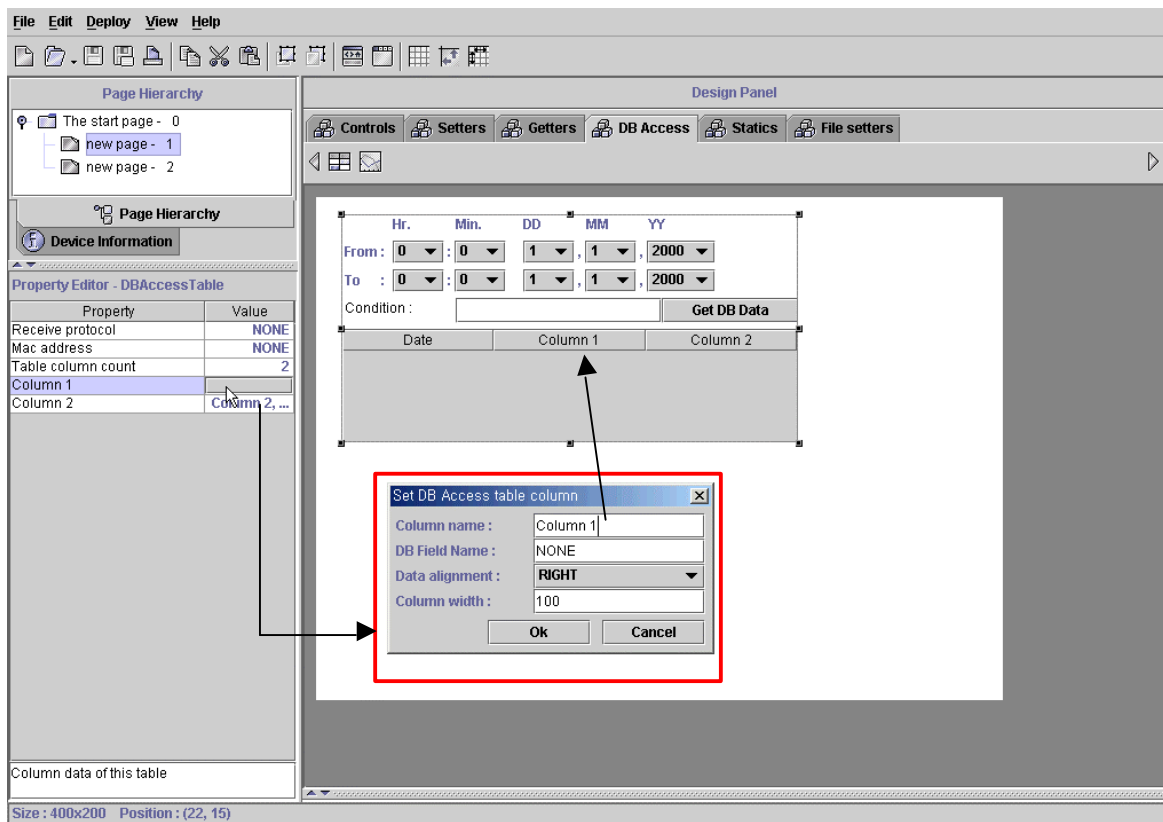
(1) Working at Run Time

Receives data from the database Server and displays the values on the Table.



[Fig 3.36.1 DB Access Table]

(2) Setting Property at Design Time



[Fig 3.36.2 Set DB Access Table Property]

- Receive Protocol : Input Protocol Name (table name) to be read from database.
- Mac address : Input Device Mac Number to be read from DataBase. If nothing is input(None), DeviceSelector Mac will be applied when running Applet.
- Table column count : Sets the number of fields to be read from the DataBase
Equal number of columns will be created.
- Column N : Select to open the Set DB Access table column window and edit.

< Set DB Access table column >

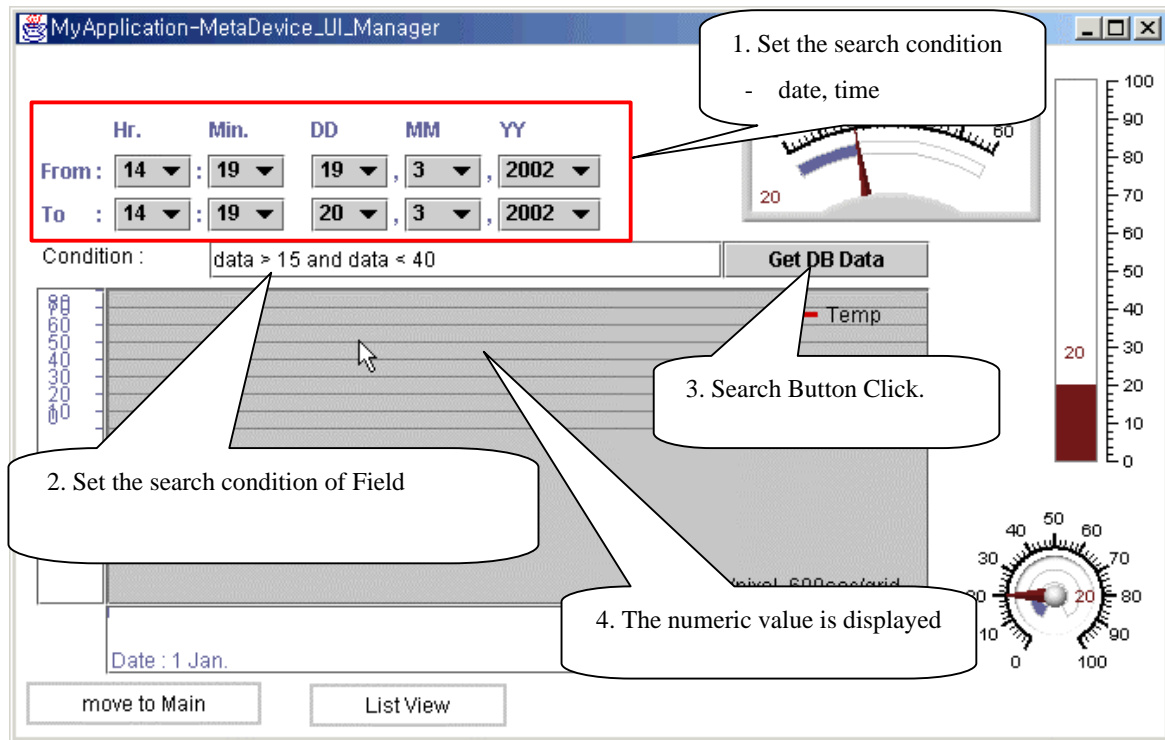
- Column name : Sets Column name which will be shown in the display table.
- DB field name : Actual DataBase Field Name.
- Data Alignment : Sets the position of Column(Right, Center, Left)
- Column width : Column width.

7.2. DB Access Graph

(1) Working at Run Time

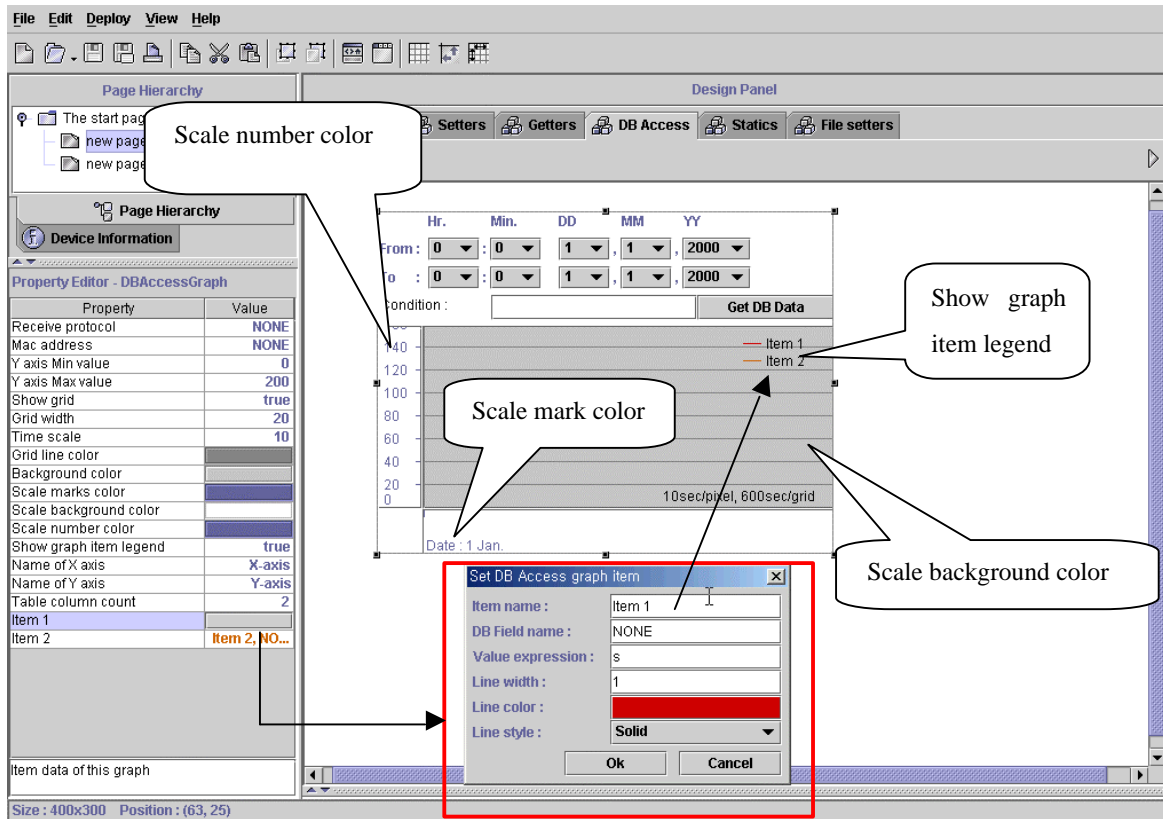
Receives data from the database Server and displays the value on the Graph

The value is numeric.



[Fig 3.37.1 DB Access Graph]

(2) Setting the Property at Design Time



[Fig 3.37.2 Setting DB Access Graph Property]

- Receive protocol : Input Protocol Name (Table Name) to be read from database.
- Mac address: Input the Device Mac Number to be read from database.
- Y Axis Min value : Sets Y axis minimum value
- Y Axis Max value : Sets Y axis maximum value
- Show grid : If True , the grid line is shown.
- Grid width : Sets Grid width
- Time scale : The time interval.
- Grid line color : Sets Grid line color
- Background color : Sets Component background color
- Scale mark color : Sets scale mark color
- Scale background color : Sets scale background color
- Scale number color : Sets scale number color
- Show graph item legend : If True, item legend is shown.

- Name of X axis : Sets X axis text title.
- Name of Y axis: Sets Y axis text title
- Table column count: The number of fields to be displayed
- item N : click to open Set DB Access Graph item window and set value of each column.
 - < Set DB Access graph item >
 - item name : Sets the name of item in the component
 - DB field name: DataBase Field Name
 - value expression : The value read from the DataBase will be saved as variable S.
 - The graph will show the value as S
 - If value expression is S and S is 10, the graph will display 10
 - If value expression is $S*10$ and S is 10, the graph will display 100
- Line width : Sets Graph line width
- Line color : Sets Graph line color
- Line style : Dashed or solid

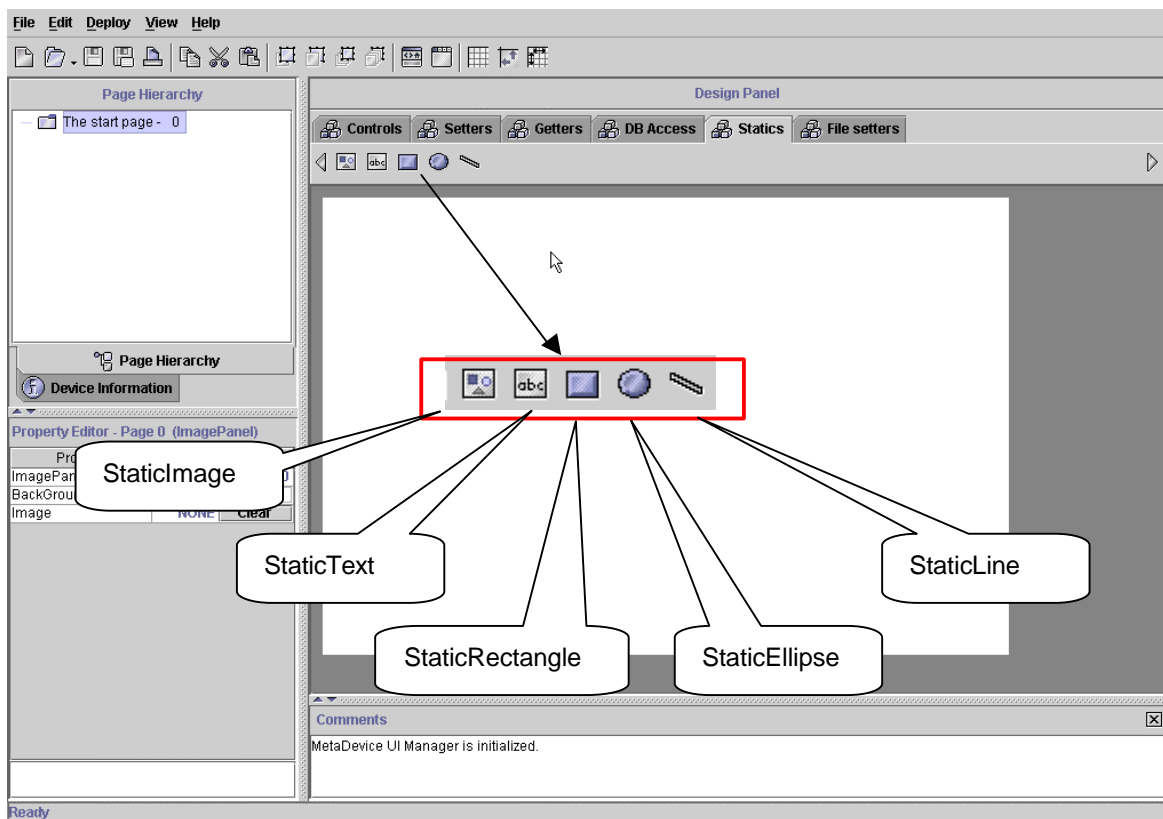
8 Static Component Group

(1) What is **Static Component**?

Static Component does not vary according to data and it is used for UI decoration

(2) **Static Components**

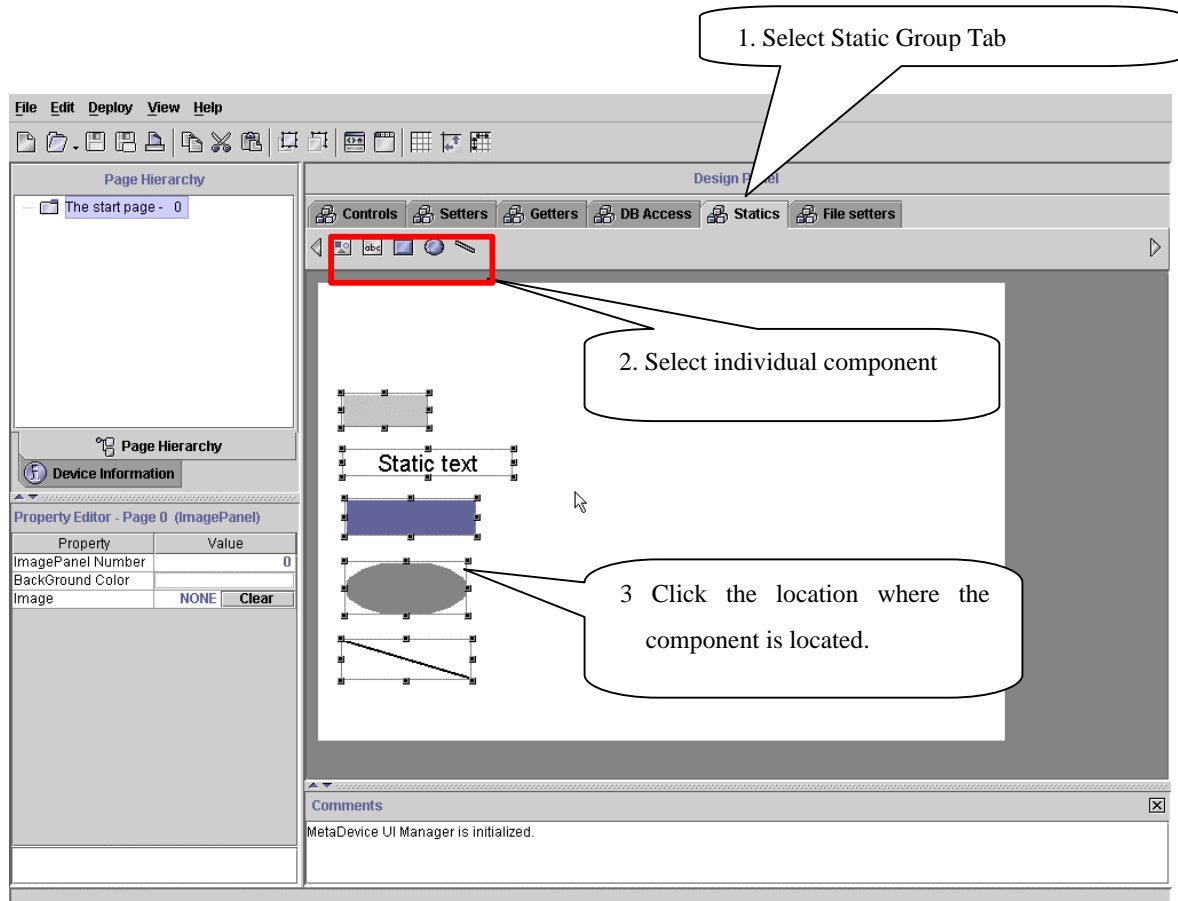
- StaticImage : Displays image
- StaticText : Displays static text defined by user.
- StaticRectangle : Draws a rectangle
- StaticEllipse : Draws an ellipse.
- StaticLine : Draws a line.



[Fig 3.38.1 Static Components]

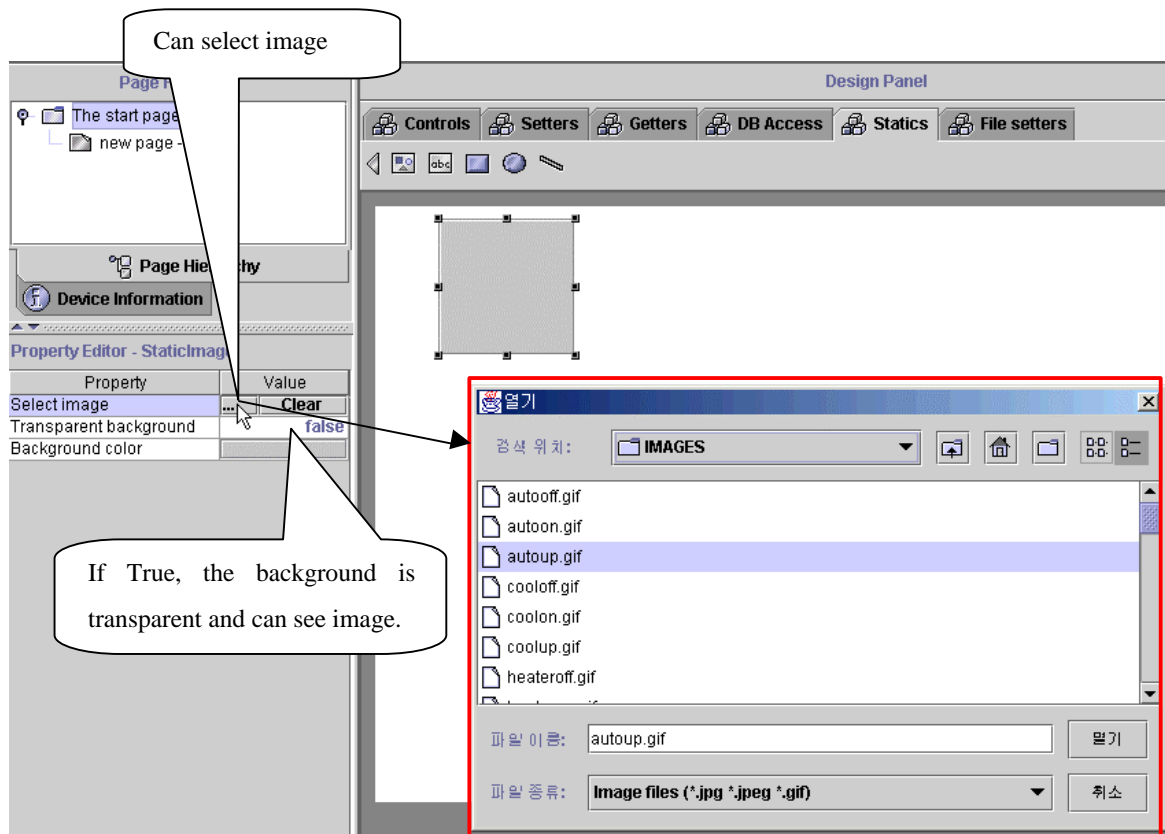
(3) Static Component Editing

- Select Static Group Tab
- Select individual component and click the location where the component should be located.
- Set property.



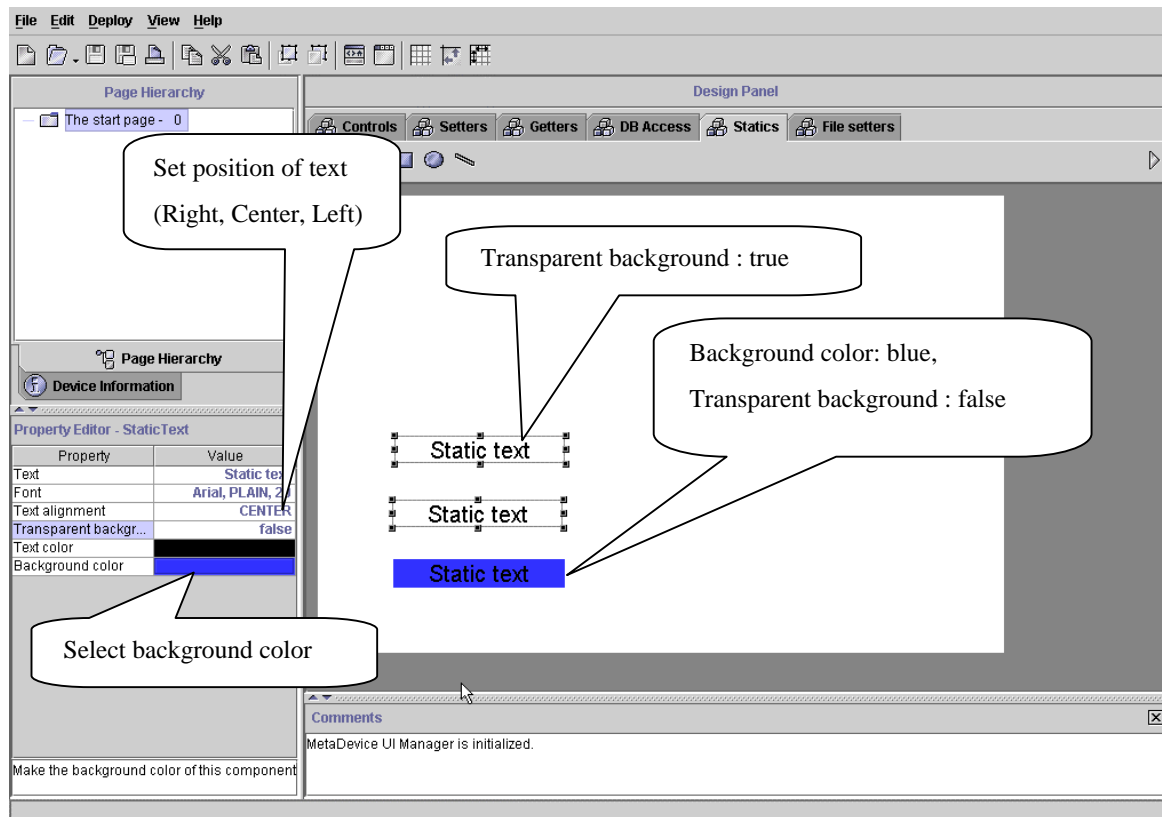
[Fig 3.38.2 Static Component Editing]

8.1 Setting Static Image Property



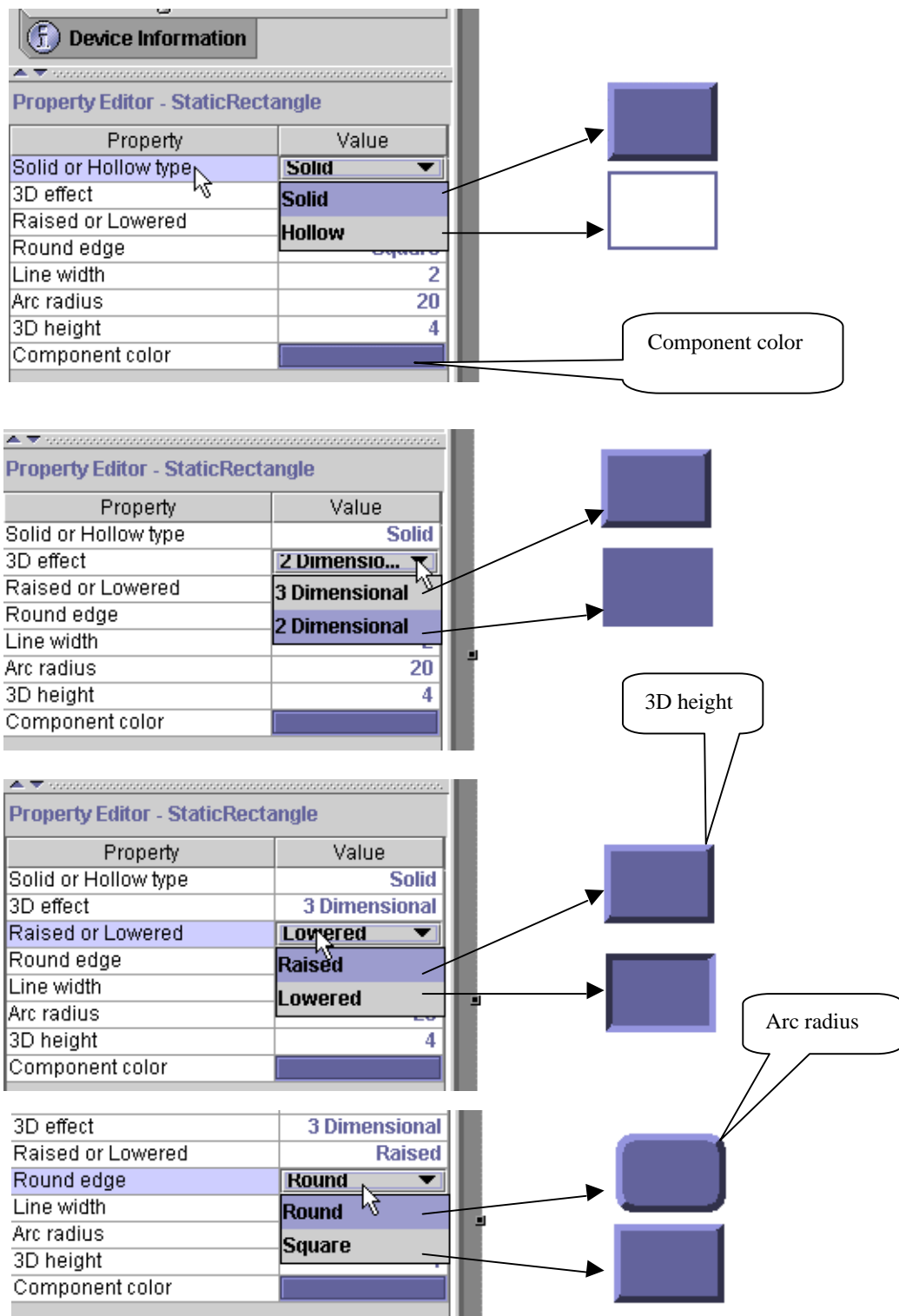
[Fig 3.39.1 Setting Static Image Property]

8.2 Setting Static Text Property



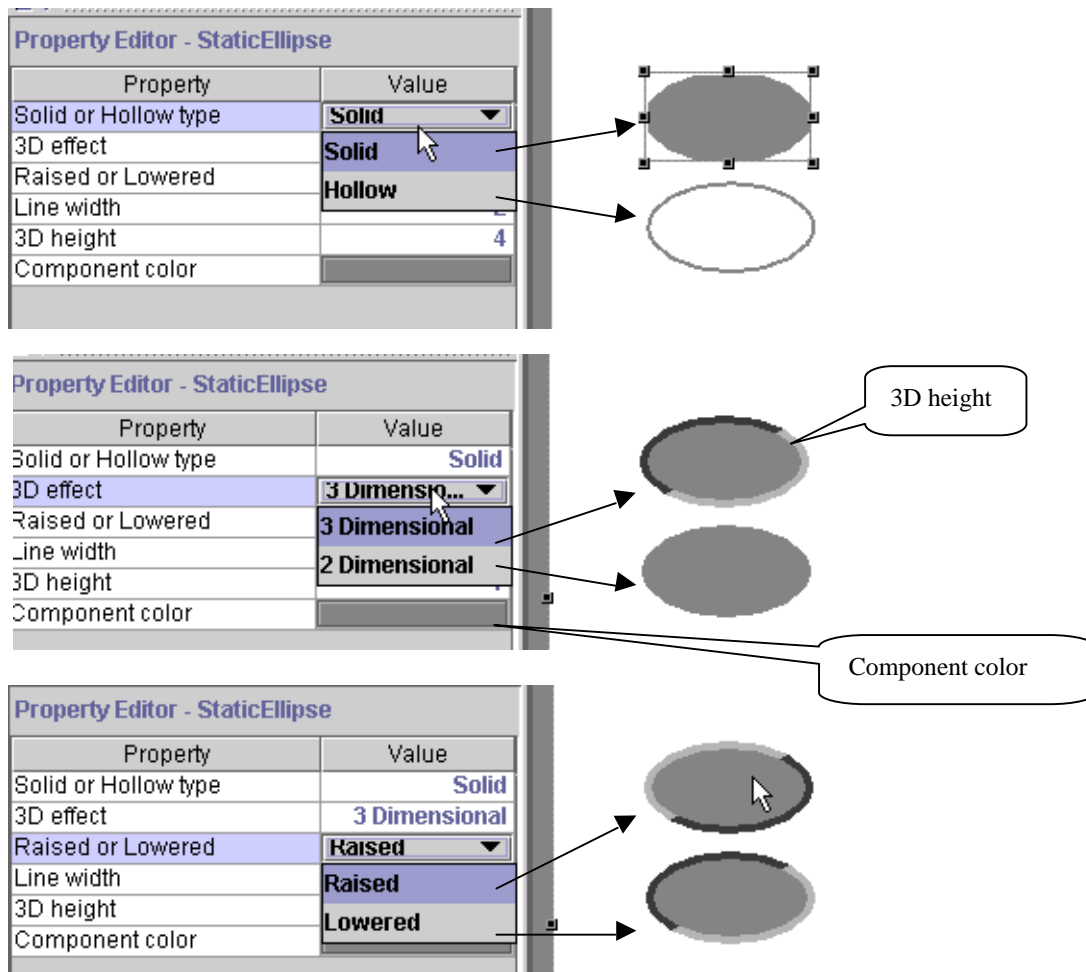
[Fig 3.39.2 Setting Static Text Property]

8.3 Setting StaticRectangle Property



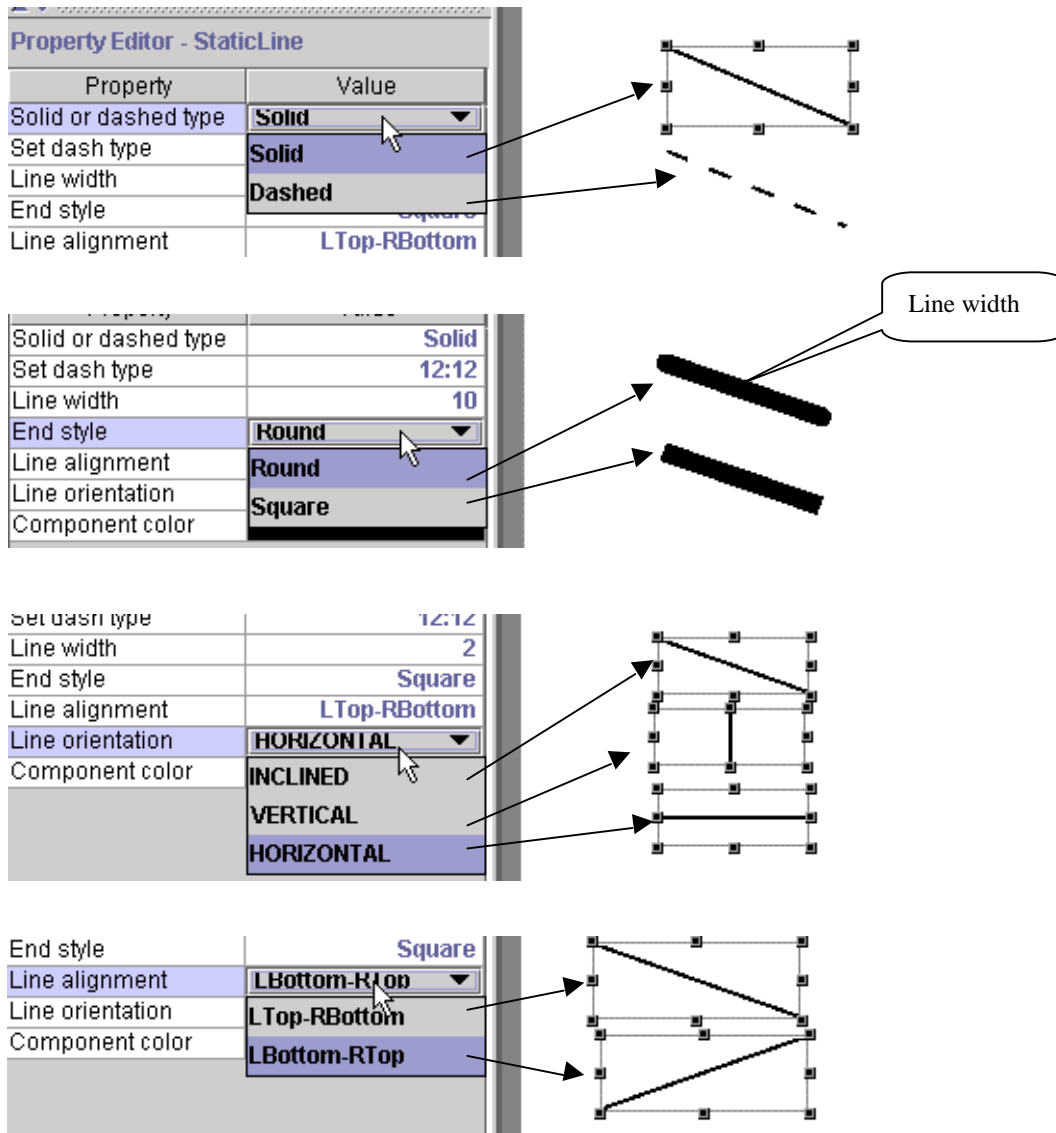
[Fig 3.39.3 Setting StaticRectangle Property]

8.4 Setting StaticEllipse Property



[Fig .3.39.4 Setting StaticEllipse Property]

8.5 Setting StaticLine Property



[Fig.3.39.5 Setting StaticLine Property]

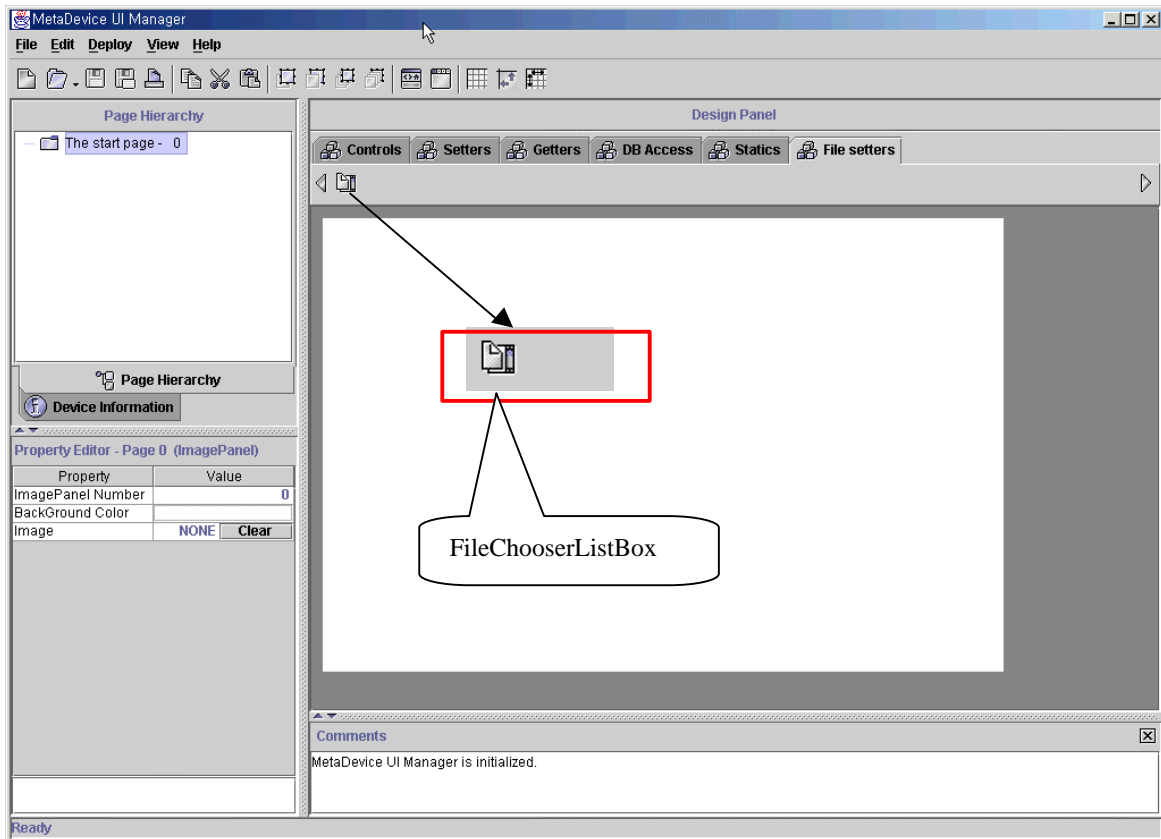
9. File Setters Component group

(1) What is File setters Component ?

Sends selected file to the Server.

(2) File setters Component

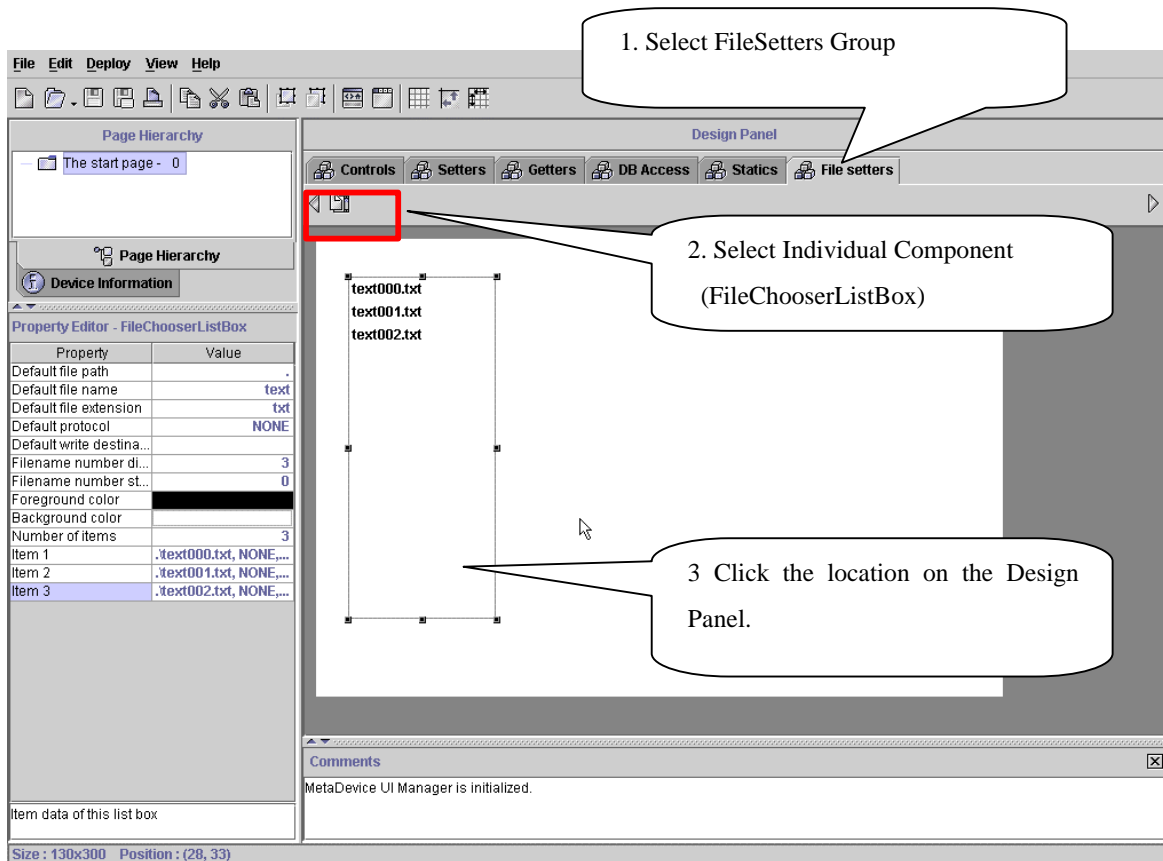
- FileChooserListBox : Sets file to be sent to the Server.



[Fig 3.40.1 File setters Components]

(3) File Setters Editing

- Select the File Setters Group from the Design Panel.
- Click the location where the component should be located.
- Set Property.

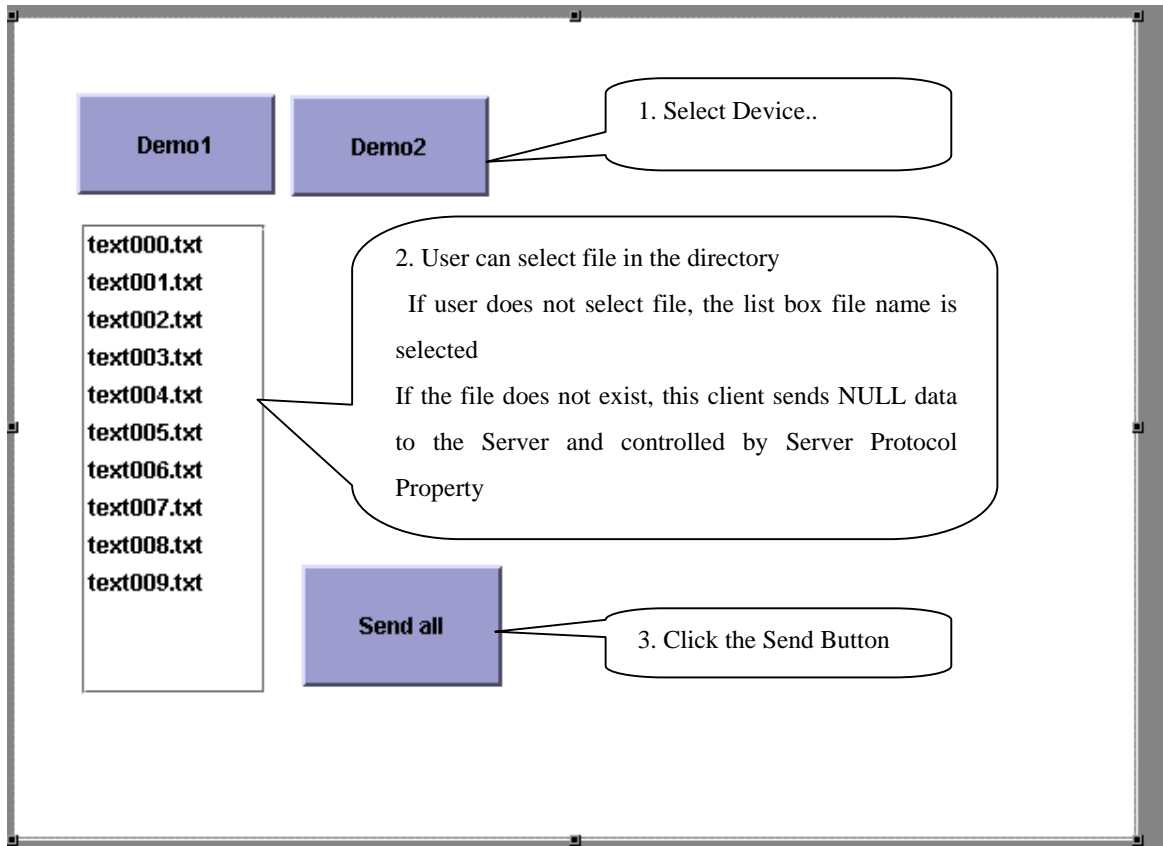
**[Fig 3.40.2 File Setters Component Editing]**

9.1 FileChooserListbox

(1) Working at Run Time

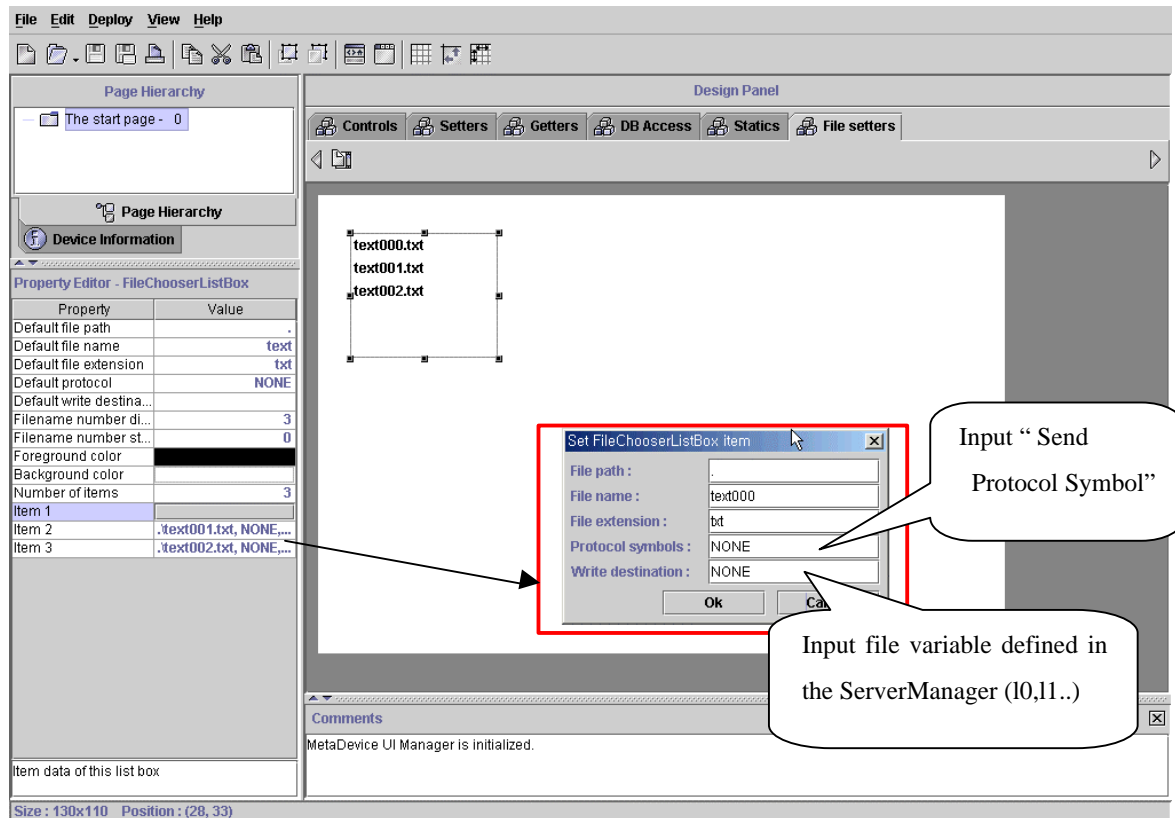
This component can be used as Application only.

The file name of list box can be selected.



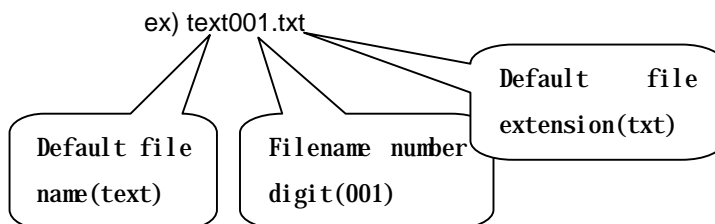
[Fig 3.41.1 FileChooserListbox]

(2) Setting the Property at Design Time



[Fig 3.41.2 Set FileChooserListbox Property]

- Default file path : Input the path for reading File.
Input directory based on the Application location.
- Default file name : Default file name.
- Default file extension : Default file name extension. ex)txt
- Default protocol : Default send protocol symbol to be listed on the item
- Default write destination : Protocol file variable (I0,I1..)
- Filename number digit : Input digit following Default file name.
- Filename number start of index : Input initial index number of file.
- Number of items : The number of items to be made
- item N : To set property of each file made, open Set FileChooserListBox and select property.

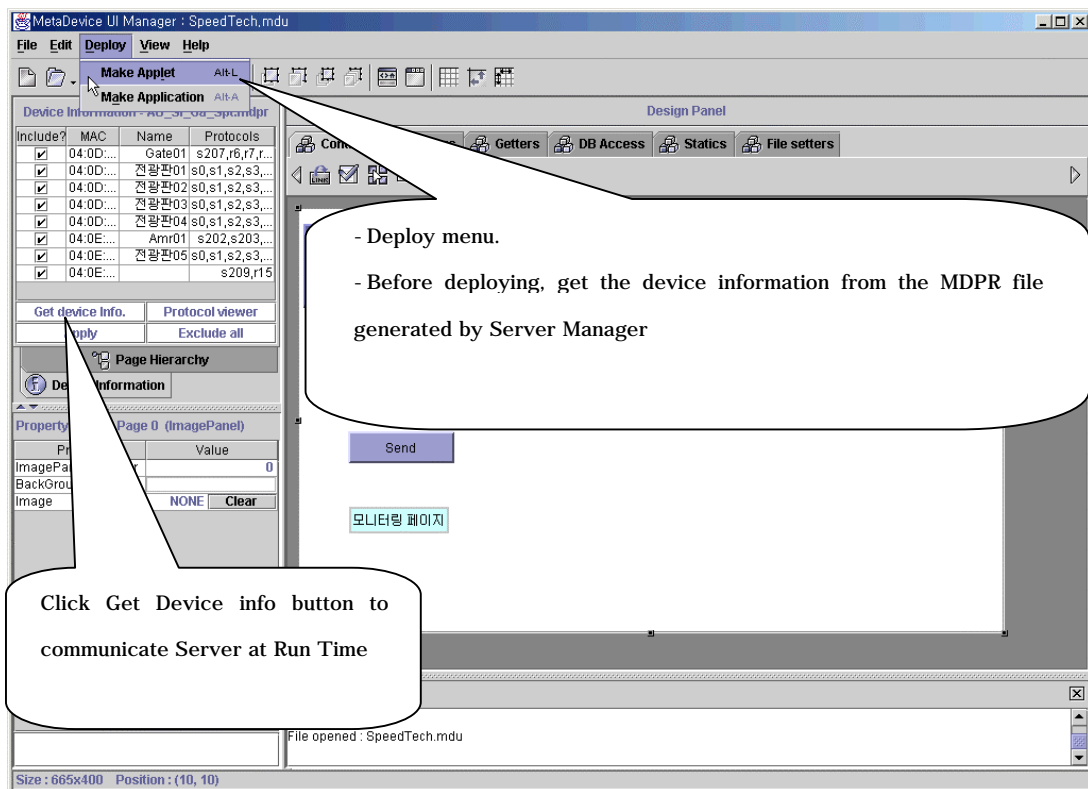


9. Executing Deploy

(1) What is Deploy ?

Deploy is when the user generates applet/application programs to communicate with the Server after completing UI design using UI Manager

When you deploy applet, a separate web server is needed for the applet.

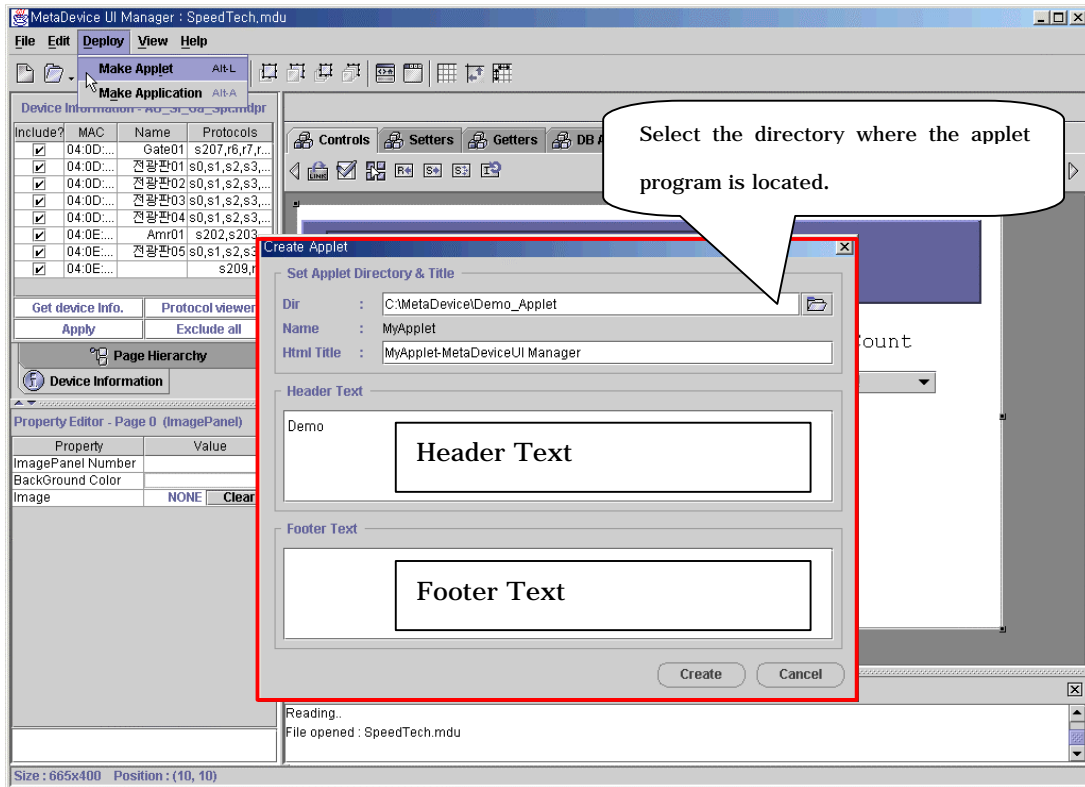


[Fig 3.42 Deploy Menu]

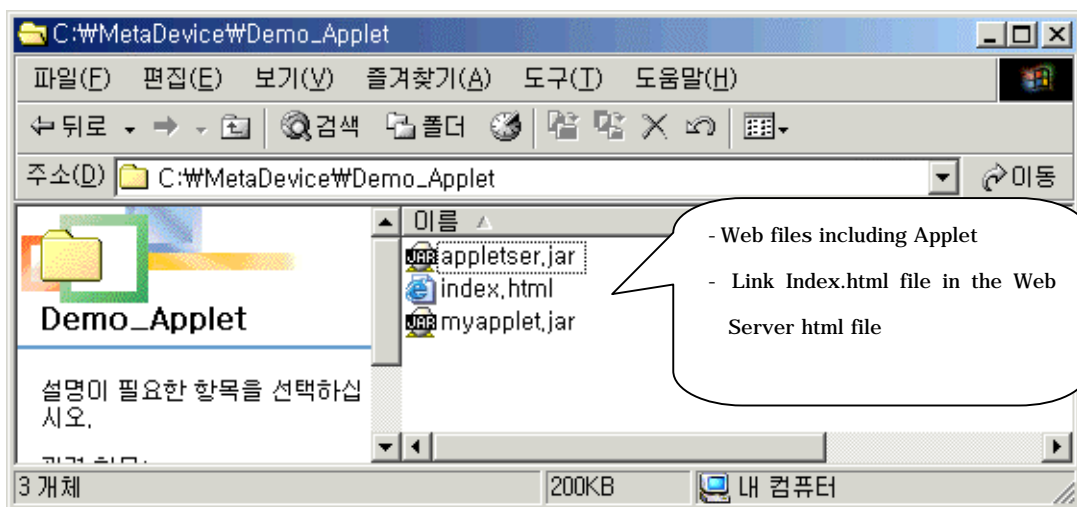
(2) Applet File Creation and Working at Run Time

- Select "Make Applet" from the **Deploy Menu Bar** and [Fig 3.43] window will appear.

Select the directory for the Applet program, type in headnotes and footnotes and click "Create " to create Applet program in the selected directory.

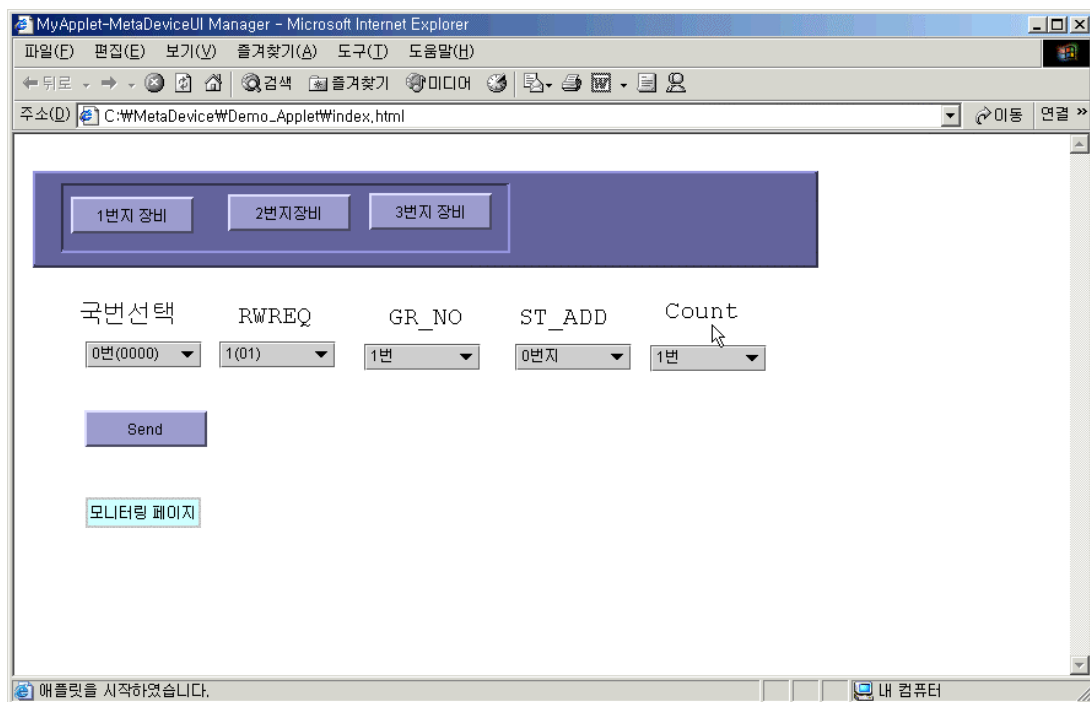
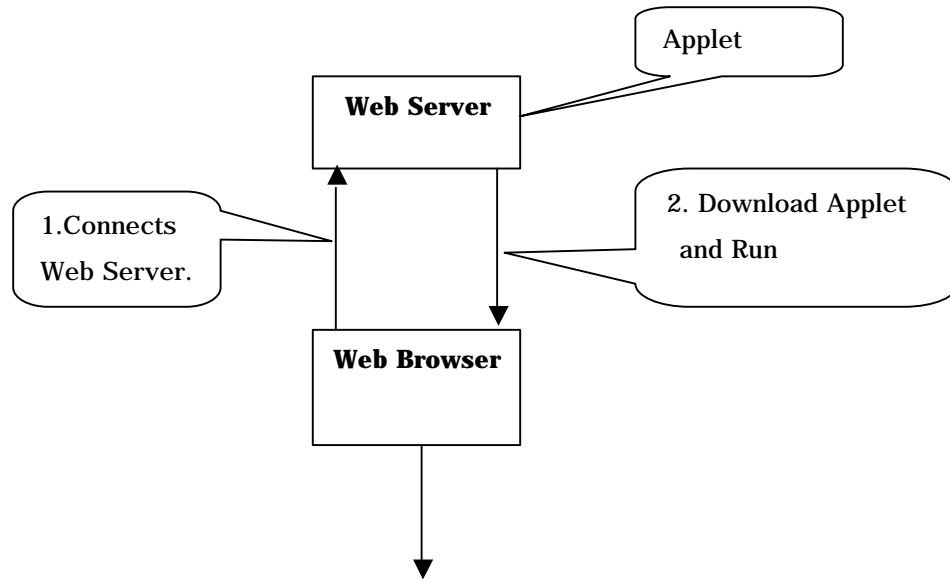


[Fig 3.43 Create Applet File]



[Fig 3.44 Web files created by UI Manager]

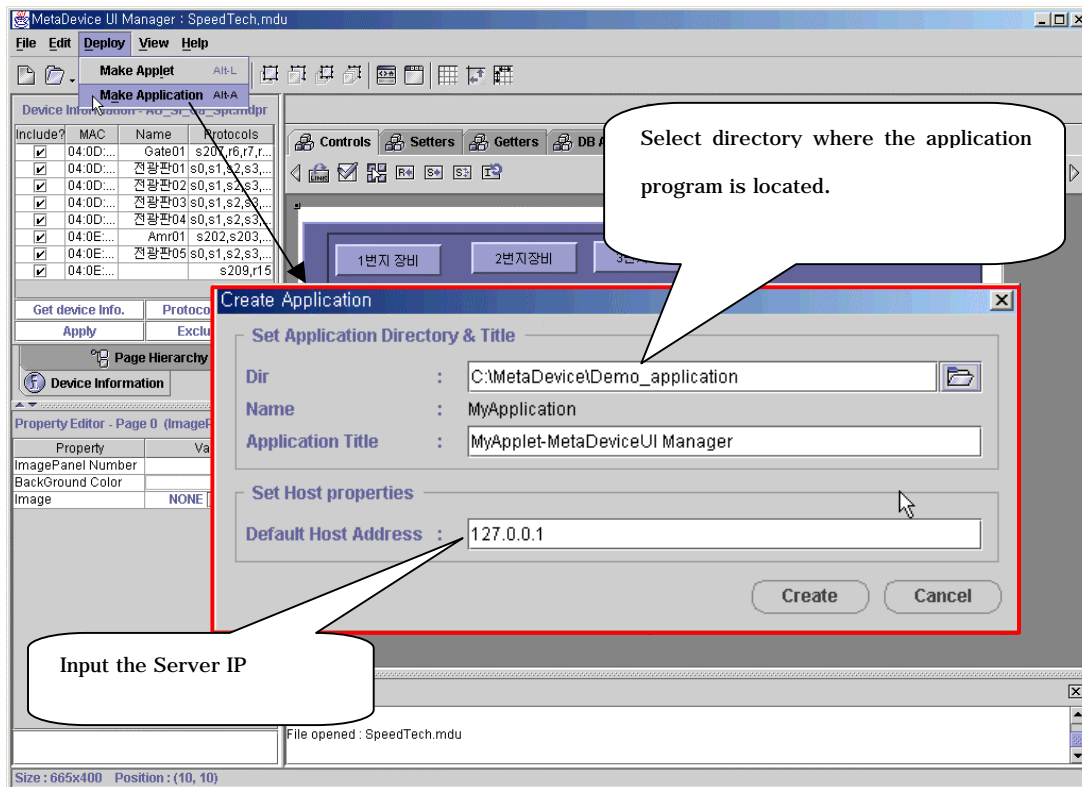
- To run applet program, a separate Web Server is needed.



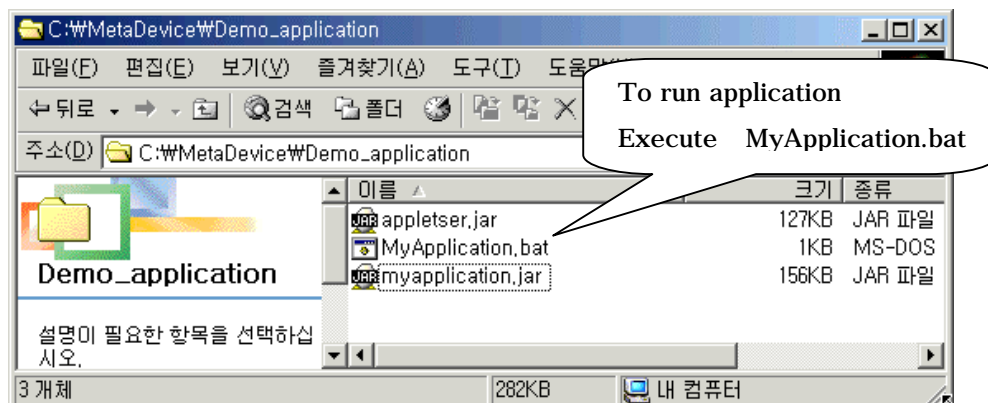
[Fig 3.45 Run Time Applet]

(3) Application File Creation and Working at Run Time

- Select "Make Application" from the **Deploy Menu Bar** and [Fig 3.46] window will appear.
Select the directory where the Application program will be made, type in the host IP and click "Create" to create Application program in the selected directory.

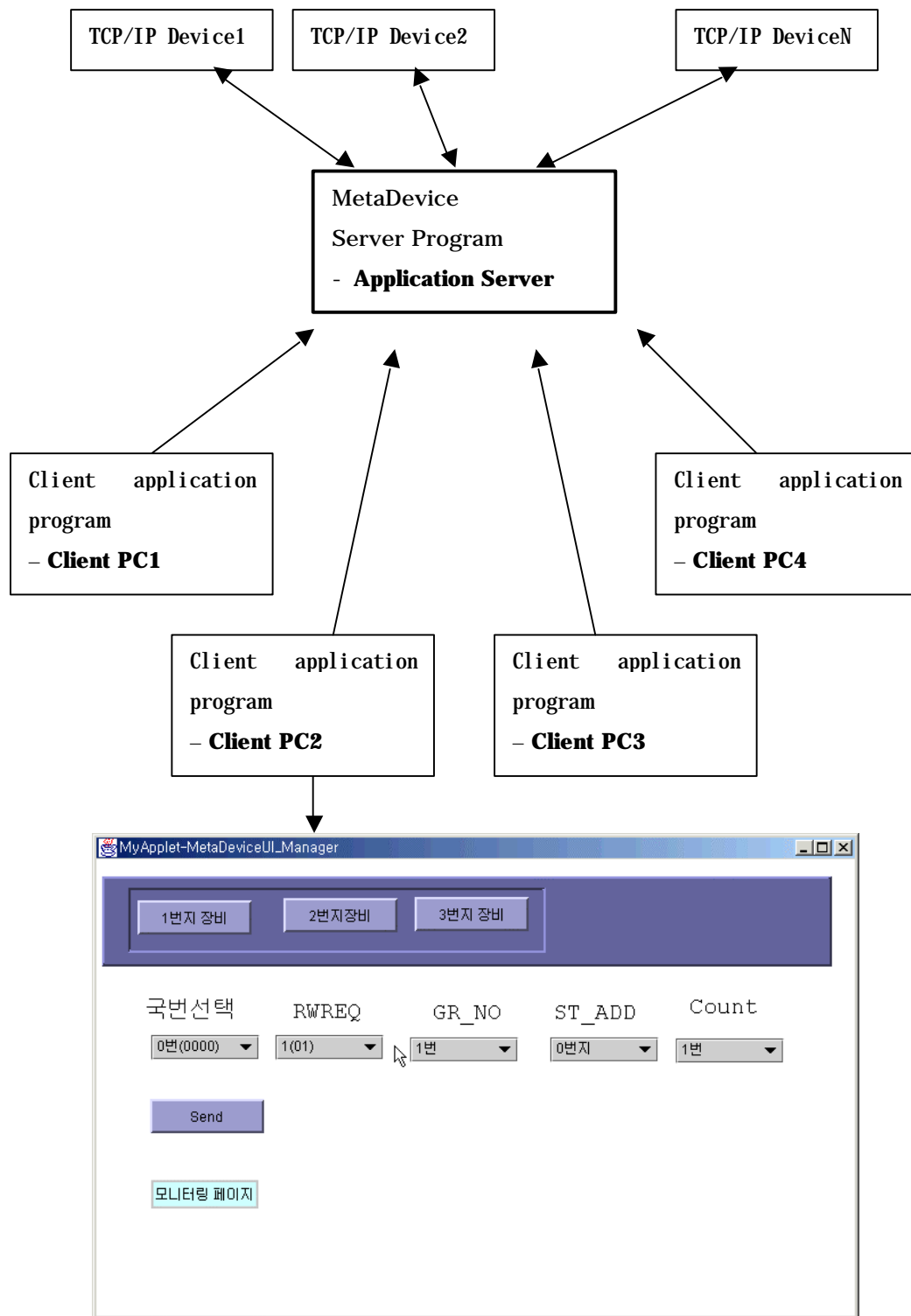


[Fig 3.46 Create Application File]



[Fig 3.47 Viewing Application files]

- Application can be run while the Server Program is running



[Fig 3. 48 Application Program at Run Time]

Part IV Examples

1 Sample Procedure

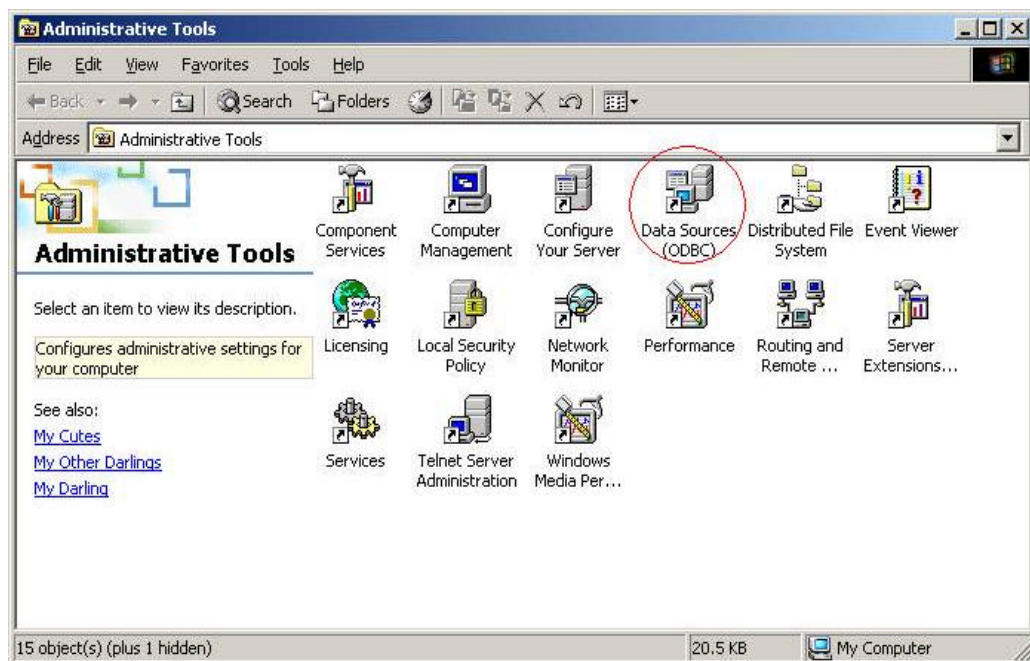
You can experience the usages of MetaDevice quickly and simply without actually connecting to the Device. The sample condition is when the Server Manager daemon type is Server(waiting for Device to connect to the Server), when the Semantics is built as Server, and when the GUI receives Protocol data from the Server.

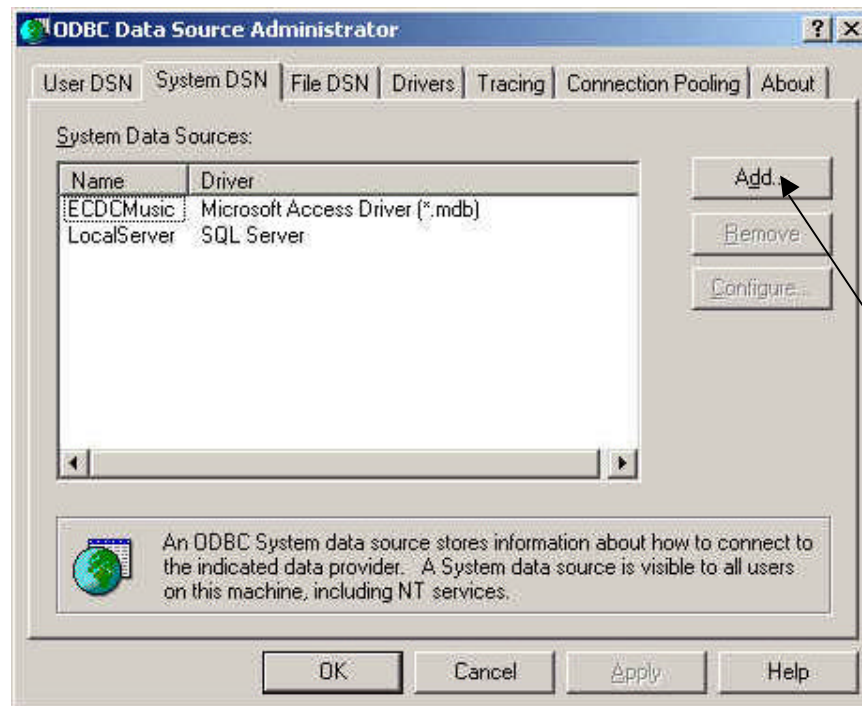
The Server Program will permanently wait because the Device is not connected to the Server and the Protocol Data value will always maintain its initial value. Therefore, the GUI component will display this initial value.

- Setting Environment

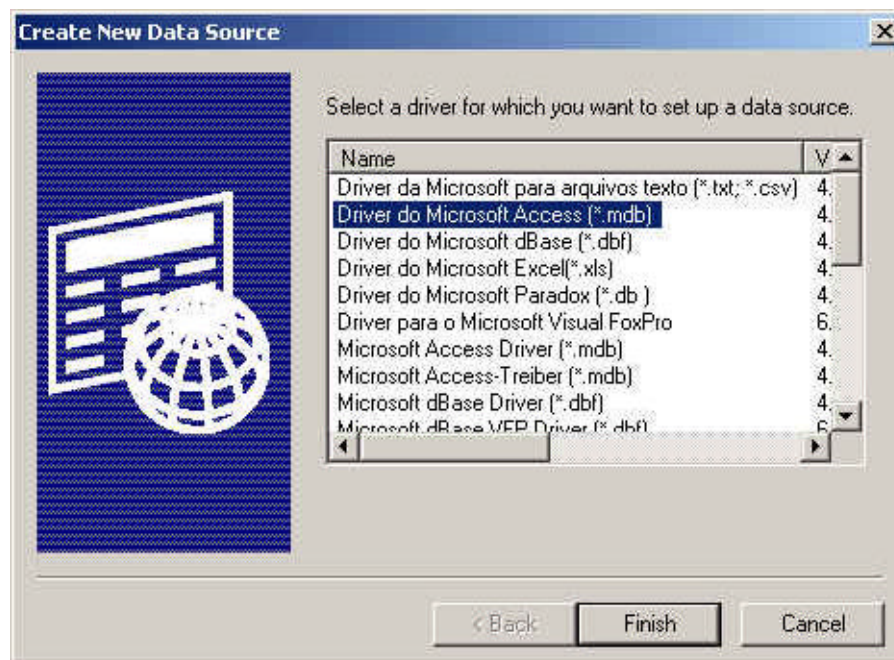
- (1) Download jdk1.3.1_02 install File from java.sun.com
- (2) After downloading, install jdk.
- (3) After installing, set java environment by following Part 1, section 1.2.3.
- (4) Execute the MetaDevice install Program distributed from SENA by following Part 1,section 2.
- (5) After installing MetaDevice, create MS Access DSN.

- Double Click DSN (ODBC) from the Start Menu -> Settings -> Control Panel

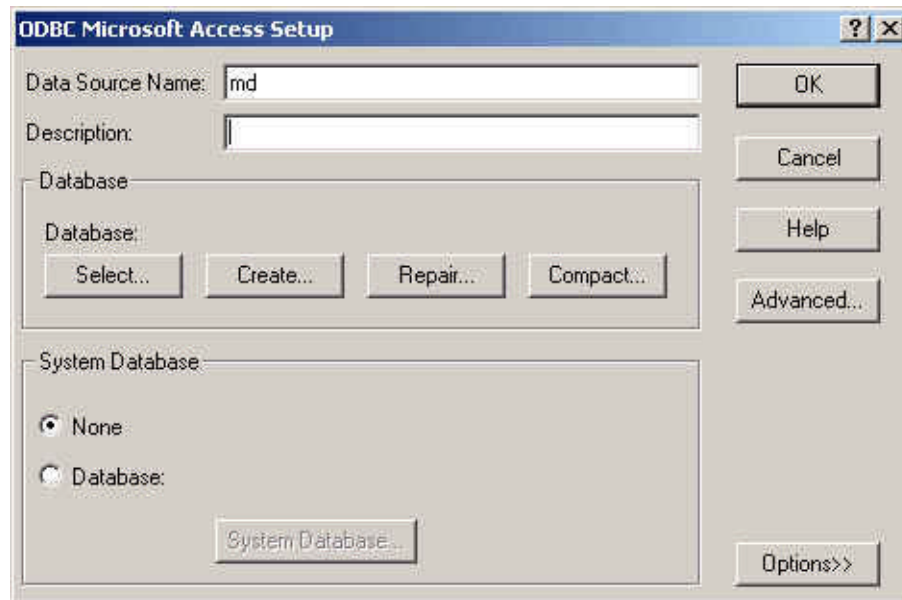




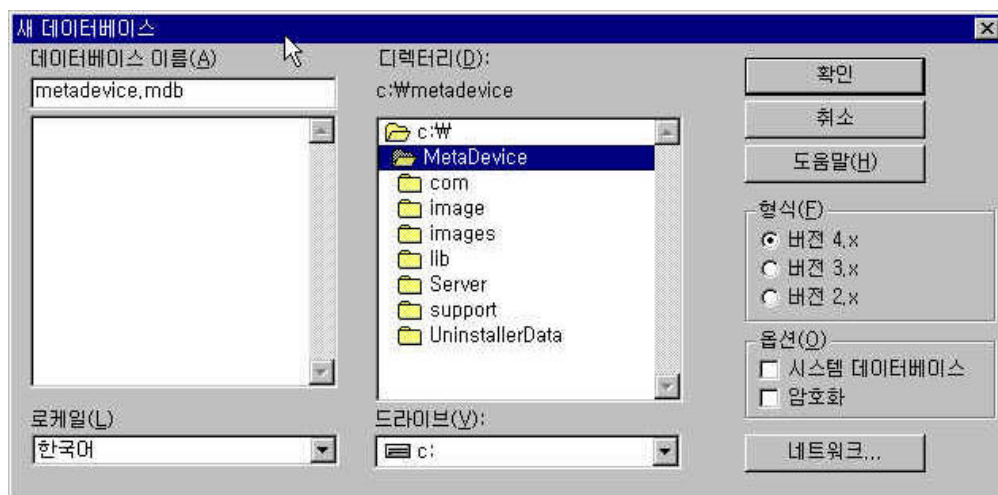
- Click Add from the user DSN Tab in the ODBC DSN Administrator.



- Double click Microsoft Access to select Driver.



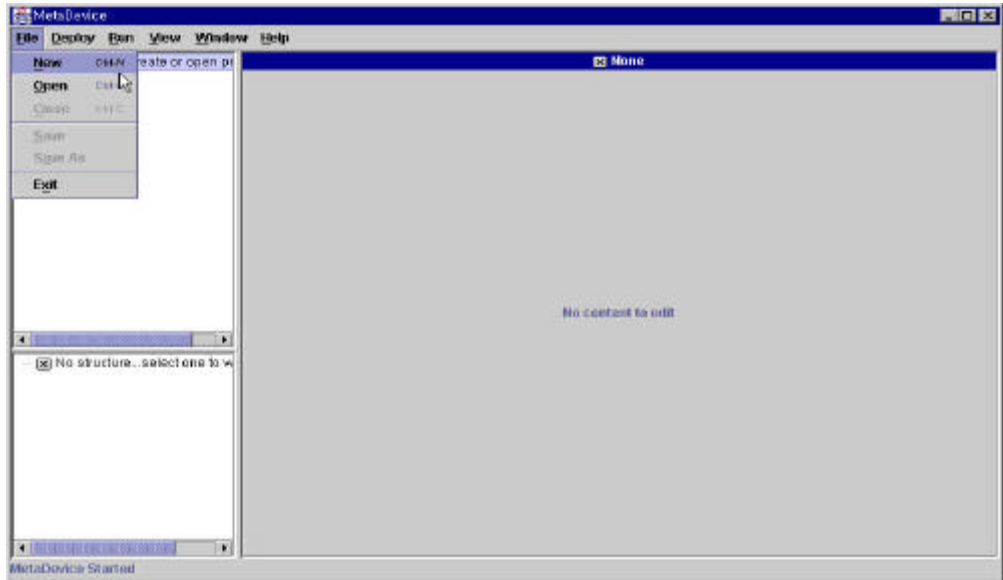
- When the Access DSN setting window appears, write “md” in the DSN field and click Create in the database tab.



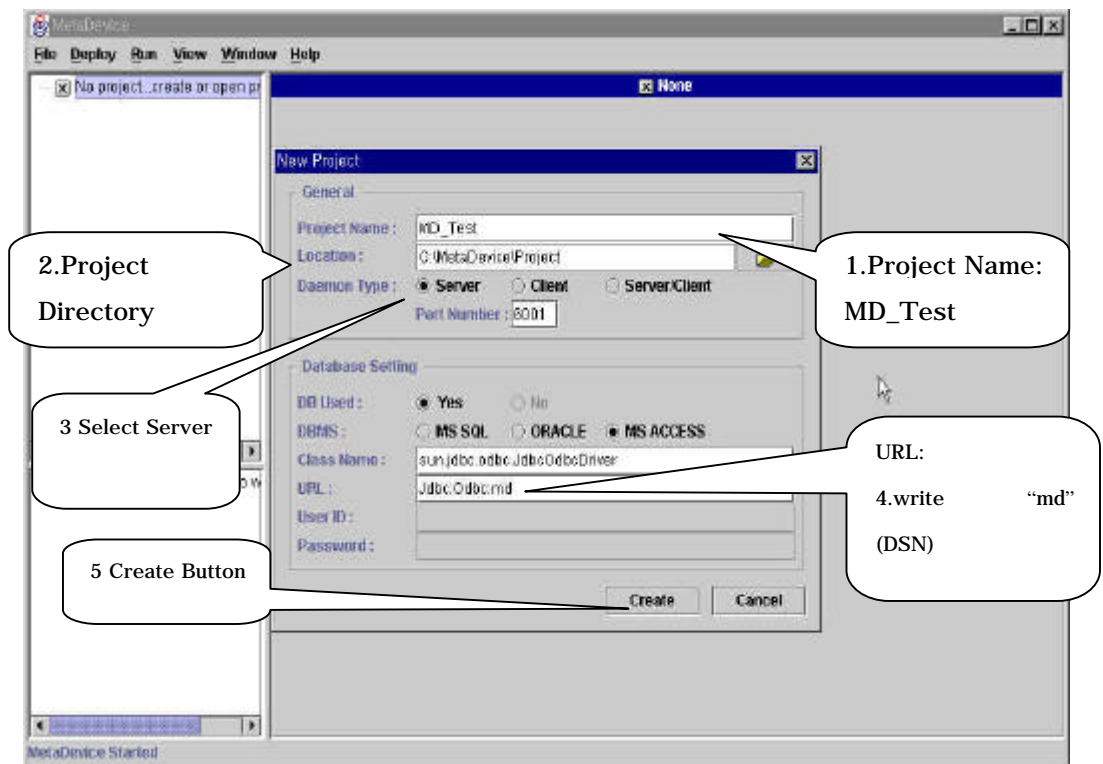
- When the database selection window appears , write “metadevice.mdb” as the database name and select MetaDevice install directory.
- Click Confirm and the message “ database is created successfully” will be shown. Click “Ok” for all menus that had appeared to create database.

- Using the MetaDevice Server Manager

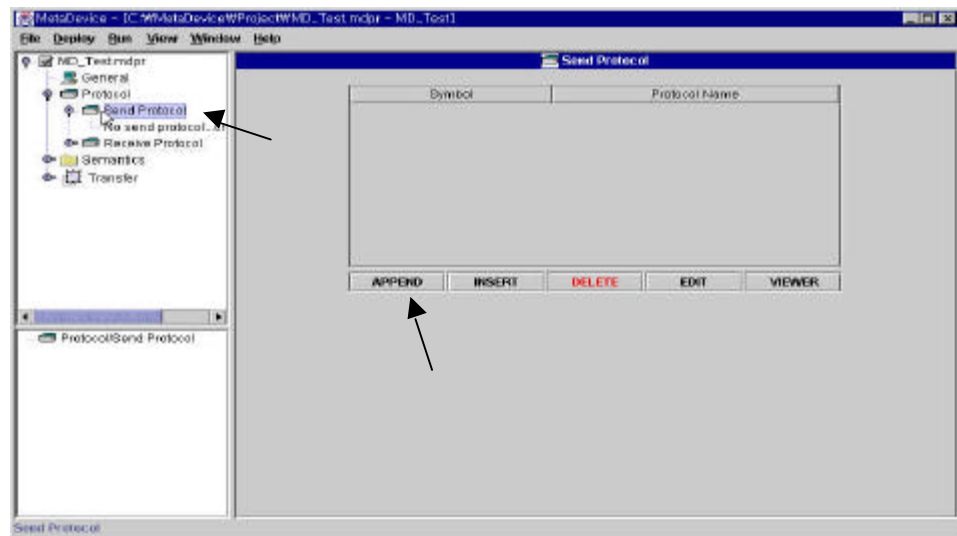
- (1) Create Project Directory under MetaDevice install directory.
- (2) Run Program by clicking the MetaDevice ServerManager ICON.



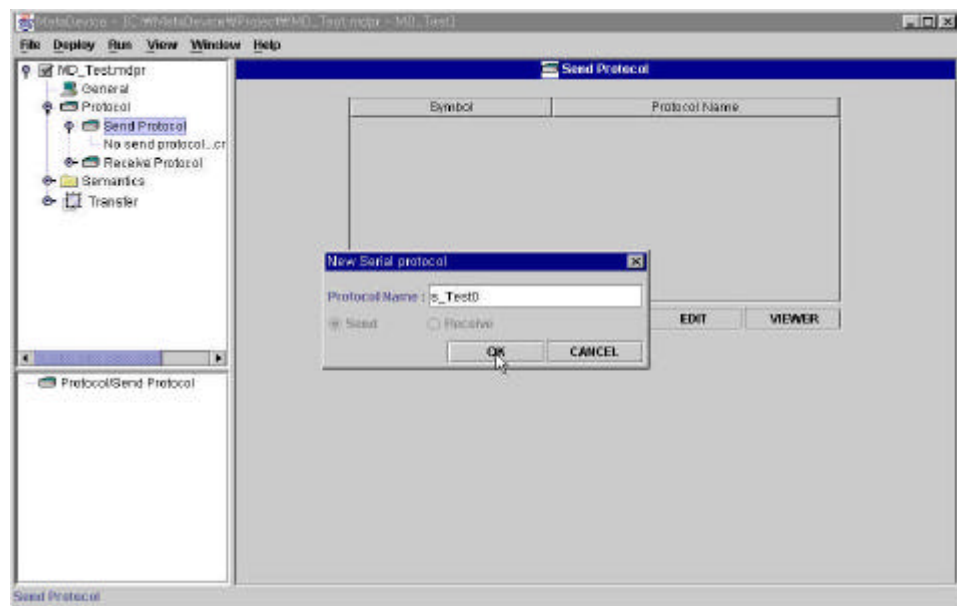
(3) Create new Project File from the File Menu.



(4) Click Create to see initial setting screen. Select Protocol and Send Protocol.



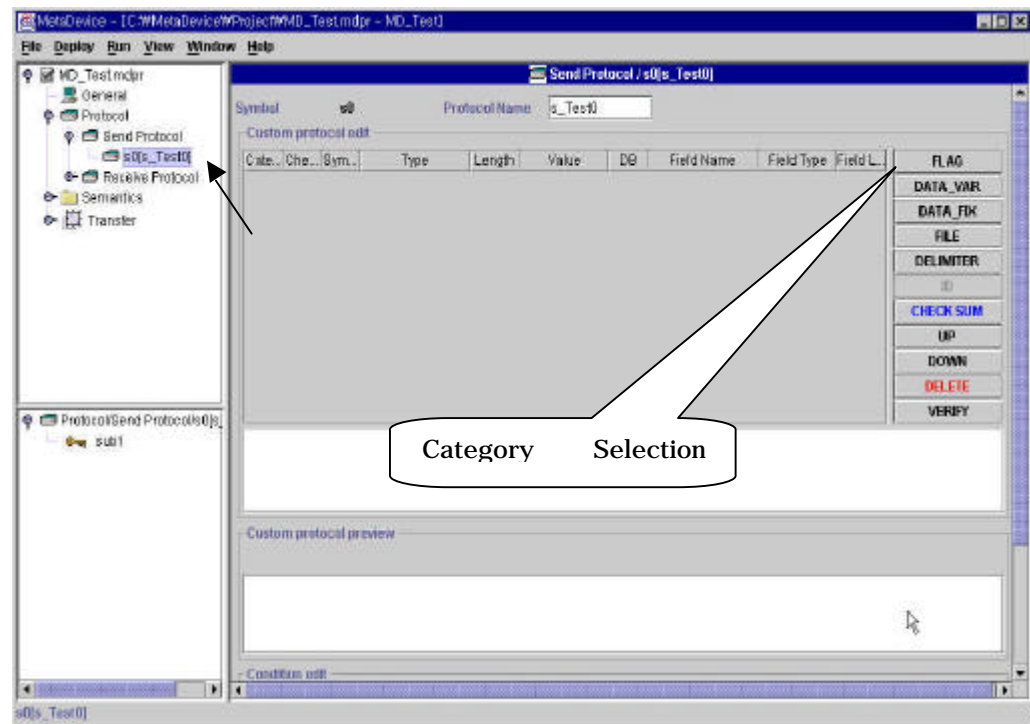
(5) Click "Append" from the Protocol Name Register Window.



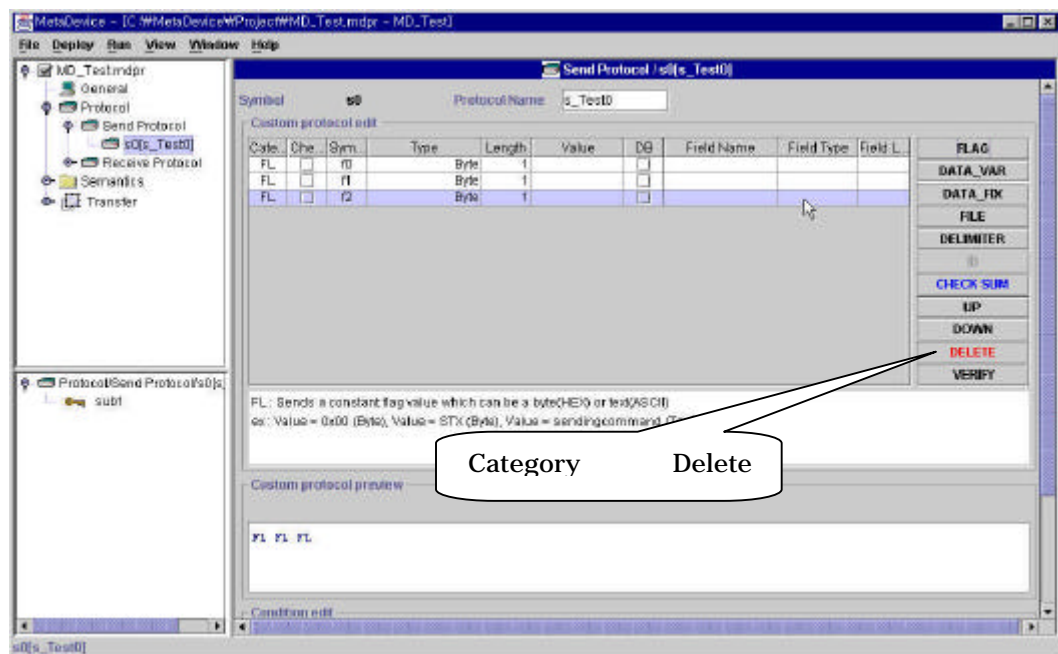
(6) write "s_Test0" for Protocol Name, and press "OK".

(7) After creating s_Test0 Protocol Name, edit Protocol Data.

Select Protocol Name and edit Protocol Data.

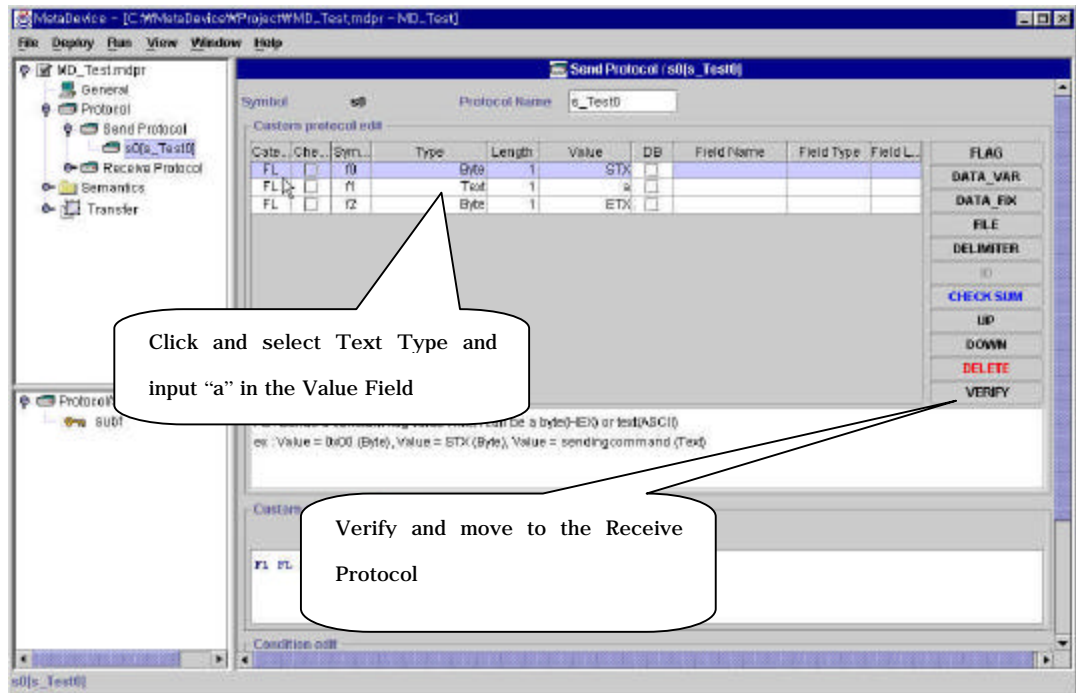


(8) Click Category Button "Flag" three times from the Protocol Editor window.

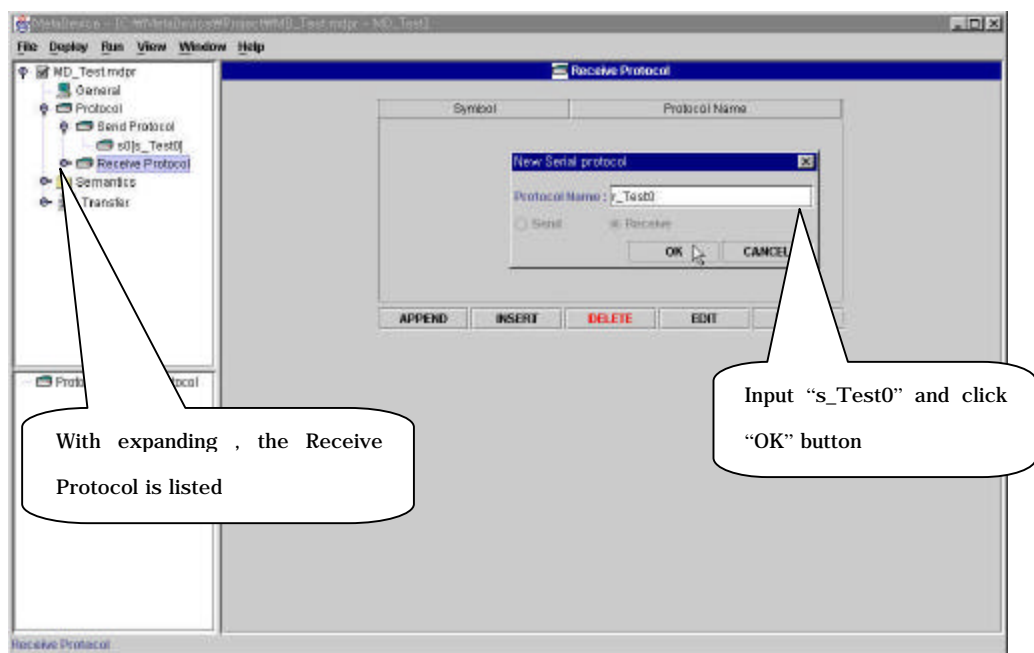


(9) From the Protocol Edit window click each value field and edit as follows.

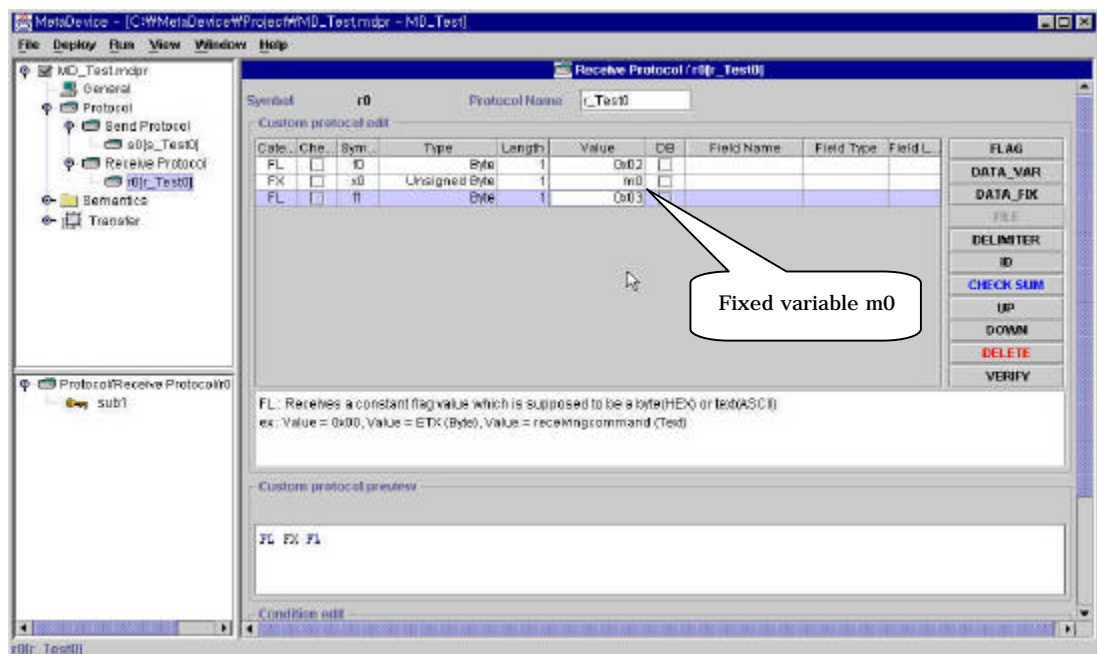
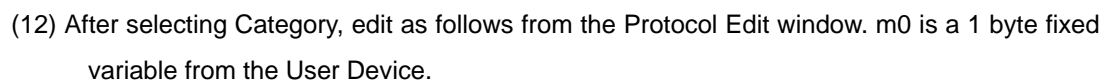
This example signifies that when sending data from Server to Device there exists a protocol same as “STX+a+ETX”.



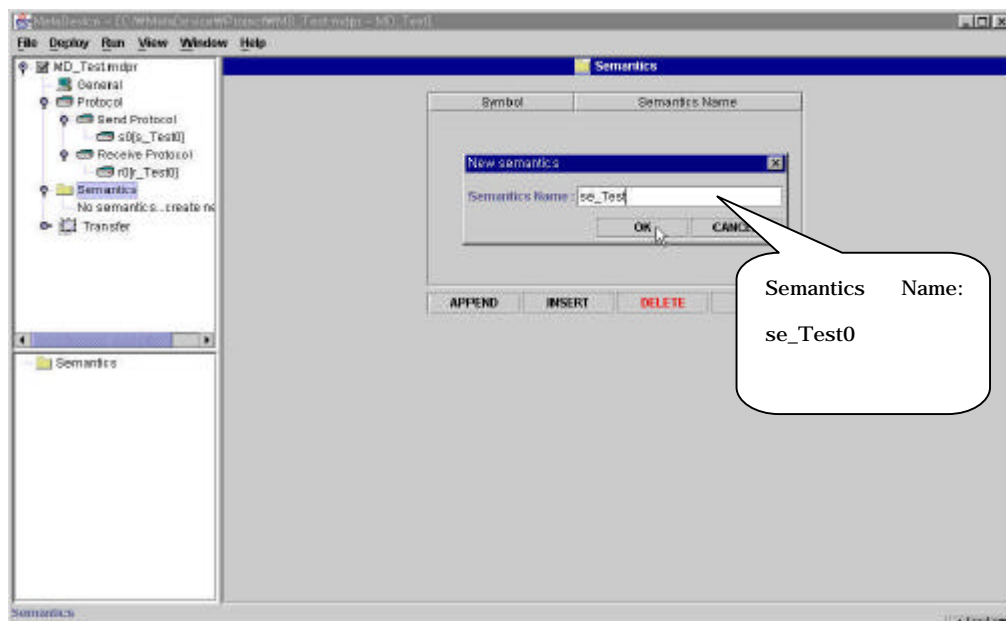
(10) Verify & Save and select Receive Protocol from the Project window.



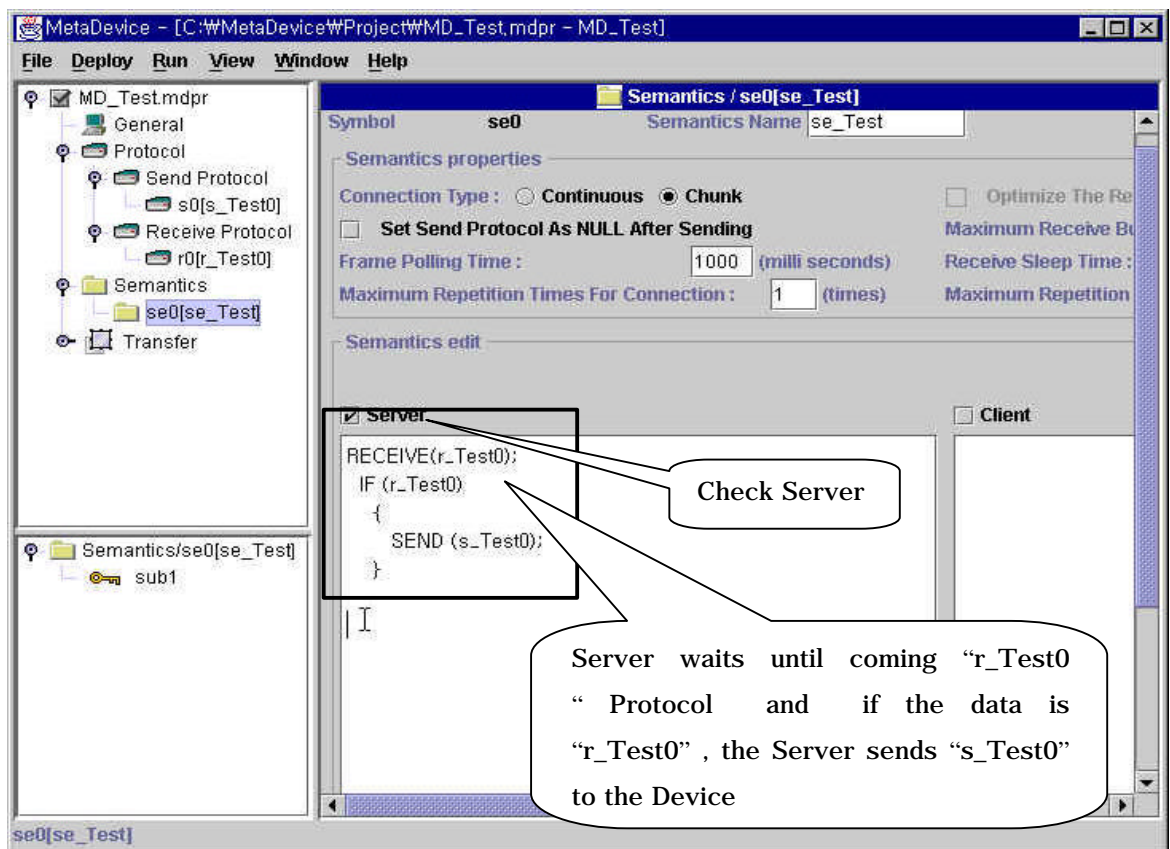
However, the 2nd Data type will be a fixed variable and the user needs to click the “DATA_FIX” Category selection Button. Category selection order is “FLAG”, “DATA_FIX”, “FLAG”.



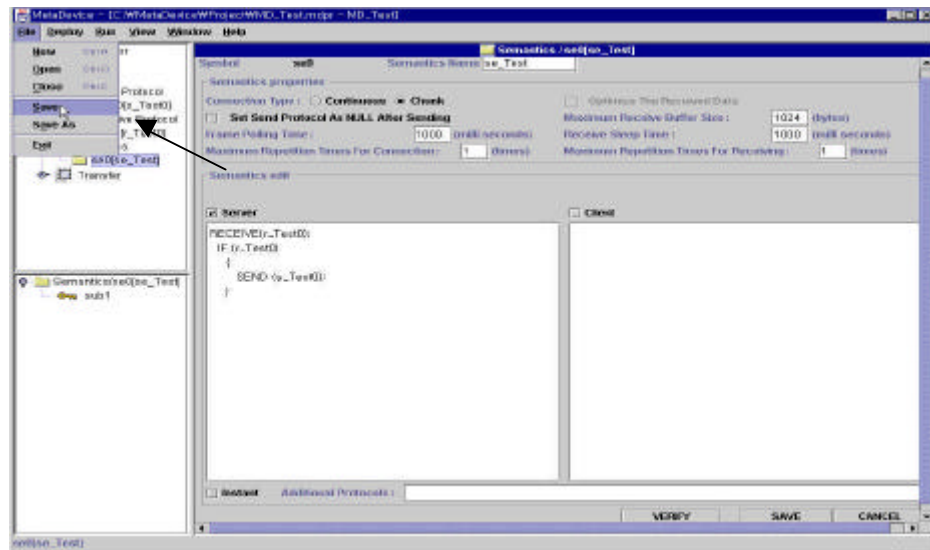
(13) After editing the Receive Protocol select “Semantics” from the edit window and register Semantics Name.



(14) Set Protocol communication rules from the Semantics Editor.

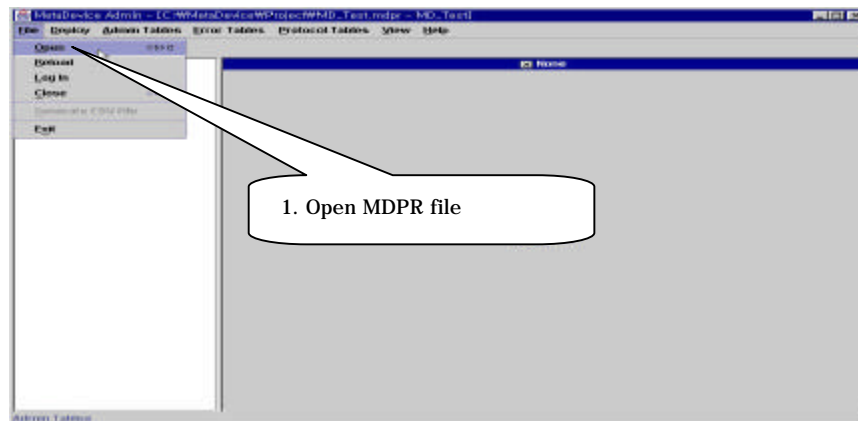


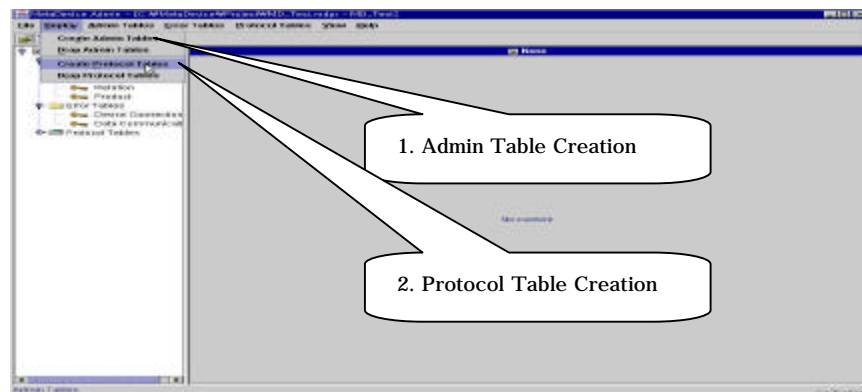
(15) Click “Save” from File Menu while working or after completion.



(16) After completing Protocol and Semantics editing, execute Admin work from the Admin Module.

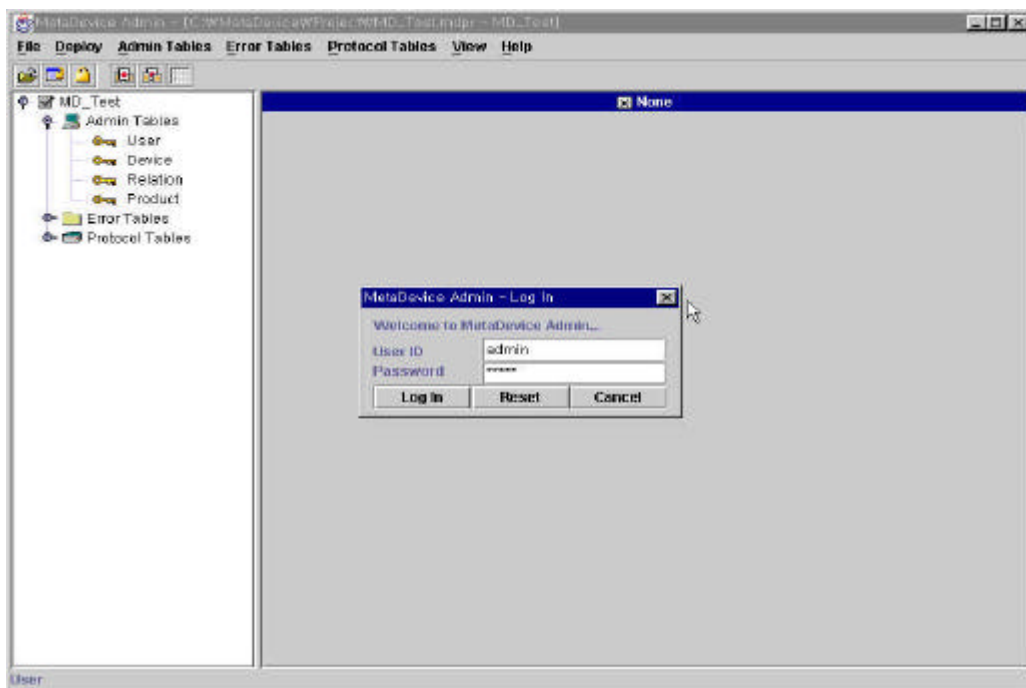
(17) Open “MD_Test.mdp” file and create Admin Table by selecting “Create Admin Tables” and “Create Protocol Tables” from the Deploy Menu.



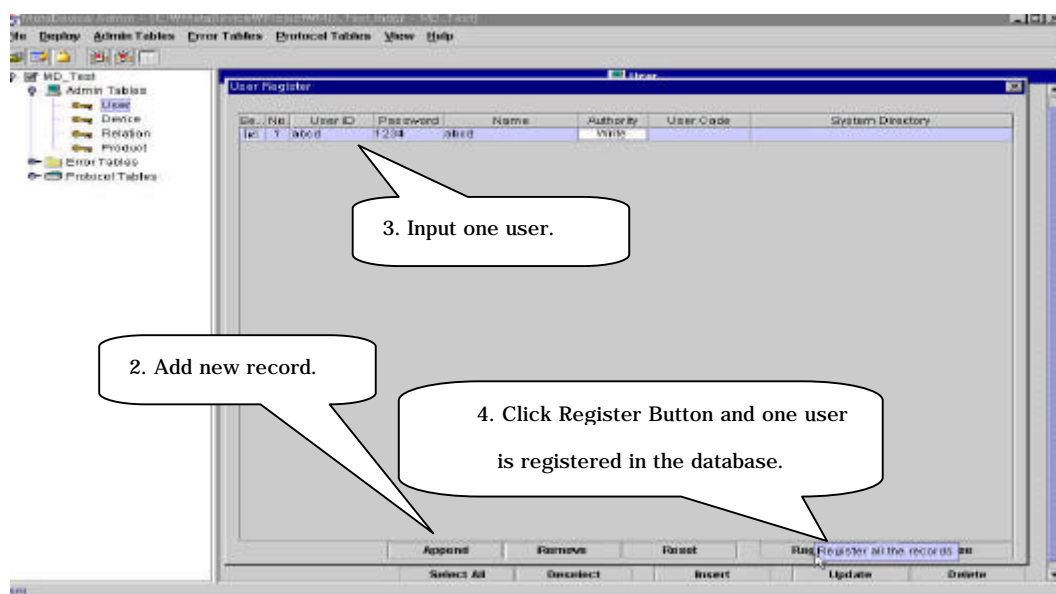
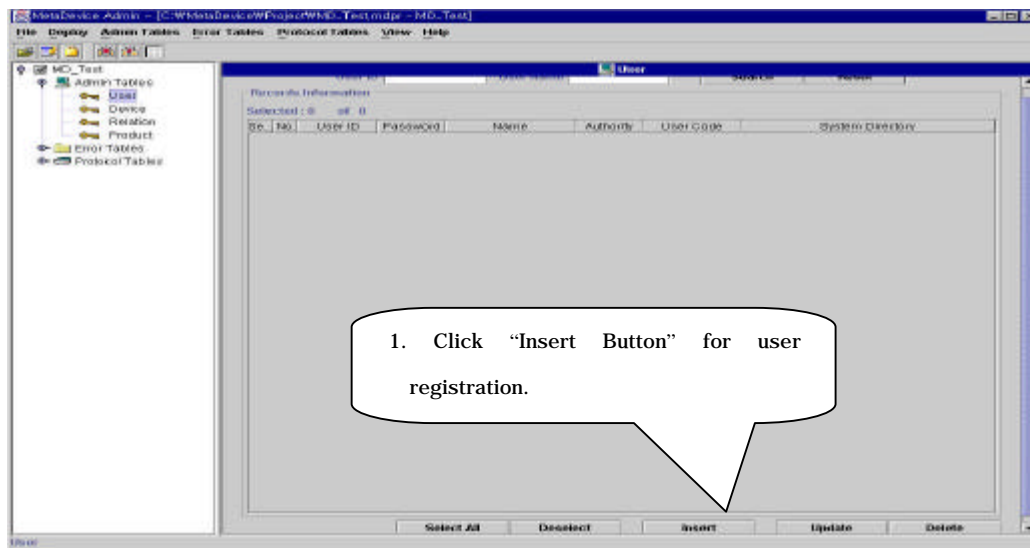


(18) After creating Admin Tables, select “User” Table for user registration.

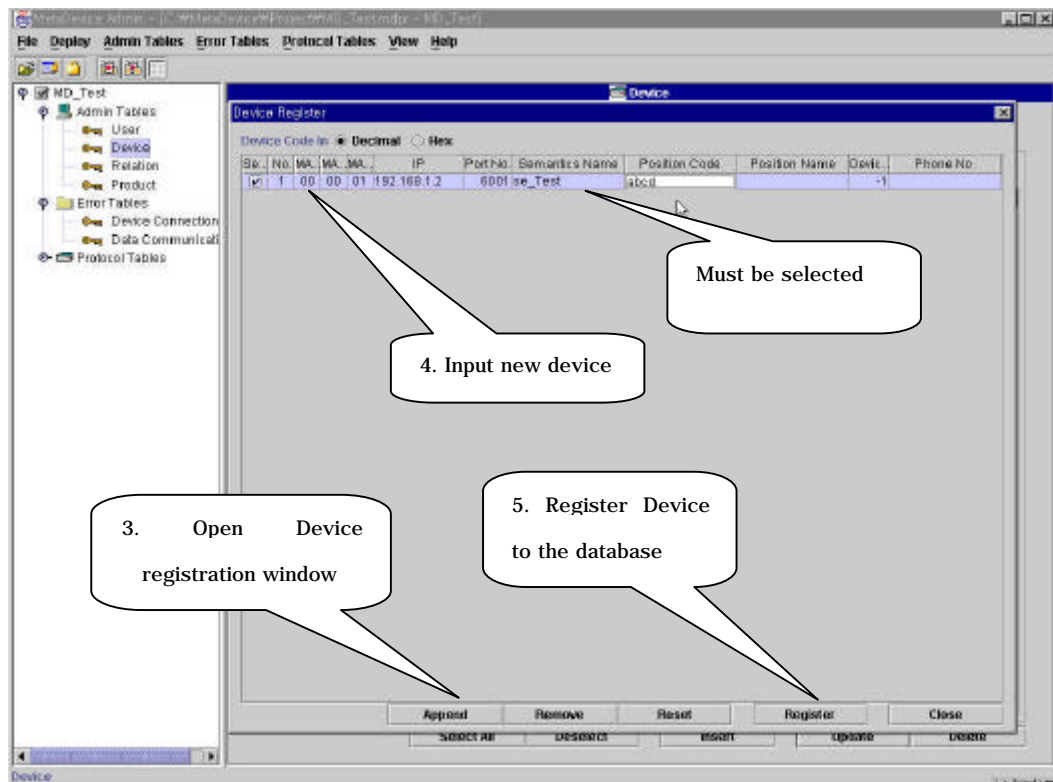
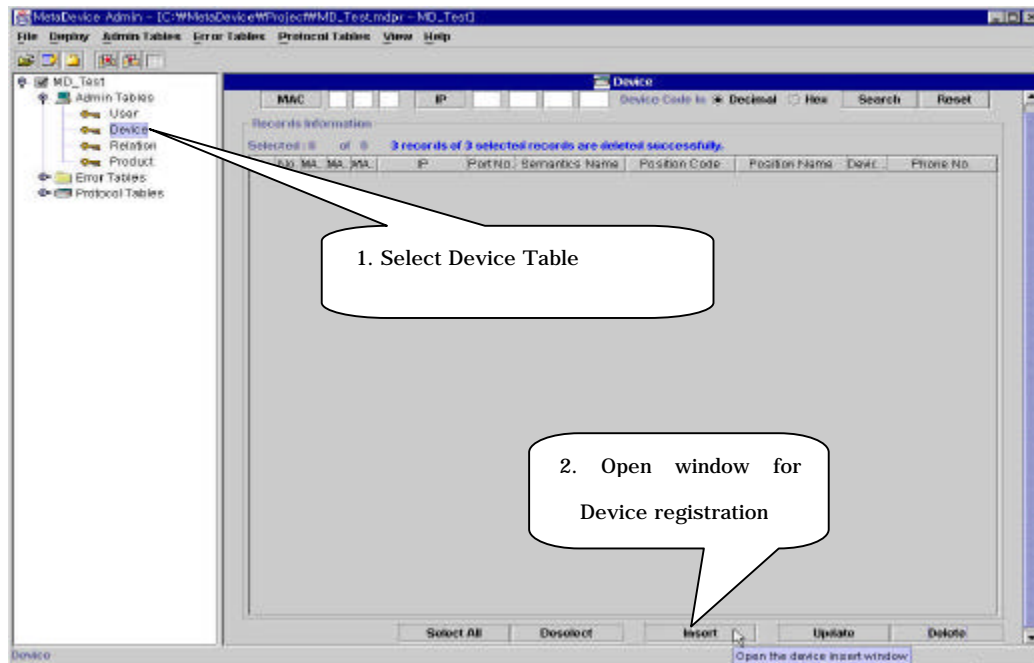
The Default Admin ID/Password is admin/admin and can be changed from the File Menu.



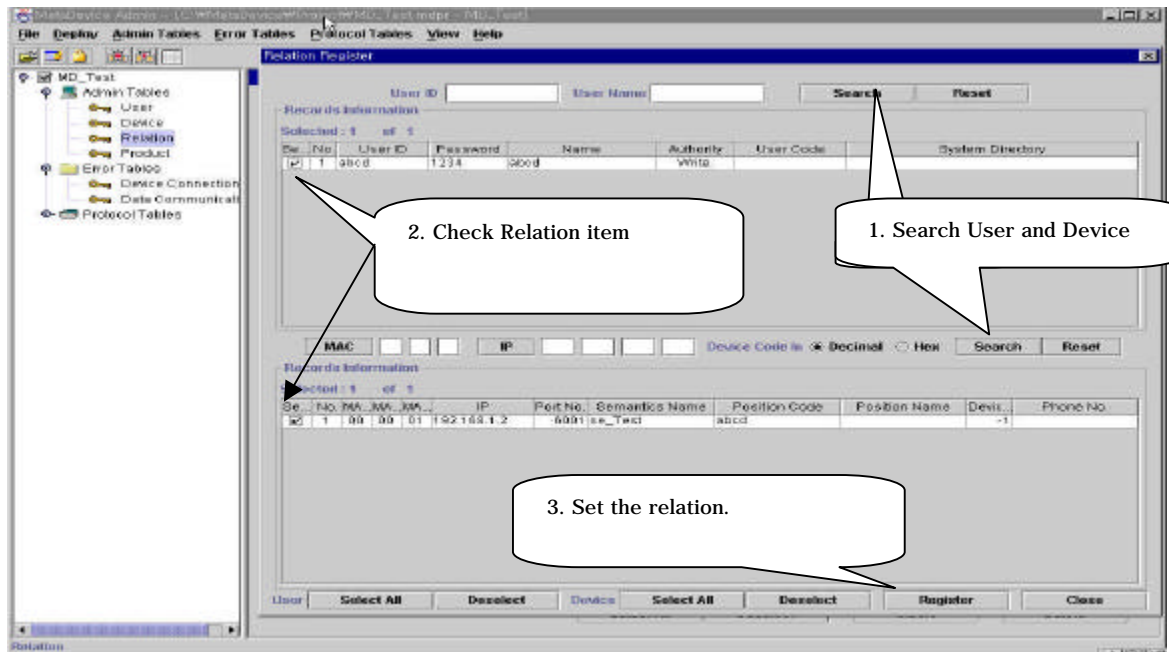
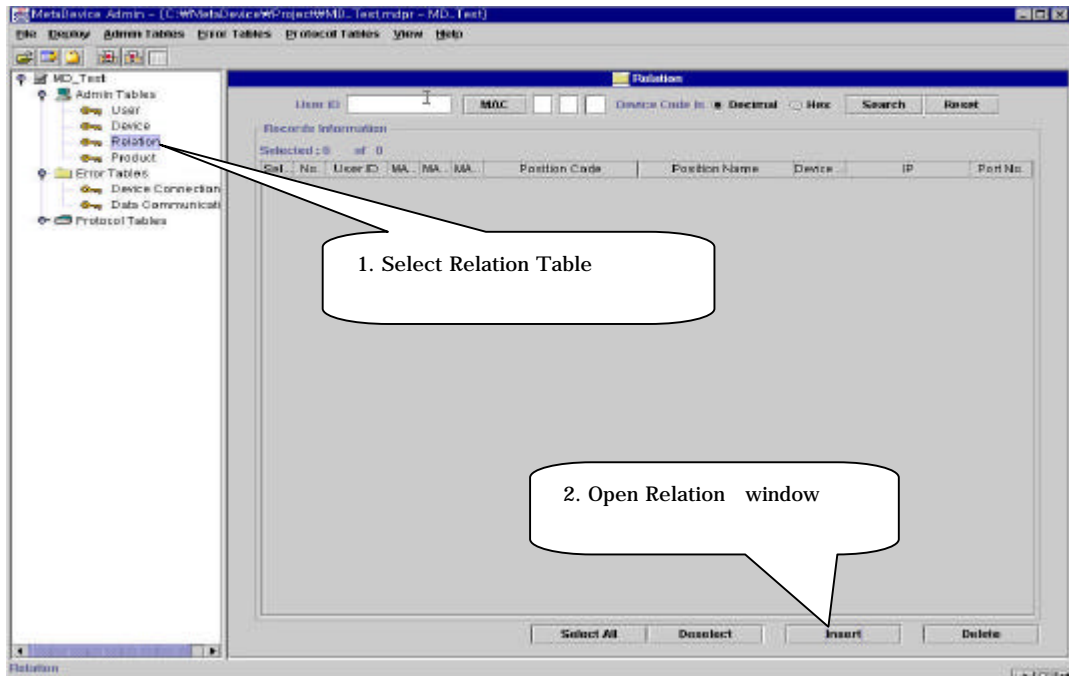
(19) Admin System ID/Passwd is “admin/admin” and register user when the registration window appears.



(20) Input user information and register Device as follows.

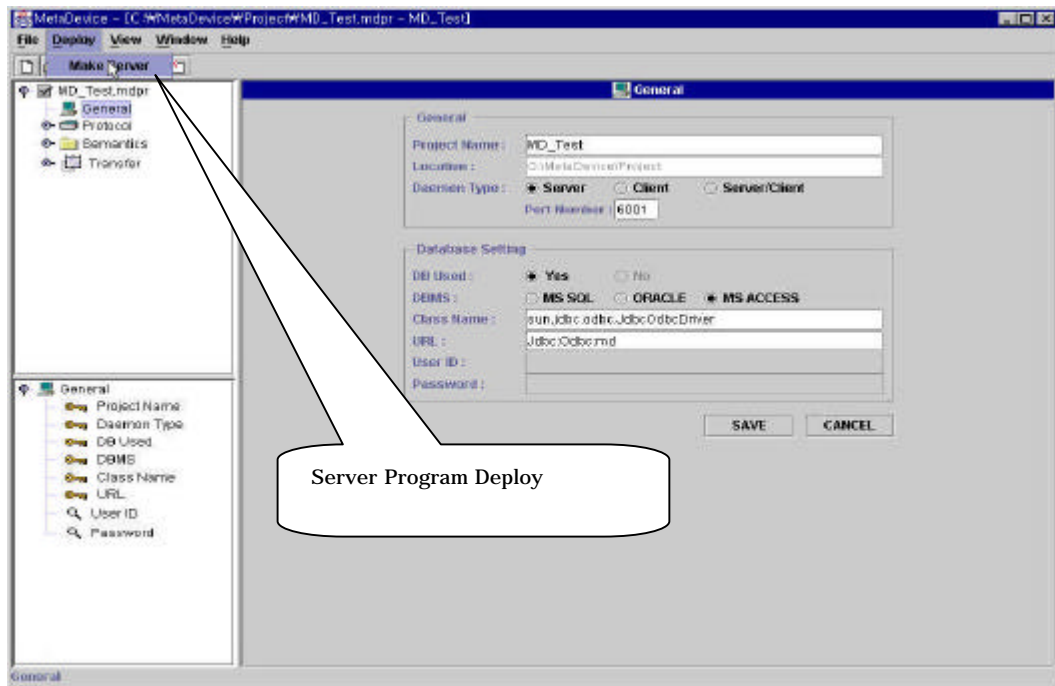


(21) Set the relation between Device and User.



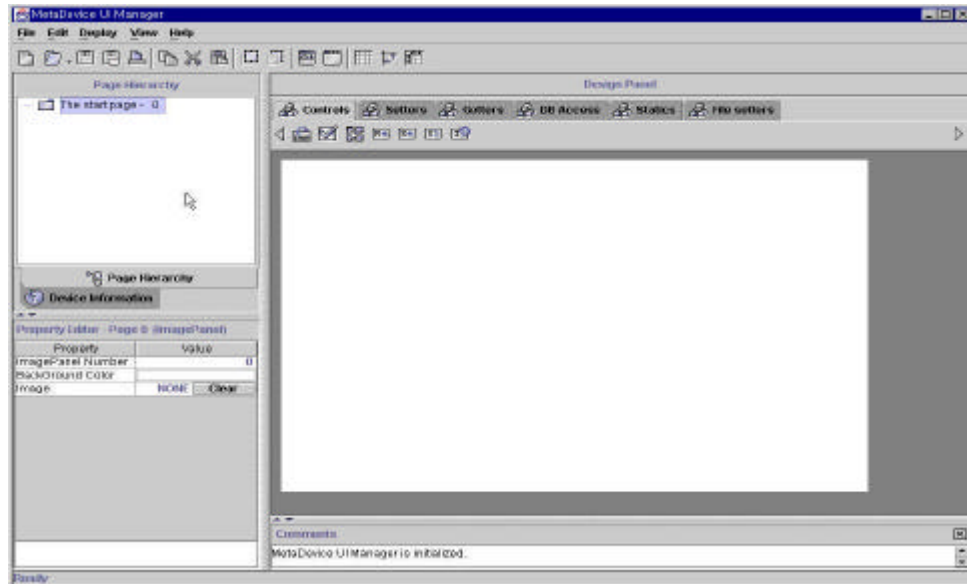
(22) Admin works has been completed. Exit Admin module.

(23) Finally, deploy Server Run-Time Program by selecting **"Make Server"** from the Deploy Menu. The **Project\Server\MD_Test.bat** file is created as **Server Run-Time Program**. Actual deployment can also be completed between procedure 15 and 16.

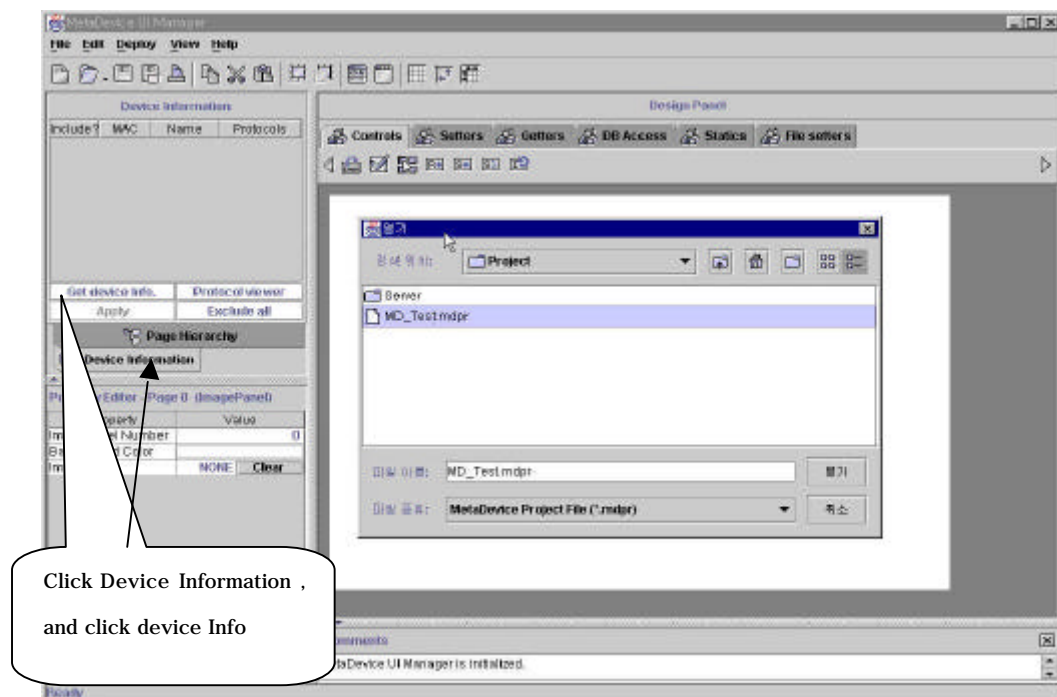


- Using MetaDevice UI Manager

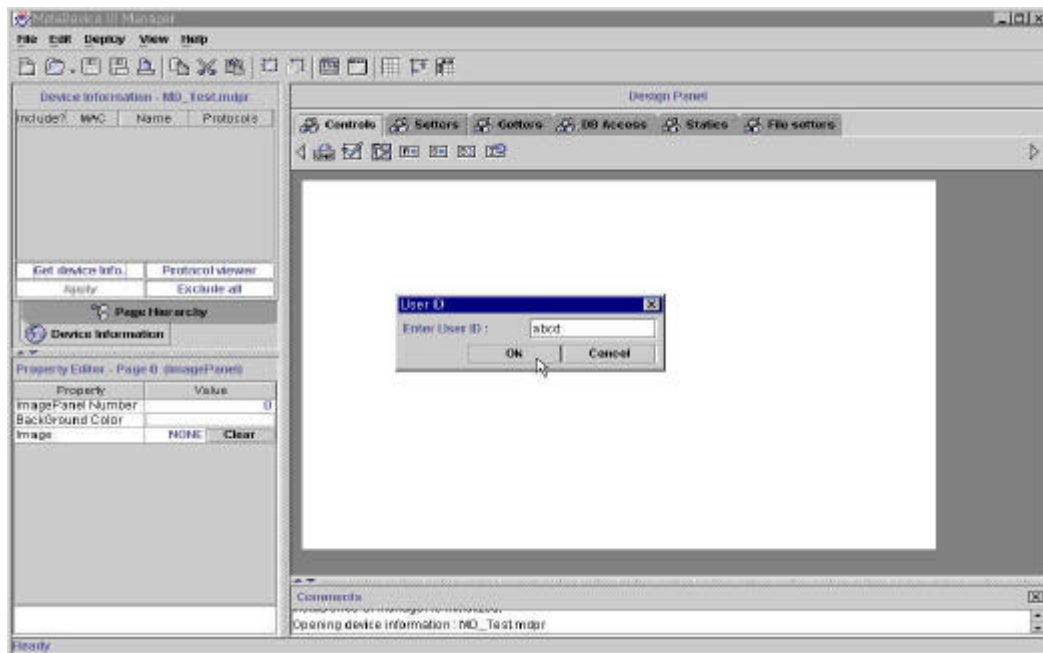
(1) Double click the UI Manager ICON to open UI Manager.



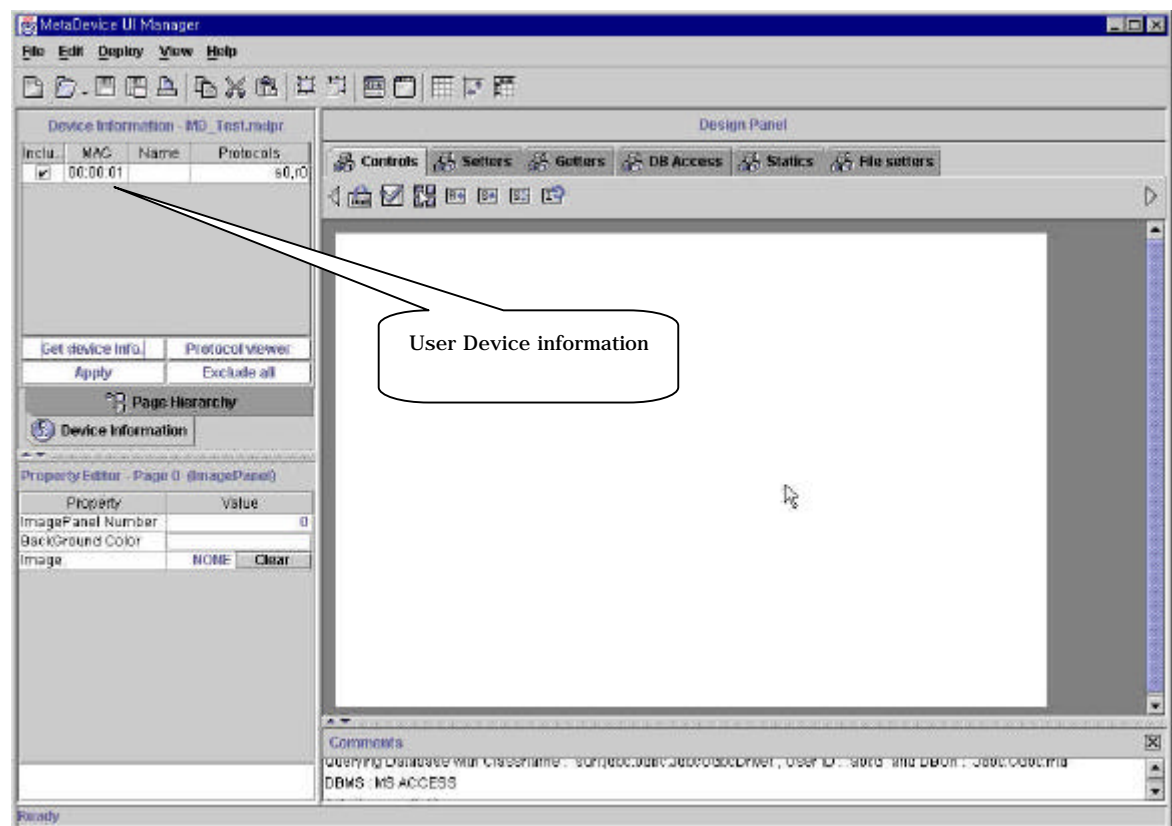
(2) Select Get device Info. Tab from the Device Information window to bring the MD_Test.mdpr file saved in the Server Manager.



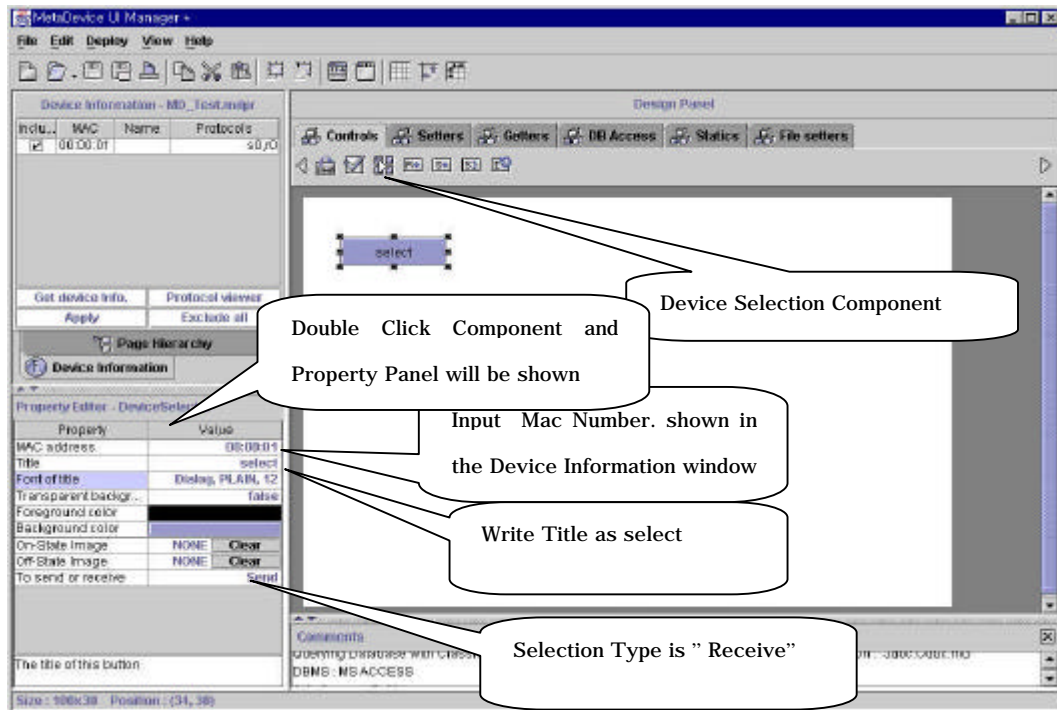
(3) Input User ID registered in the DB when importing MDPR File.



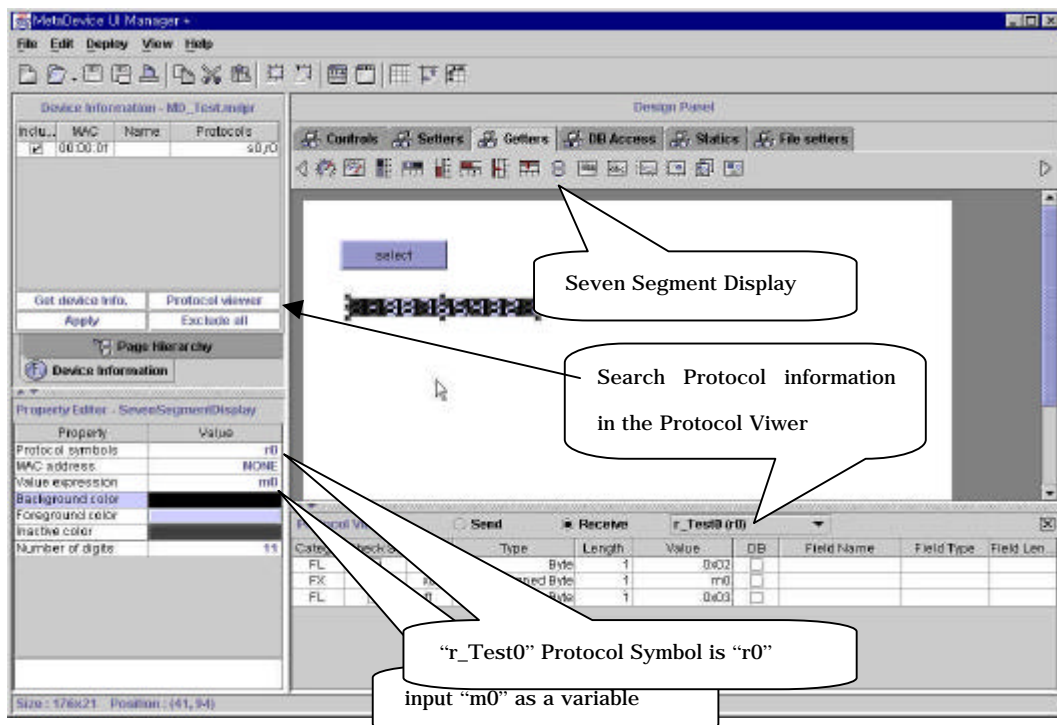
(4) Device Information related to abcd User can be seen from the Device Information window.



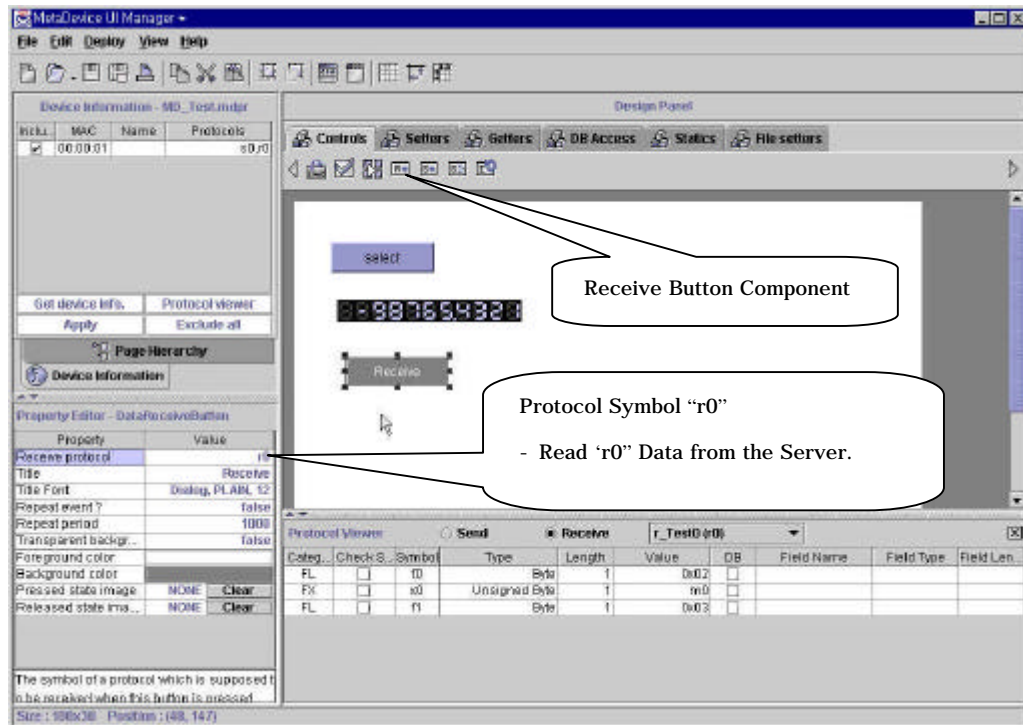
(5) Input Device Selection Component in the Design Panel.



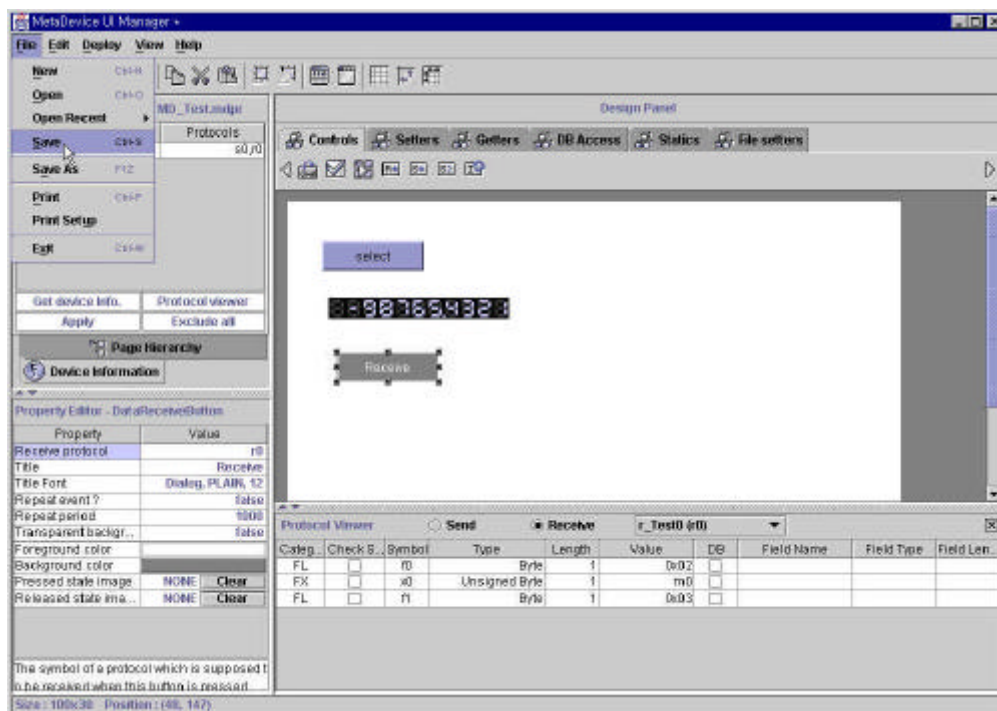
(6) Select Seven Segment Displayer from the Getters Component Group to represent variable "m0" of r_Test0 Protocol generated from the Server Manager.



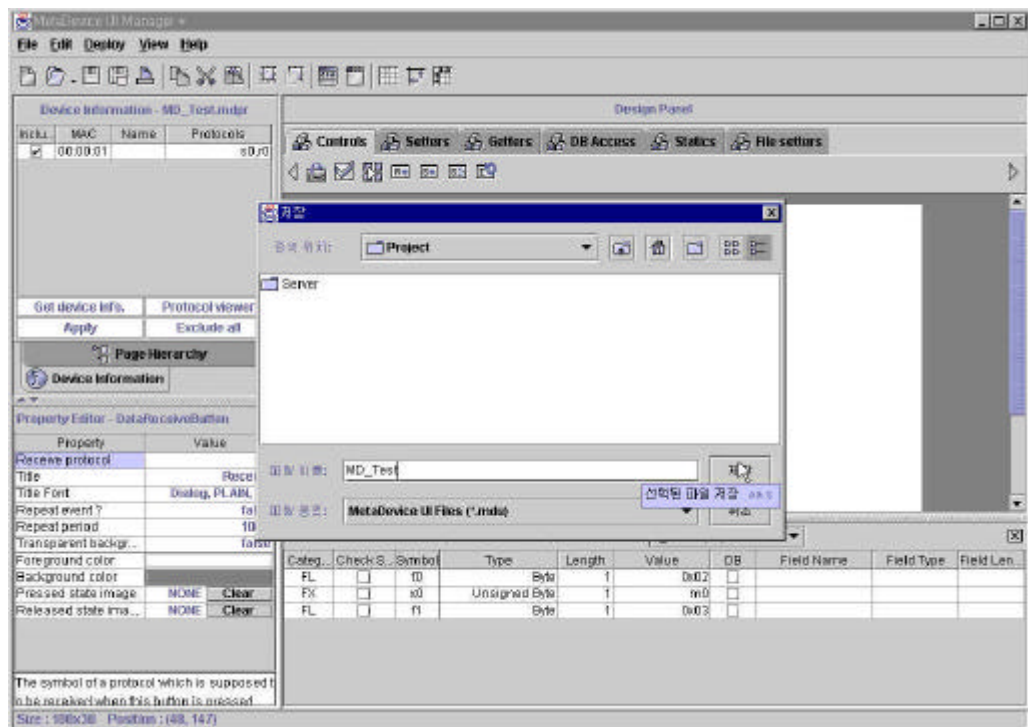
(7) Input Receive Button Component to read data from the Server.



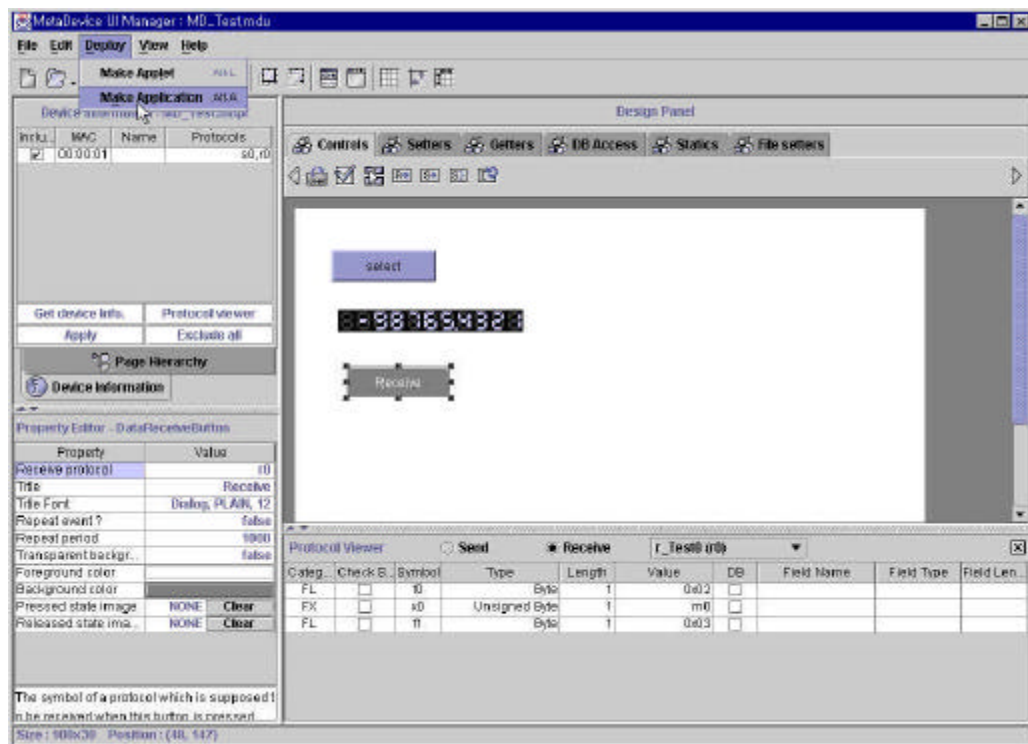
(8) Save MDU file



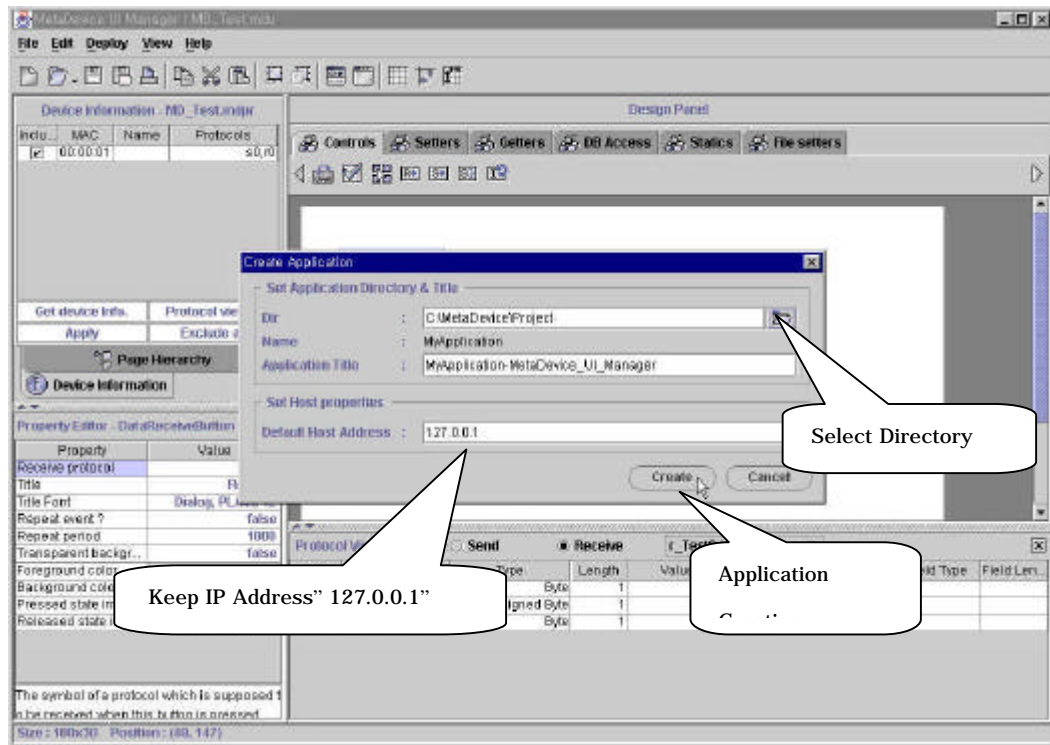
(9) Register File Name (MD_Test)



(10) After saving MDU File, deploy Application.



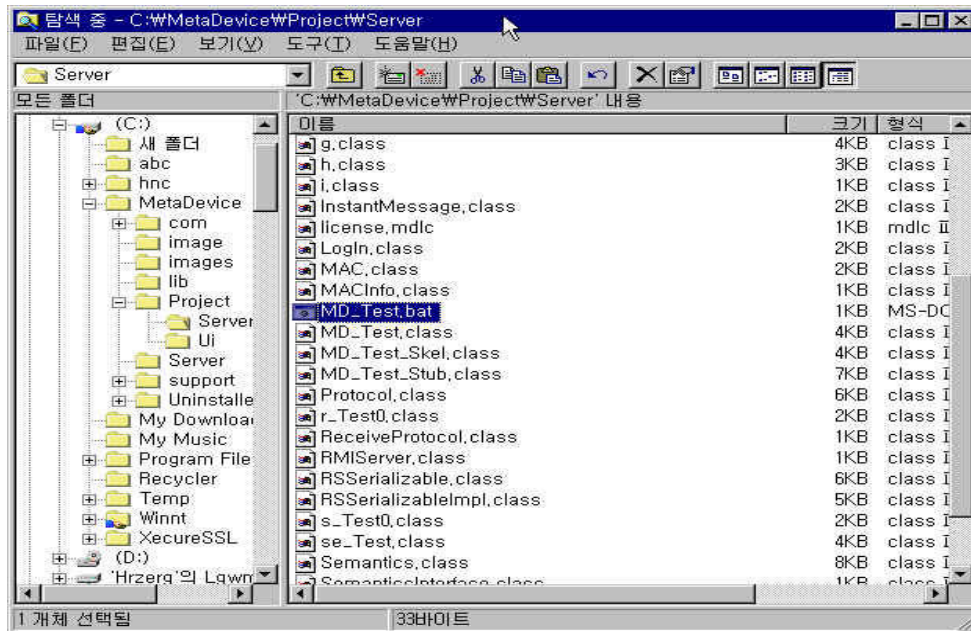
(11) Set the Deploy Directory



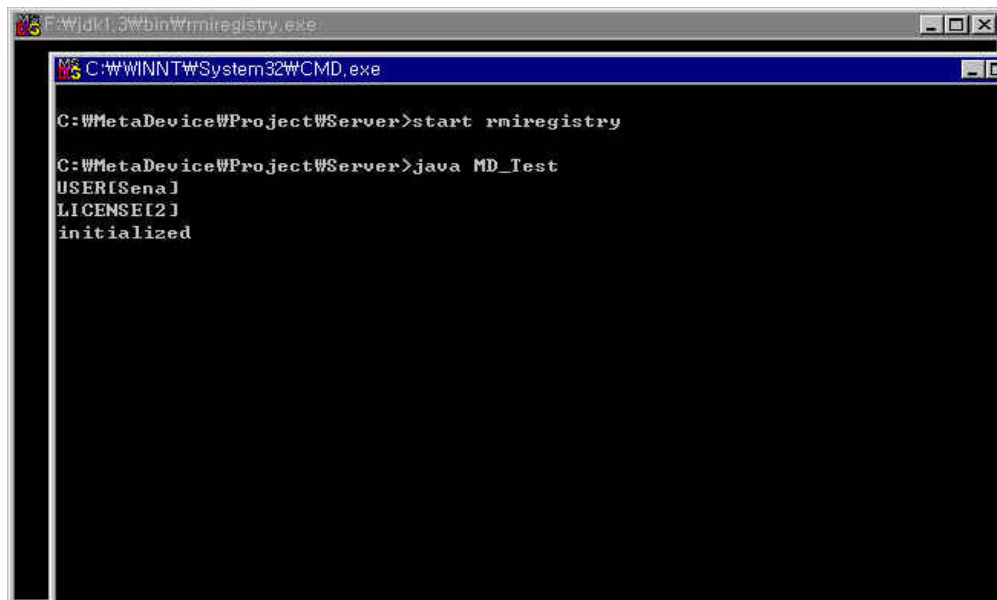
(12) All procedures have been completed. End system from the File Menu.

- Running Server Run-Time Program

(1) execute MD_Test.bat file. (Project\Server\MD_Test.bat)

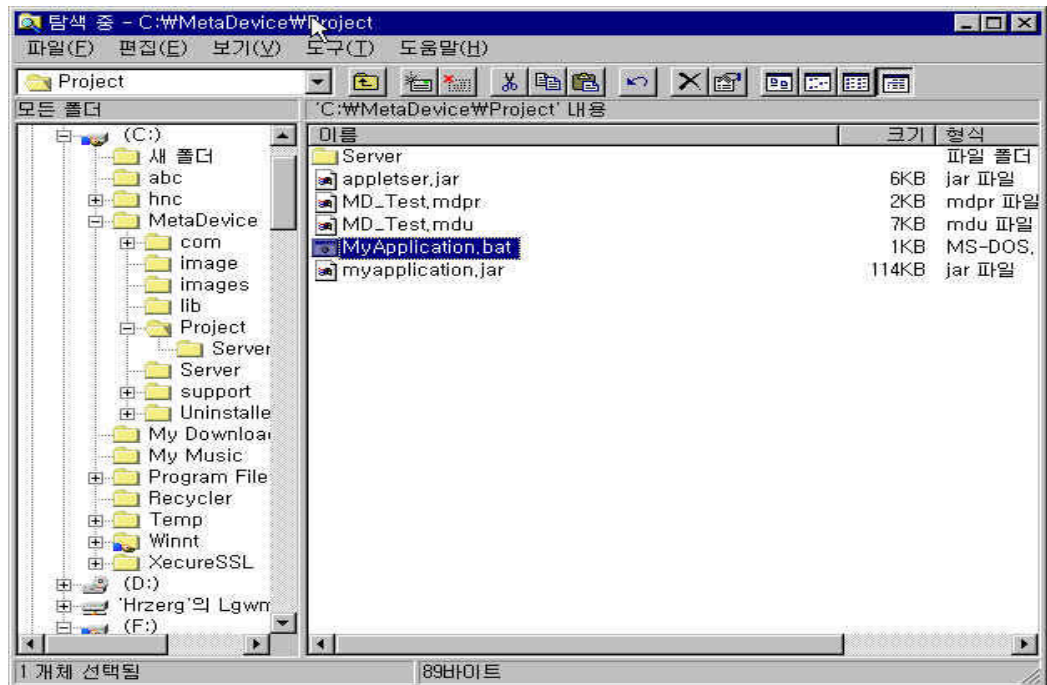


(2) If the following screen should appear when running Server.

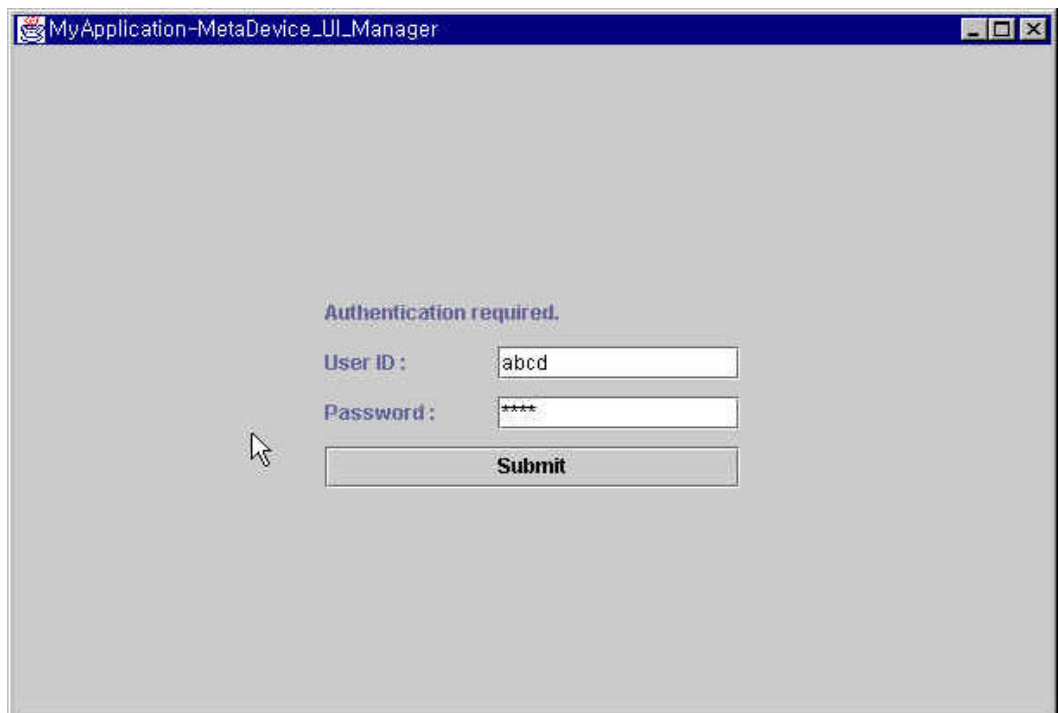


- Running Client UI Run-Time Program

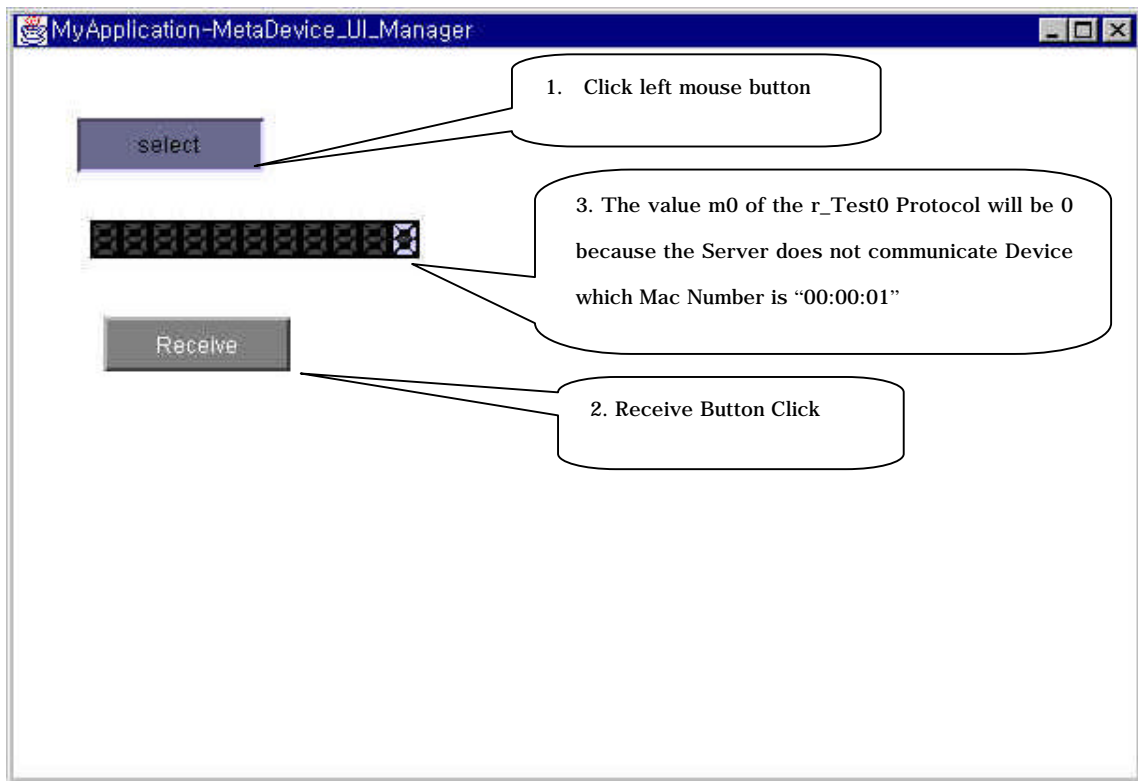
(1) execute MyApplication.bat file.



(2) Input User ID and Password (abcd/1234) saved in the database.



(3) Check the result.

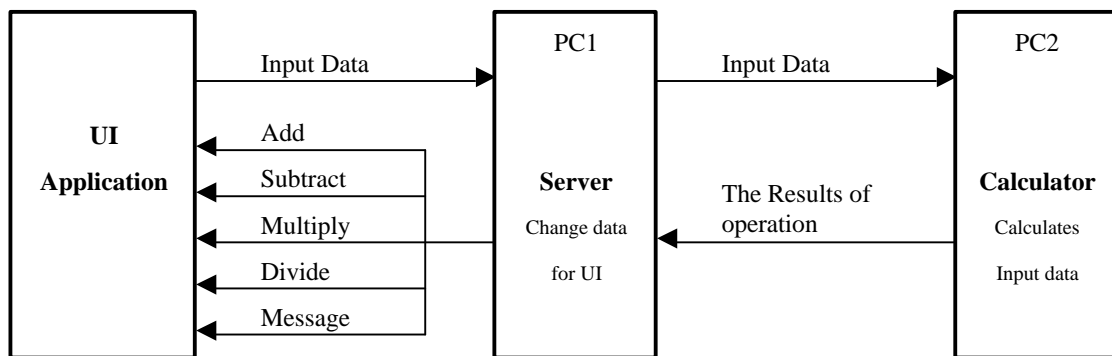


2. Arithmetic Operation Example

The following example would be of help in applying what we have learned so far. Understanding Project Creation, Protocol Editing, Semantics, Transfer Editing of the Server Manager and UI Application editing of the UI Manager will allow for a complete experience of the MetaDevice.

2.1 Project Structure

In the following example, after the user inputs two numbers in the UI Application, the data is sent to the Server and subsequently to the Calculator. The Calculator sends the results of the operation to the Server and the Server changes the data fit for UI Application. The UI Application then brings the changed data from the Server to display it on the graphic window.



[Fig 4.1.1 Project Architecture]

1. UI Application

- 1) User inputs two numbers on the UI.
- 2) UI receives operation results from Server.
- 3) UI displays the data.

2. Server Program (PC1)

- 1) Sends data received from the UI Application to Calculator.
- 2) Receives data results from Calculator.
- 3) Changes data received from Calculator to UI Application component data (Adding Results, Subtracting Results, Multiplying Results, Dividing Results, Message)
- 4) Stores results in Server Program memory.

3. Calculator (PC2)

- 1) Receives data from Server.
- 2) Sends results of calculation to Server.
- 3) This calculator PC has a User Device role.
- 4) This calculator is also used instead of TCP/IP Devices.

4. Input data

- 1) Total length : 18 bytes
- 2) Format

1	2 ~ 9	10 ~ 17	18
STX	Operand1	Operand2	ETX

- 3) STX
 - TYPE : byte
 - LENGTH : 1byte
 - VALUE : 0x02
- 4) Operand1
 - TYPE : double
 - LENGTH : 8byte
 - VALUE : Double type variable value of the user's input in UI Application
- 5) Operand2
 - Equal Operand1
- 6) ETX
 - TYPE : byte
 - LENGTH : 1byte
 - VALUE : 0x02

5. The results of operations

- 1) Total length : 35 byte
- 2) Format

1	2 ~ 9	10 ~ 17	18 ~ 25	26 ~ 33	34	35
STX	Add	Subtract	Multiply	Divide	Error	ETX

- 3) STX : Equal to the STX of the input data
- 4) Add : TYPE – double , LENGTH – 8byte , VALUE – Operand1 + Operand2
- 5) Subtract : TYPE – double , LENGTH – 8byte , VALUE – Operand1 - Operand2

- 6) Multiply : TYPE – double , LENGTH – 8byte , VALUE – Operand1 * Operand2
- 7) Divide : TYPE – double , LENGTH – 8byte
If operand 2 = 0 , VALUE – 0 , or not , VALUE – Operand1 / Operand2
- 8) Error : TYPE – unsigned byte , LENGTH – 1byte
If operand 2 = 0 , VALUE – 1 , or not, VALUE – 0
- 9) ETX : Equal to the ETX of the input data

6. Adding results

- 1) Total length : 28 byte
- 2) Format

1	2 ~ 9	10	11 ~ 18	19	20 ~ 27	28
STX	Operand 1	AddSign	Operand2	EqualSign	Add	ETX

- 3) STX : Equals the STX of the input data
- 4) Operand1 : TYPE – double , LENGTH – 8byte first operand
- 5) AddSign : TYPE – text , LENGTH – 1byte , VALUE – “+”
- 6) Operand1 : TYPE – double , LENGTH – 8byte second operand
- 7) EqualSign : TYPE – text , LENGTH – 1byte , VALUE – “=”
- 8) Add : Adding results
- 9) ETX : Equals the ETX of the input data

7. Subtracting Results

- 1) Total length : 28 byte
- 2) Format

1	2 ~ 9	10	11 ~ 18	19	20 ~ 27	28
STX	Operand 1	SubtractSign	Operand2	EqualSign	Subtract	ETX

- 3) STX : Equals the STX of the input data
- 4) Operand1 : TYPE – double , LENGTH – 8byte first operand
- 5) SubtractSign : TYPE – text , LENGTH – 1byte , VALUE – “-”
- 6) Operand1 : TYPE – double , LENGTH – 8byte second operand
- 7) EqualSign : TYPE – text , LENGTH – 1byte , VALUE – “=”
- 8) Subtract : Subtracting Results
- 9) ETX : Equals the ETX of the input data

8. Multiplying Results

- 1) Total length : 28 byte

2) Format

1	2 ~ 9	10	11 ~ 18	19	20 ~ 27	28
STX	Operand 1	MultiplySign	Operand2	EqualSign	Multiply	ETX

- 3) STX : Equals the STX of the input data
- 4) Operand1 : TYPE – double , LENGTH – 8byte first operand
- 5) MultiplySign : TYPE – text , LENGTH – 1byte , VALUE – “*”
- 6) Operand1 : TYPE – double , LENGTH – 8byte second operand
- 7) EqualSign : TYPE – text , LENGTH – 1byte , VALUE – “=”
- 8) Multiply : Multiplying Results
- 9) ETX : Equals the ETX of the input data

9. Division Results

- 1) Total length : 28 byte
- 2) Format

1	2 ~ 9	10	11 ~ 18	19	20 ~ 27	28
STX	Operand 1	DivideSign	Operand2	EqualSign	Divide	ETX

- 3) STX : Equals the STX of the input data
- 4) Operand1 : TYPE – double , LENGTH – 8byte first operand
- 5) DivideSign : TYPE – text , LENGTH – 1byte , VALUE – “/”
- 6) Operand1 : TYPE – double , LENGTH – 8byte second operand
- 7) EqualSign : TYPE – text , LENGTH – 1byte , VALUE – “=”
- 8) Divide : Division Results
- 9) ETX : Equals the ETX of the input data

10. Message

- 1) Total length : variable length
- 2) Format

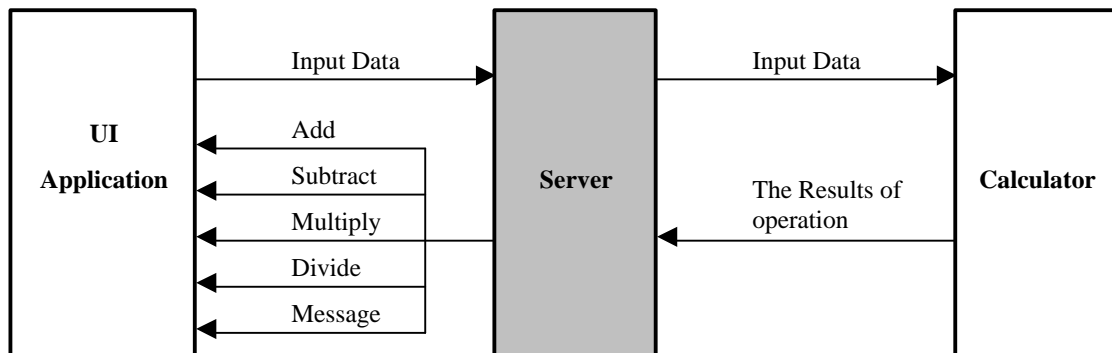
1	2 ~ n	n+1
STX	Message	ETX

- 3) STX : Equals the STX of the input data
- 4) Message : The Message which is sent to UI Program
 - Normal : The results of operation is Error = 0
 - Dividing 0 : The results of operation is Error = 1
 - When the results do not come from the Calculator

- When the format results are different from those from the Calculator.
- 5) ETX : Equals the ETX of the input data

2.2 Server Implementation

The Server receives data from the UI Application to send it to the Calculator, and also receives results from the Calculator to transfer the data fit for UI Application display.



[Fig 4.1.2 Project Structure from Server Viewpoint]

1. General Property

- 1) Project Name : TheServer
Create Project with name as TheServer.
- 2) Location : D:\Manual\ThisIsServer
Create Project file under D:\Manual\ThisIsServer.
- 3) Daemon Type : Client
Server Daemon Type is Client because the Server sends and receives data first to and from the Calculator, which acts as a Device.
- 4) DBMS : MS ACCESS
Use MS ACCESS.
- 5) Class Name : sun.jdbc.odbc.JdbcOdbcDriver
Use JdbcOdbcDriver.
- 6) URL : Jdbc:Odbc:server
Use ODBC DSN as server.
- 7) User ID, Password : Don't input the value.

2. Send Protocol

- 1) Send Protocol formats data that is to be sent to the Device.
- 2) Set data to be sent to the Calculator, assuming the role of the device, as Send Protocol.
- 3) Define Send Protocol name as "SPData".

3. Receive Protocol

- 1) The Receive Protocol formats data needed by the UI Application or the data received from the Device(Calculator).
- 2) Define Receive Protocol which is received from the Device(Calculator) as Receive Protocol: "RPSResult".
- 3) Define Adding, Subtracting, Multiplying, Dividing, Message data displayed by UI Application as Receive Protocols such as RPAdd, RPSubtract, RPMultiply, RPDivide, or RPMessage.

4. Transfer

- 1) Implement RPSResult data received from the Device(Calculator) to those required by the UI Application such as RPAdd, RPSubtract, RPMultiply, RPDivide, or RPMessage and define Transfer as TRResult.
- 2) If RPSResult is not transferred from Device(Calculator), implement TRNoData Transfer that displays in the RPMessage Protocol that operation results have not been transferred.
- 3) If RPSResult transferred from Device(Calculator) does not fit Protocol format, implement TRParseError Transfer that displays in the RPMessage that Protocol format is different.
- 4) In case of other errors, implement TRUnknownError Transfer that will be displayed in RPMessage. But this process does not occur in actuality.

5. Semantics

Implement SETheServer Semantics to have the Server and Calculator communicate with each other and transfer the RPSResult received as RPAdd, Rpsubtract, RPMultiply, RPDivide, RPMessage, required by the UI Application.

6. Implementation Environment

- 1) OS – Windows 98
- 2) JDK – jdk1.3.1_02 installed
- 3) MS ACCESS – Microsoft Access 2000 installed
- 4) IP Address – 192.168.0.32

7. Creating Work Folder

- 1) Create Manual folder in the D:\.
- 2) Create ThisIsServer folder in D:\Manual.
- 3) Check whether D:\Manual\ThisIsServer folder has been created.

8. Setting ODBC DSN

- 1) Select [Start / Settings / Control Panel] to open the [Control Panel].
- 2) Open the [ODBC DSN Administrator] by double clicking [ODBC Data Source (32bit)] in the [Control Panel] window.
- 3) Display the list of [System Data Source] by selecting the [System DSN] Tab in the [ODBC DSN Administrator] window. (Can register in the User DSN.)
- 4) Click [Add] to open [CreateNew Data Source] Dialog.
- 5) Click [OK] Button after selecting [Driver do Microsoft Access (*.mdb)] from the [Driver Selection] list of the [Create New Data Source] dialog window.
- 6) Type "server" in the text box of [Data Source Name] when the [Setup ODBC Microsoft Access] window opens, and Click [Create].
- 7) Select [d:\Manual\ThisIsServer] folder from the [New DataBase] window.
- 8) Type "server.mdb" in the [DataBase Name] text box and click [OK].
- 9) Click [Ok] in the [Setup ODBC Microsoft Access] window .

9. Creating Project

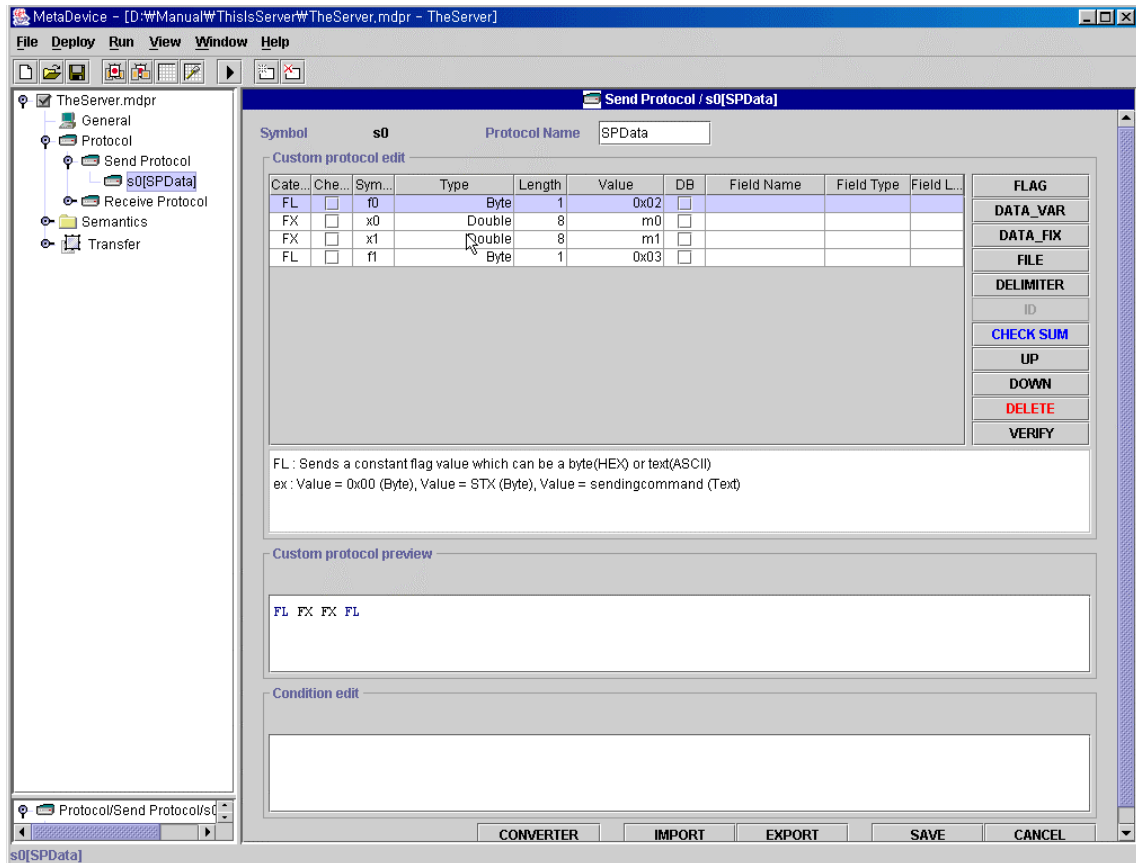
- 1) Run Server Manager.
- 2) Select [File/New] to open [New Project] dialog window.
- 3) Set general properties from the [New Project] dialog window.
 - Project Name : TheServer
 - Location : D:\Manual\ThisIsServer (select Location)
 - Daemon Type : Client (select Client radio button)
 - DBMS : MS ACCESS (select MS ACCESS radio button)
 - Class Name : sun.jdbc.odbc.JdbcOdbcDriver
 - URL : Jdbc:Odbc:server
- 4) Click [SAVE] button to create [TheServer] Project.

10. Creating Send Protocol

- 1) Click the [Send Protocol] node under [Project] node of [Project Pane] in the upper left corner of the screen.
- 2) Open [New Serial Protocol] by clicking [Append] button under [Send Protocol List Panel] from the [Content Panel] in the upper right corner of the screen.
- 3) Type "SPData" in the [Protocol Name] text box and click [OK] to create "SPData" Protocol.

11. Editing Send Protocol

- 1) Double click [SPData] Protocol of the [Send Protocol List Panel] to open [Send Protocol Edit Panel] .
- 2) Edit as follows from the [Send Protocol Edit Panel].



[Fig 4.2 Editing Server SPData Protocol]

- 3) [SPData] Protocol is the data sent to the Calculator and its value is inputted from the UI Application.
- 4) FIXED DATA Type “m0” is Operand1 and FIXED DATA Type “m1” is Operand 2.

12. Creating Receive Protocol

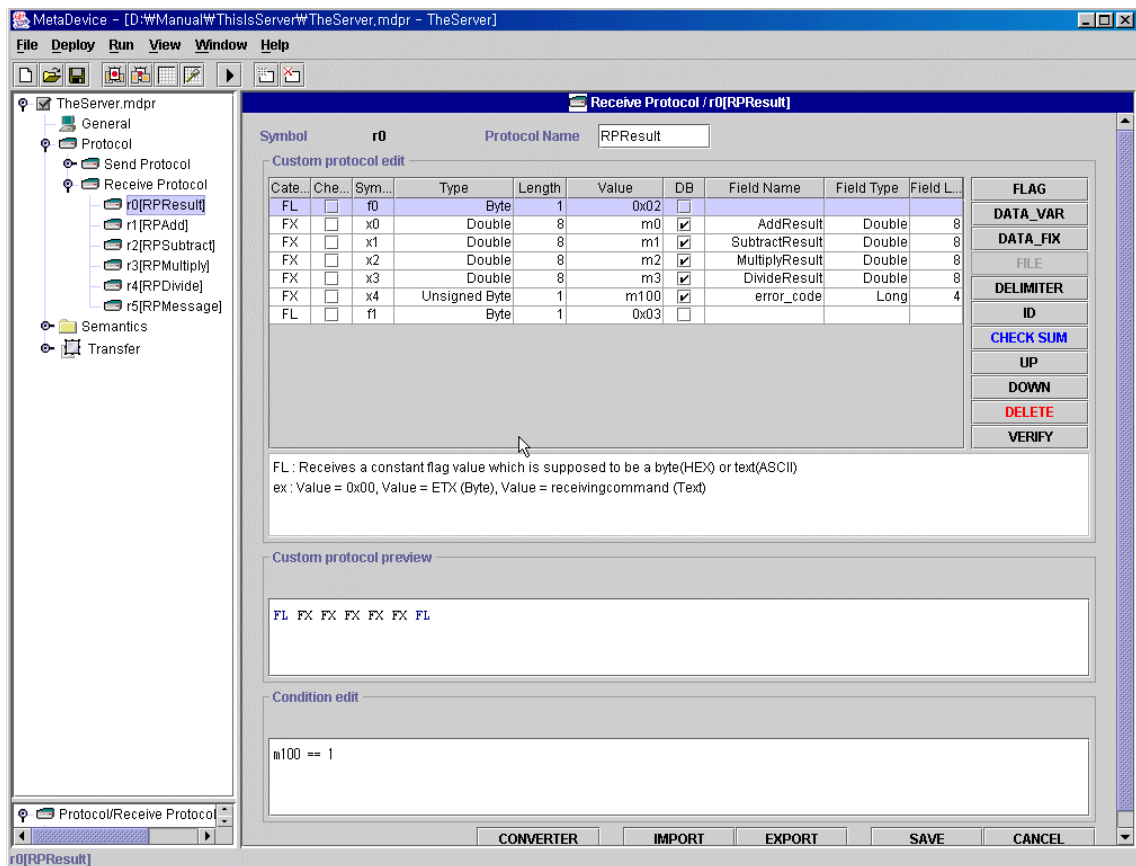
- 1) Click [Receive Protocol] node under [Project] of the [Project Panel] in the upper left corner of the screen.
- 2) Open [New Serial Protocol] dialog window by clicking [Append] under the [Receive Protocol List Panel] of the [Content Panel] in the upper right corner of the screen.
- 3) Type “RPRResult” in the [Protocol Name] text box of the [New Serial Protocol] dialog window

and click [OK] to create [RResult] Protocol.

- 4) Repeat procedure 2) ~ 3) five times with only the Protocol Name different. The other Protocol Names are “RPAdd, RPSubtract, RPMultiply, RPDivide, RPMessage”

13. Editing RResult Receive Protocol

- 1) Double click the “RResult” Protocol to open [Receive Protocol Edit Panel].
- 2) Edit as follows from the [Receive Protocol Edit Panel].



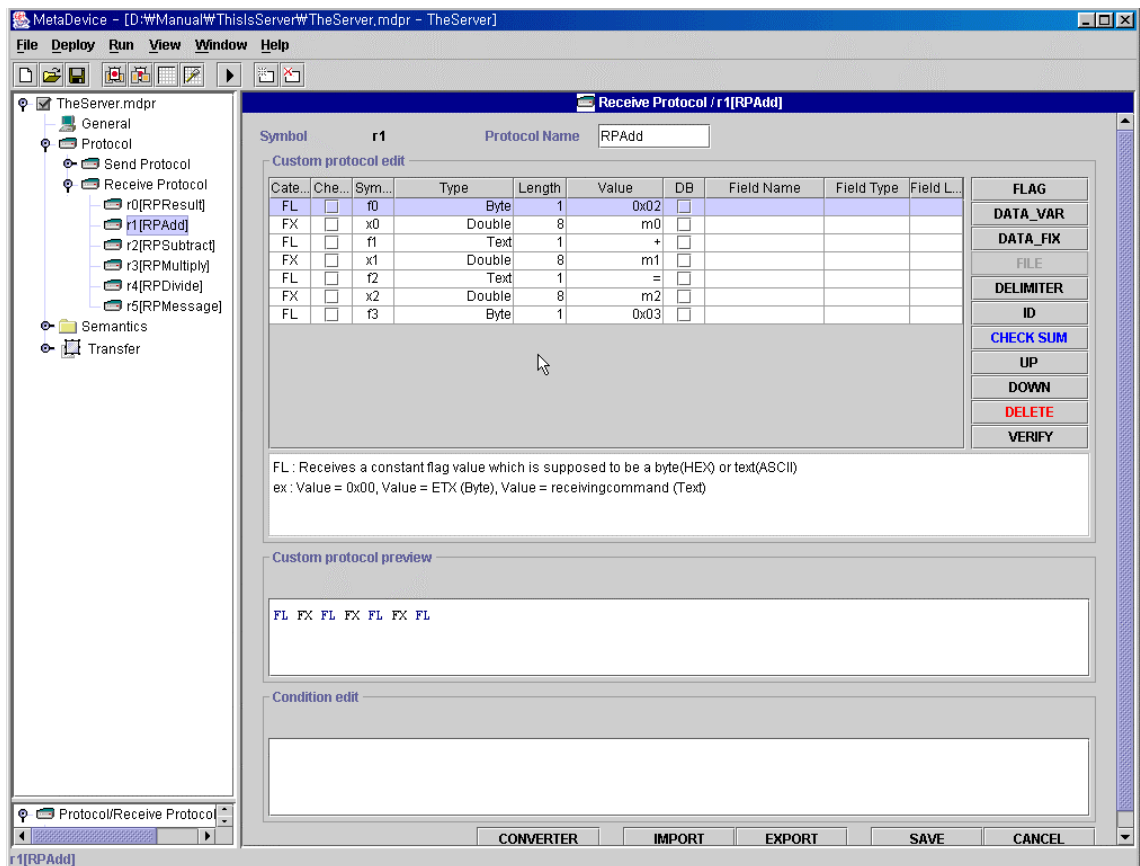
[Fit 4.3 Editing Server RResult Protocol]

- 3) The [RResult] Protocol is the operation result data transferred from the Calculator. As required by UI Application, the protocol is changed to RPAdd, RPSubtract, RPMultiply, RPDivide, and RPMessage.
- 4) FIXED DATA Type “m0” show adding results. FIXED DATA Type “m1” show subtracting results. FIXED DATA Type “m2” shows multiplying results, and FIXED DATA Type “m3” shows dividing results. FIXED DATA Type “m100” is in its normal state when it is 0, and when it is 1, [SPData] Operand2 is 0 and Operand1 is divided by 0.
- 5) To store the protocol data to the DB, check the [DB] edit field and input the [Field Name], [Field Type], [Field Length] in the [Protocol Edit Table].

- 6) Input [m100 == 1] in the [DB Storage Condition Edit] window. Allow data storage only in cases when the [RPSResult] [m100] is 1 (Operand2 is 0). Otherwise, all data will be saved.

14. Editing RPAAdd Receive Protocol

- 1) Double click [RPAAdd] Protocol from the [Receive Protocol List Panel] to open the [Receive Protocol Edit Panel].
- 2) Edit the RPAAdd Protocol as follows.

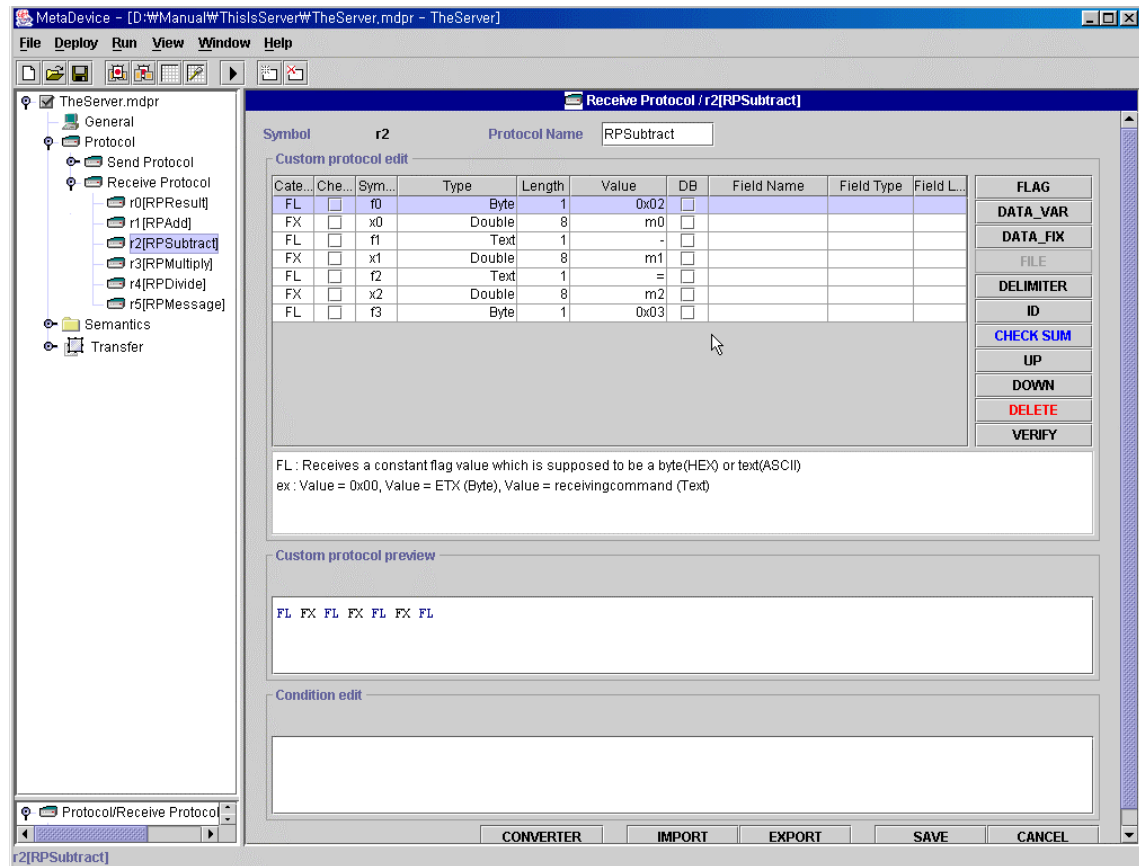


[Fig 4.4 Editing Server RPAAdd Protocol]

- 3) The [RPAAdd] Protocol is one of the changed protocols from the [RPSResults] to display operation results in the UI Application. It is periodically taken by the UI Application to be displayed on screen.
- 4) FIXED DATA Type "m0" is [SPData] Operand1 m0 value. FIXED DATA Type "m1" is [SPData] Operand2 m1. FIXED DATA Type "m2" shows adding results.
- 5) Edit [RPSubtract], [RPMultiply], [RPDive] Protocol by repeating the RPAAdd Protocol Editing method. Each "m2" are the results of adding, subtracting, multiplying, and dividing. Export

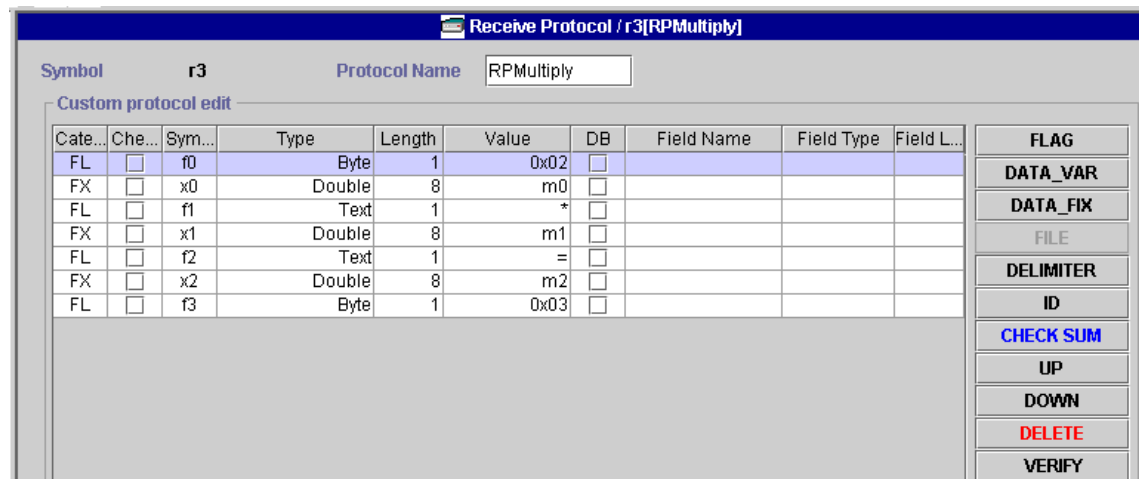
[RPAdd] Protocol and import [RPAdd] to easily edit other protocols.

15. Editing RPSubtract Receive Protocol



[Fig 4.5 Editing Server RPSubtract]

16. Editing RPMultiply Receive Protocol



[Fig 4.6 Editing Server RPMultiply]

17. Editing RPDivide Receive Protocol

Cate...	Che...	Sym...	Type	Length	Value	DB	Field Name	Field Type	Field L...	FLAG
FL	<input type="checkbox"/>	f0	Byte	1	0x02	<input type="checkbox"/>				DATA_VAR
FX	<input type="checkbox"/>	x0	Double	8	m0	<input type="checkbox"/>				DATA_FIX
FL	<input type="checkbox"/>	f1	Text	1	/	<input type="checkbox"/>				FILE
FX	<input type="checkbox"/>	x1	Double	8	m1	<input type="checkbox"/>				DELIMITER
FL	<input type="checkbox"/>	f2	Text	1	=	<input type="checkbox"/>				ID
FX	<input type="checkbox"/>	x2	Double	8	m2	<input type="checkbox"/>				CHECK SUM
FL	<input type="checkbox"/>	f3	Byte	1	0x03	<input type="checkbox"/>				UP
										DOWN
										DELETE
										VERIFY

[Fig 4.7 Editing Server RPDivide]

18. Editing RPMessage Receive Protocol

- 1) Double click [RPMessage] Protocol to open [Receive Protocol Edit Panel].
- 2) Edit [RPMessage] Protocol as follows.

Cate...	Che...	Sym...	Type	Length	Value	DB	Field Name	Field Type	Field L...	FLAG
FL	<input type="checkbox"/>	f0	Byte	1	0x02	<input type="checkbox"/>				DATA_VAR
VA	<input type="checkbox"/>	v0	Byte Array	f1	v0	<input type="checkbox"/>				DATA_FIX
FL	<input type="checkbox"/>	f1	Byte	1	0x03	<input type="checkbox"/>				FILE
										DELIMITER
										ID
										CHECK SUM
										UP
										DOWN
										DELETE
										VERIFY

[Fig 4.8 Editing Server RPMessage]

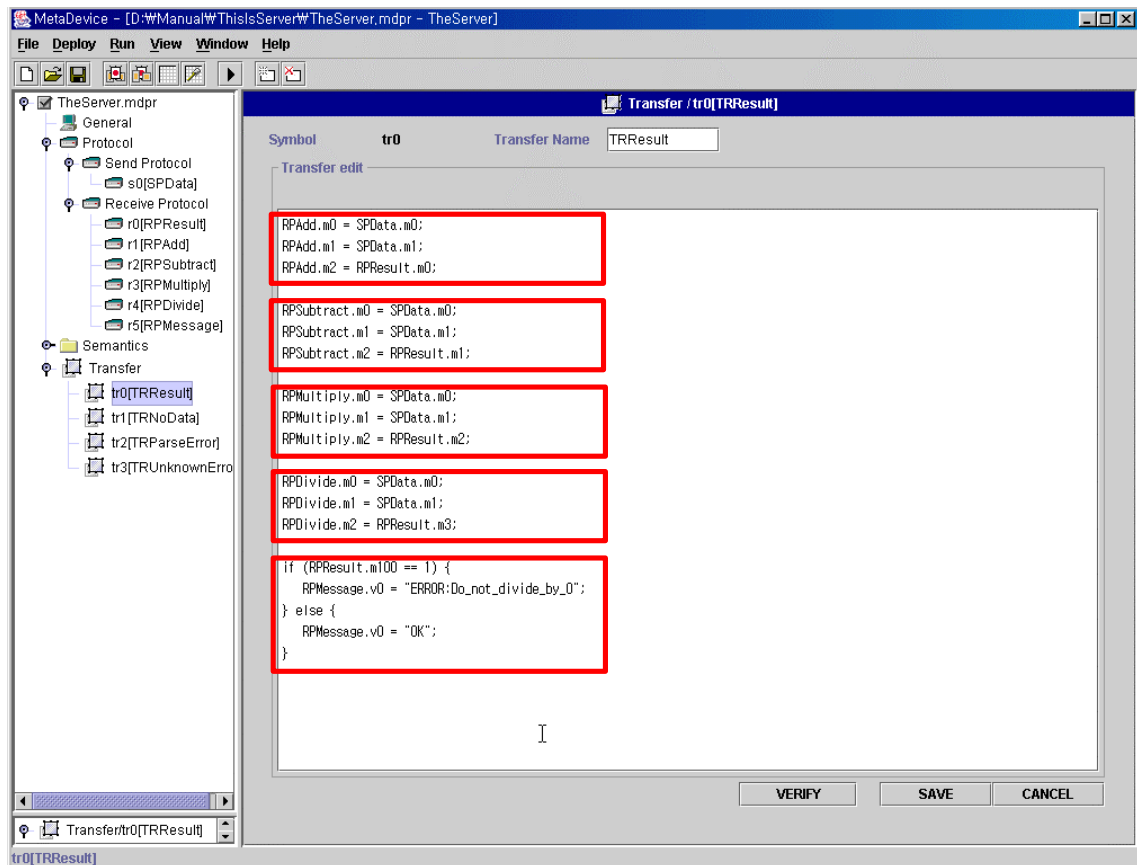
- 3) The Message from the Calculator is changed to [RPMessage] Protocol to display in the UI Application (Normal State or 0 dividing error). Also when the Server does not receive data from the Calculator or if the data format is different from the [RPResults] Protocol even though the Server receives the data , the Message is changed to [RPMessage].
- 4) VARIABLE DATA Type "v0" is the message which will be displayed in the UI Application.

19. Creating Transfer

- 1) Click the [Transfer] node in the [Project Panel] in the upper left corner of the screen.
- 2) Open the [New transfer] dialog window by clicking [Append] from the [Transfer List Panel] of [Content Panel] in the upper right corner of the screen.
- 3) Type "TRResult" in the [Transfer Name] text box of [New transfer] and click [OK] to create [TRResult].
- 4) Repeat procedure 2) ~ 3) three times with only switching [Transfer Name]. Transfe Names are TRNoData, TRParseError, TRUnknownError.

20. Editing TRResult Transfer

- 1) Click [TRResult] Transfer in the [Transfer List Panel] to open [Transfer Edit Panel].
- 2) Edit as follows from the [Transfer Edit Panel].



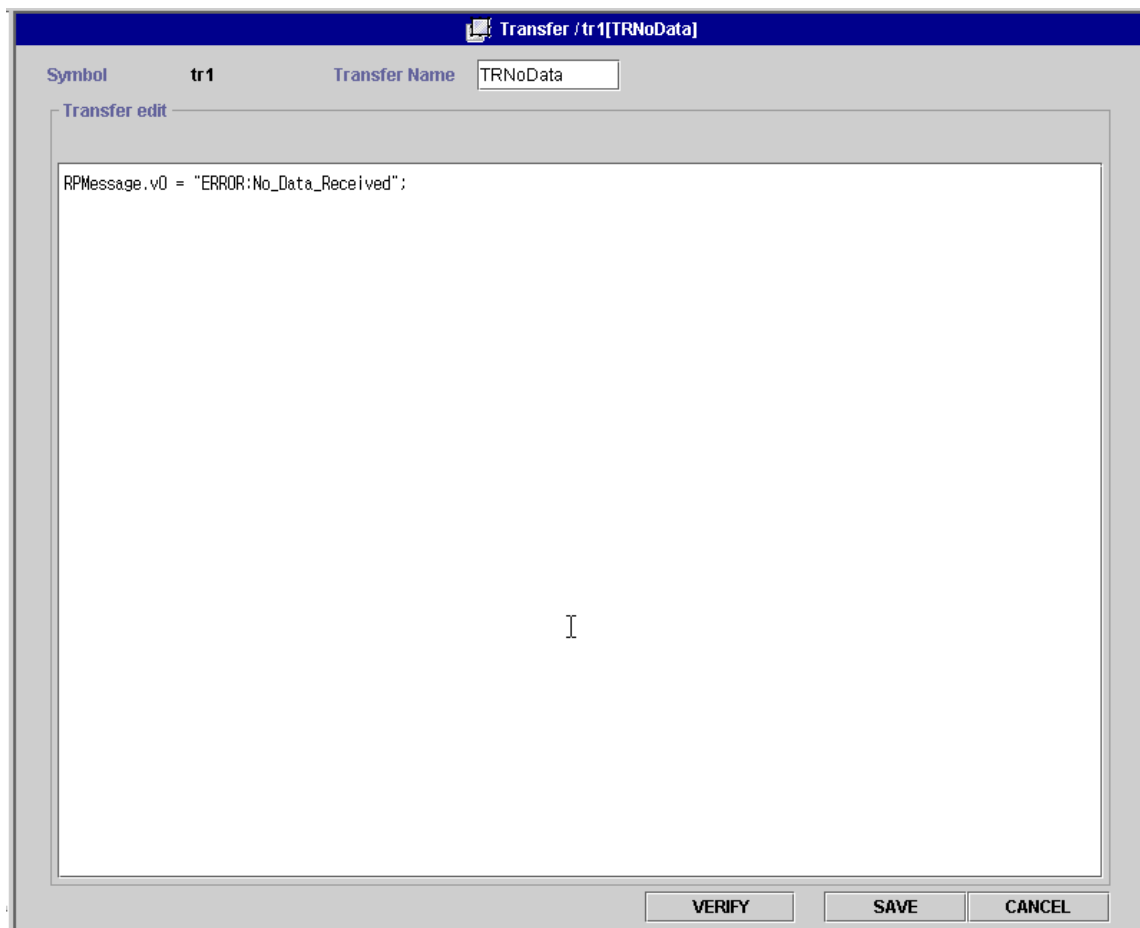
[Fig 4.9 Editing Server TRResult]

- 3) [TRResult] Transfer from the Calculator is changed to RPAAdd, RPSubtract, RPMultiply, RPDivide, RPMMessage Protocol to display in the UI Application.

- 4) The first section is changing to [RPAdd] using [SPData] and [RPResult].
- 5) The second is changing to [RPSubtract] using [SPData] and [RPResult].
- 6) The third is changing to [RPMultiply] using [SPData] and [RPResult].
- 7) The fourth is changing to [RPDivide] using [SPData] and [RPResult].
- 8) The fifth is changing to [RPMessage] using [RPResult]. If [RPResult. m100] is 1, it is changed to Error Message. If 0, it is changed to message displaying normal state. (Notice : Ver1.3 does not support Space " " on the String Constant ,Please use Underscore ("_") instead of Space.)

21. Editing TRNoData Transfer

- 1) Double click [TRNoData] Transfer to open [Transfer Edit Panel].
- 2) Edit as follows from the [Transfer Edit Panel].

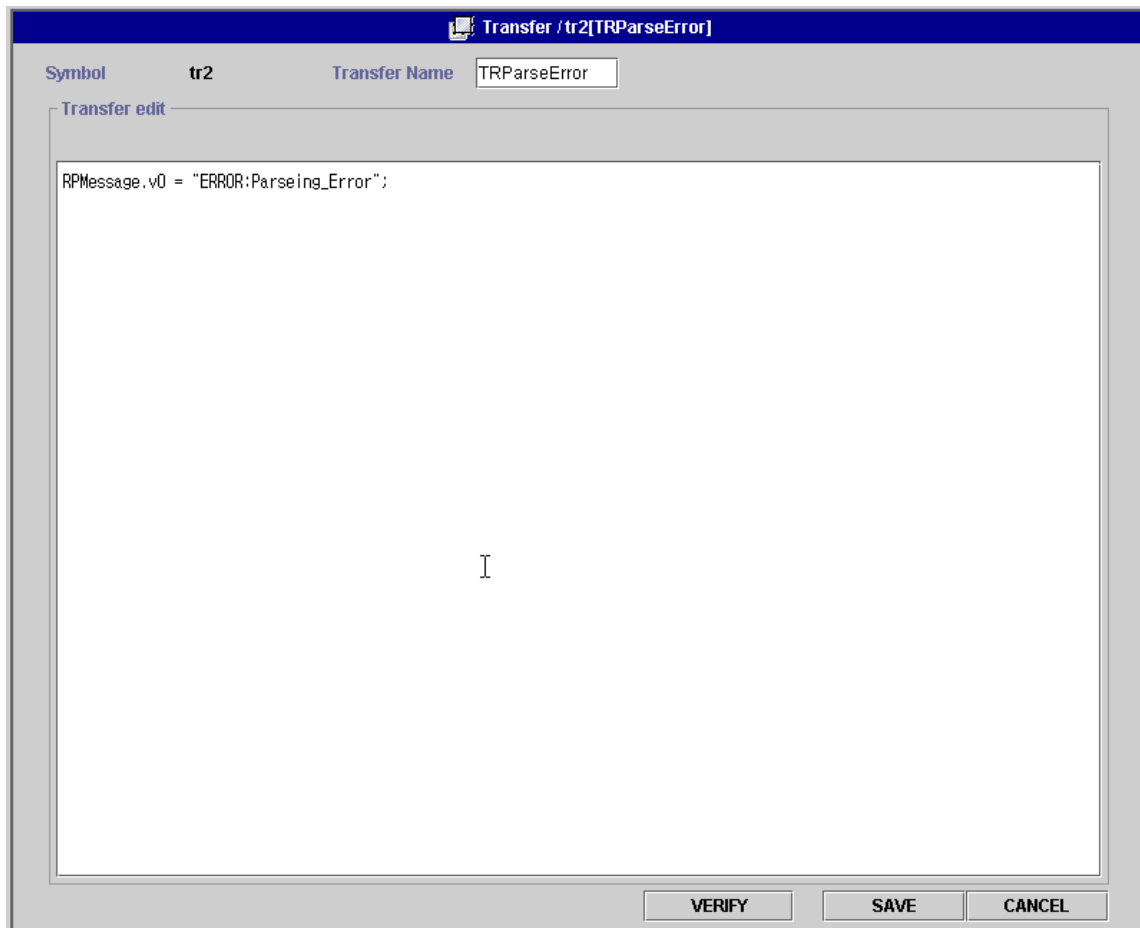


[Fig 4.10 Editing Server TRNoData]

- 3) [TRNoData] Transfer is used to display "ERROR:No_Data_Received" message in the UI Application when it can not receive [RPResult] from the Calculator.

22. Editing TRParseError Transfer

- 1) Double click [TRParseError] Transfer to open [Transfer Edit Panel].
- 2) Edit as follows in the [Transfer Edit Panel].

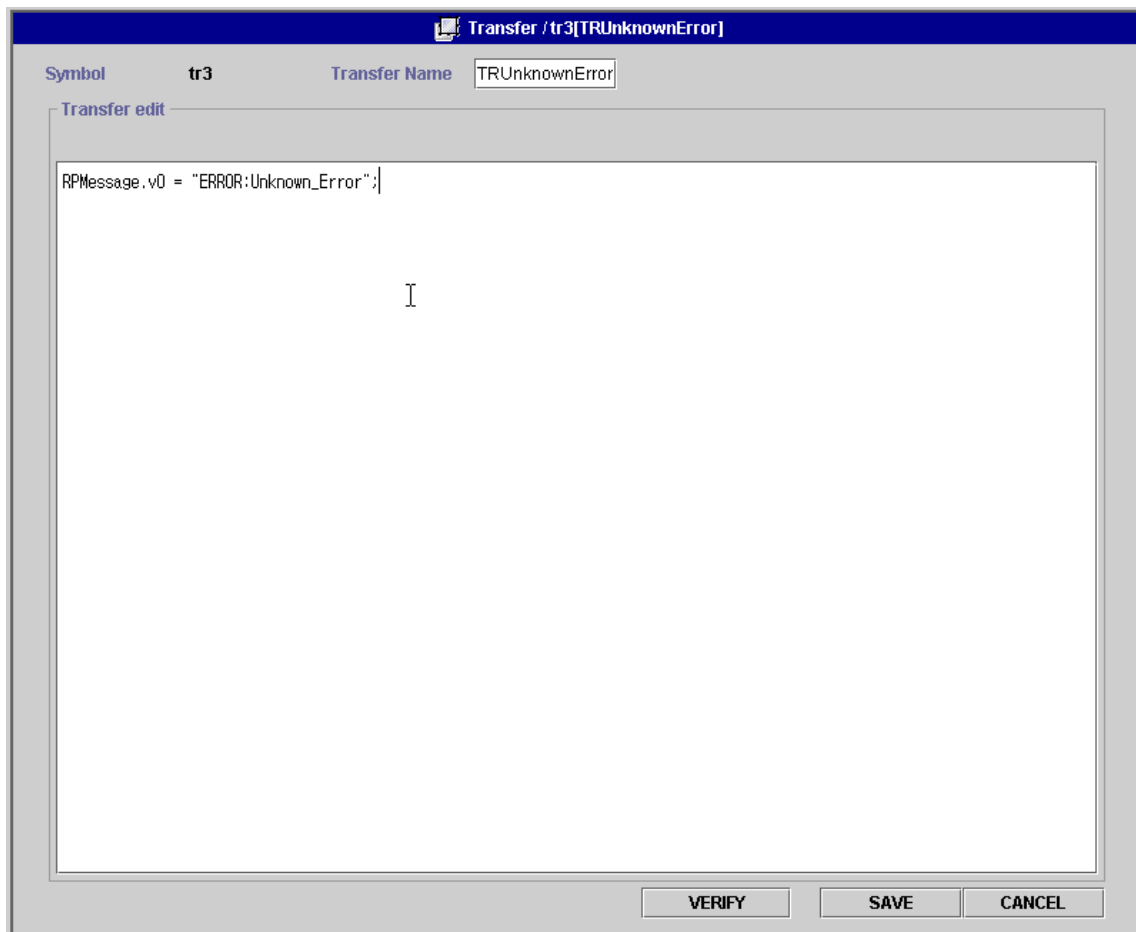


[Fig 4.11 Editing Server TRParseError]

- 4) [TRParseError] Transfer is used to display "ERROR:Parsing Error" message in the UI Application when the data received from the Calculator is different from the [RPResult] format.

23. Editing TRUnknownError Transfer

- 1) Double click [TRUnknownError] Transfer in the [Transfer List Panel] to open the [Transfer Edit Panel].
- 2) Edit as follows from the [Transfer Edit Panel].



[Fig 4.12 Editing Server TRUnknownError]

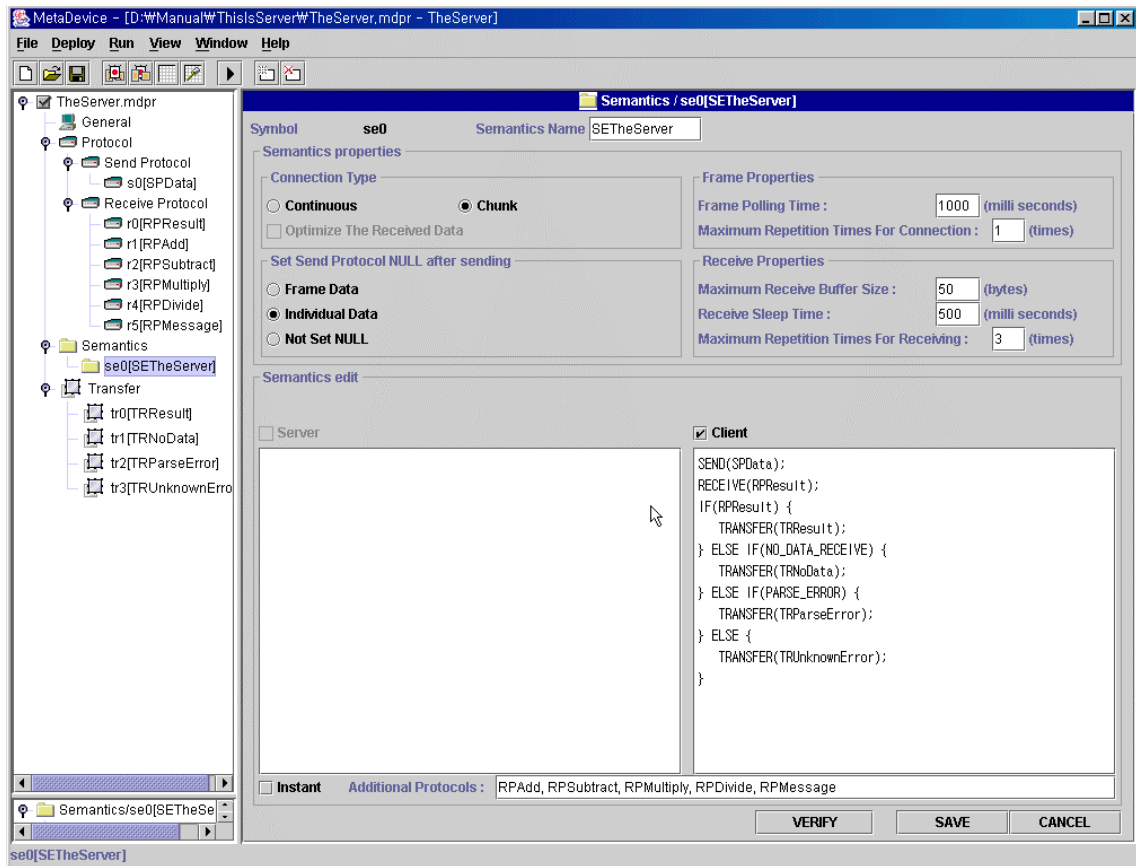
- 5) [TRUnknown Error] Transfer is used to display "ERROR:Unknown_Error" message in the UI Application when other miscellaneous errors occur. One example would be a Check Sum error, but since no Check Sum fields have been used in this sample, such unknown errors will not appear.

24. Creating Semantics

- 1) Click the [Semantics] node in [Project Pane] in the upper left screen.
- 2) Click "Append" from the [Semantics List Panel] under [Content Pane] and open [New semantics] dialog window.
- 3) Type [SETheServer] in the [Semantics Name] text box of the [New semantics] dialog window and click "OK" to create [SETheServer] Semantics.

25. Editing SETheServer Semantics

- 1) Double click [SETheServer] Semantics in the [Semantics List Panel] to open [Semantics Edit Panel].
- 2) Edit Semantics as follows in the [Semantics Edit Panel].



[Fig 4.13 Editing Server SETheServer]

- 3) [Connection Type] is [Chunk]. This means that the Server sends [SPData] to Calculator after connection and disconnects after receiving [RPResult].
- 4) [Set Send Protocol NULL after sending] is [Individual Data]. This means that after sending [SPData] to the Calculator, the Server does not send [SPData] anymore unless new [SPData] is transferred from the UI Application.
- 5) Set [Frame Polling Time] as [1000] to allow the client frame to communicate every second with the Calculator.
- 6) [Maximum Repetition Times For Connection] is 1. The Server will only try to connect with the Calculator once.
- 7) [Maximum Receive Buffer Size] is 50 (more than [RPResult] size). This will allow the Server to create a 50 Byte buffer when it is receiving [RPResult] from the Calculator.
- 8) [Receive Sleep Time] is [500] and [Maximum Repetition Times For Receiving] is [3]. When the Server tries to receive [RPResult] data, this will allow the Server to wait 1.5(500 X 3) Sec

if no data is transferred from the Calculator. The Server will also wait 0.5 sec after receiving data to check if there are any further data to be received. (Note : If [Maximum Repetition Times For Receiving] is set as [1], communication between programs made by Server Managers is impossible. Thus, select a price larger than 1.

- 9) Edit [Client Semantics] in the [Client Semantics Editor].

```

SEND(SPData);                Server connects to Calculator and sends [SPData]
RECEIVE(RPResult);          Server tries to receive [RPResult] from Calculator
IF(RPResult) {                If the data from the Calculator is [RPResult],
    TRANSFER(TRResult);        execute TRResult.
} ELSE IF(NO_DATA_RECEIVE) { If Server does not receive any data from the Calculator
    TRANSFER(TRNoData);       execute TRNoData.
} ELSE IF(PARSE_ERROR) {    If the data from the Calculator is not in [RPResult] format
    TRANSFER(TRParseError);   execute TRParseError.
} ELSE {                      In other cases
    TRANSFER(TRUnknownError); execute TRUnknownError.
}

```

- 10) Protocols [RPAdd], [RPSubtract], [RPMultiply],[RPDivide],[RPMMessage] used in Transfer should be set in the [Additional Protocols] text box. [SPData],[RPResult] Protocols which are used in the SEND(),RECEIVE() method of [Client Semantics Edit] do not have to be set in the [Additional Protocols] text box.

26. Admin

- 1) Click [File / Save] menu and save project file(TheServer.mdpr) in the ServerManager.
- 2) Run Admin Module
Open the TheServer.mdpr created by Server Manager. [File/Open].
- 3) Create Admin Table
Click Admin Module [Deploy / Create Admin Tables] menu and create Admin Table.
Input [admin/admin] for [User ID/PassWd] and click [Login].
- 4) Create Protocol Table
Click Admin Module [Deploy / Create Protocol Tables] menu and create Protocol Table.
- 5) Register User
Click Insert in the User Table and then click Append to create a new Record. Input [jrkim/jrkim] in the [User ID] and [Password] field. Input [JRKim] in the [Name] field. To register the new User, click [Register].
- 6) Register Device
Since the Calculator is a Device from the position of the Server, register Calculator as Device. Select the Device Table and click Insert to create new record by clicking Append. [MAC

Number] field is [00], [00], [01], [Current IP] is [192.168.18.10], [Port No] is [6001], [Semantics] is [SETheServer] and [Device Code] is [-1]. The [Current IP] is Calculator computer IP., In this example the Calculator IP operates in the computer with IP 192.168.18.10 because one system can not run two Server Manager Programs. Click [Register] to register Calculator as Device.

7) Setting Relation

Select the Relation Table to open the Relation window by clicking Insert. Click [User Search] to display the registered users list and click [Device Search] to display the Device list. Select [jrkim] from the User list and select [0][0][1] Device from the Device list. Click [Register] to set User and Device relation.

8) For more detailed information , refer to Part II ServerManager/Admin “Section 2. Admin ”.

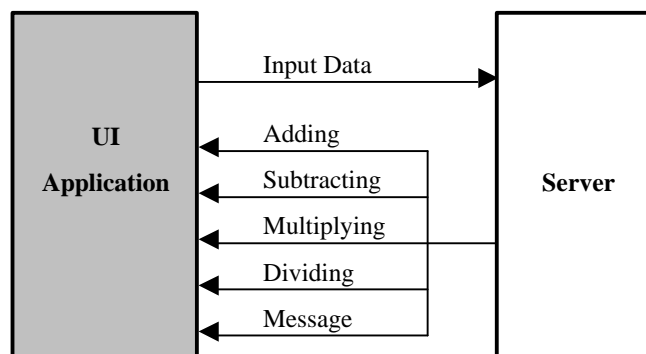
27. Create Server Run Time Program

- 1) Click [Deploy / Make Server] from the ServerManager menu.
- 2) The Server Program is created under D:\Manual\ThisIsServer\Server folder.

2.3 Implementing UI Application

The UI Program transfers the two operands input by the user to the Server, and then displays the operations results received from the Server.

Build UI according to the Server Program [SPData] Protocol Format for inputting data, and build UI according to the Server Program [RPAdd], [RPSubtract], [RPMultiply], [RPDivide] Protocol format for displaying operation results. Print message according to [RPMMessage Protocol].



[Fig 4.15 Project Architecture of UI Application View]

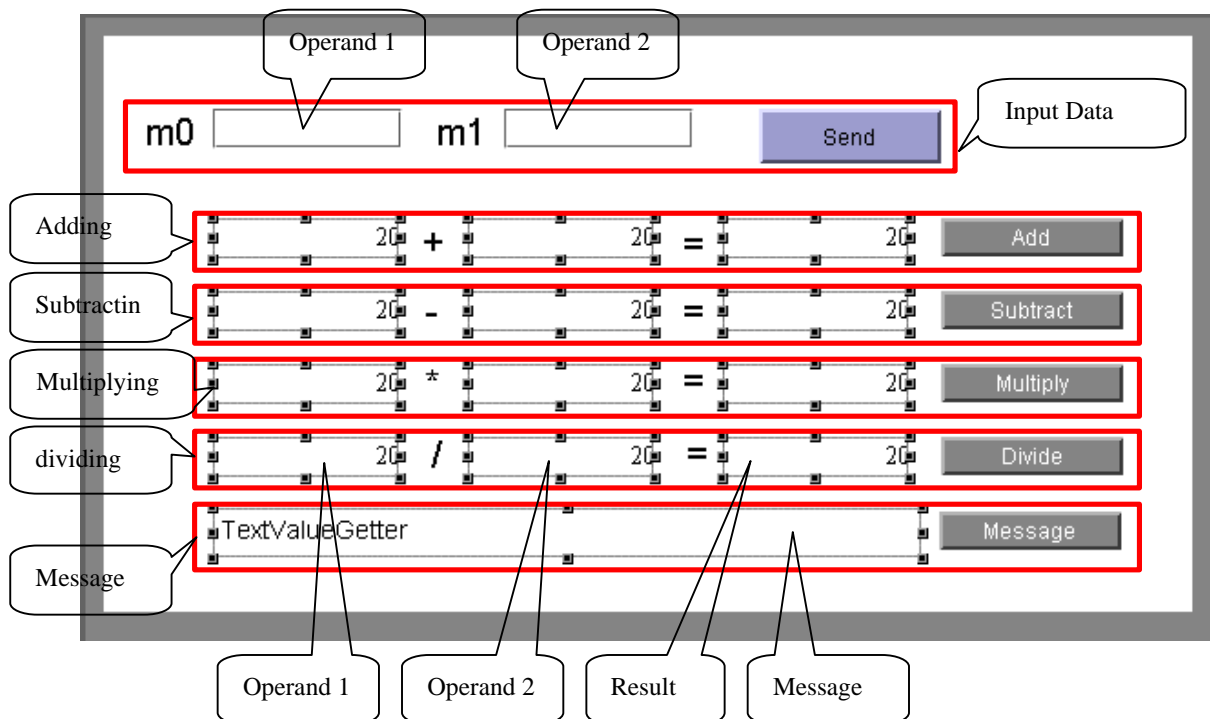
1. Create Folder

Create ThisIsUI Folder under D:\Manual.

2. Create UI Project

- 1) Run UI Manager.
- 2) Create UI Project by selecting [File / New].
- 3) Click [Get device Info.] of [Device Info & Page Panel] to select [D:\Manual\ThisIsServer\TheServer.mdpr] (which is created by ServerManager in the [Open] dialog window) and click [OPEN].
- 4) Type [jrkim] registered in the [User ID] dialog when creating the [Server] program and click [OK] . The UI Manager will bring Device information which will be communicating with the UI Application.

3. Build Design Panel



[Fig 4.16 Build Design Panel]

- 1) [StaticText] Component of [Statics] Component Group
 - [m0], [m1] of input data
 - Adding, Subtracting, Multiplying, Division Results [+], [-], [*], [/], [=]
- 2) [InputFieldValueSetter] Component of [Setters] Component Group
 - Text Field(Operand [Operand1][Operand2] Input Field)
- 3) [DataSendButton] Component of [Controls] Component Group
 - [Send] Button of input data

- 4) [LabelValueGetter] Component of [Getters] Component Group
 - Adding, Subtracting, Multiplying, Dividing Operand [Operand1][Opeand2] and [Result] Text
- 5) [TextLabelValueGetter] Component of [Getters] Component Group
 - [Message] Text Field
- 6) [DataReceiveButton] Component of [Controls] Component Group
 - The results of [Add], [Subtract], [Multiply], [Divide], [Message] Button

4. Set Input Data Properties

- 1) Set [SPData] to be sent to Server program when the user inputs two operands and clicks Send.
- 2) Set [m0] StaticText Properties as follows in the [Property Panel].

Property Editor - StaticText	
Property	Value
Text	m0
Font	Arial, PLAIN, 20
Text alignment	CENTER
Transparent backgr...	true
Text color	
Background color	

In all other StaticText Properties change only the [Text] field. Other properties should be same as [m0].

[Fig 4.17 [m0] StaticText Properties]

- 3) For [m1] StaticText Properties set [m1] in [Text].
- 4) Set [Operand1] InputFieldValueSetter Properties as follows.

Property Editor - ValueSetterInputField	
Property	Value
Protocol symbols	s0
MAC address	00:00:01
Value expression	NONE
Write destination	m0
Write expression	s
Value input or Text i...	Value input
Font	Helvetica, PLAIN, 12
Foreground color	
Background color	

[MAC address] : Use Device Mac Address registered in the Admin Program of the Server.

[Protocol symbols]: Use Server [SPData] Protocol Symbol

[Write destination] : Define [SPData] location (Operand1 value, SPData m0)

[Fig 4.18 [Operand1] InputFieldValueSetter Properties]

- 5) [Operand2] InputFieldValueSetter Properties are the same with [Operand1] but the [Write destination] value is [SPData] Operand2 [m1].

- 6) Set [Send] DataSendButton Properties as follows.

Property Editor - DataSendButton	
Property	Value
Send protocol	s0
Receive protocol	NONE
Title	Send
Title Font	Dialog, PLAIN, 12
Repeat event ?	false
Repeat period	1000
Transparent backgr...	false
Foreground color	
Background color	
Pressed state image	NONE Clear
Released state ima...	NONE Clear

[Send protocol] : [SPData] Protocol Symbol of Server Program

[Fig 4.19 [Send] DataSendButton Properties]

5. Set Adding Results Properties

- 1) Click [Add] to Display the [RPAAdd] Protocol data.
- 2) Set [Operand1] LabelValueGetter Properties as follows:

Property Editor - LabelValueGetter	
Property	Value
Protocol symbols	r1
MAC address	00:00:01
Value expression	m0
Text Font	Serif, PLAIN, 14
ForeGround Color	
BackGround Color	
Horizontal alignment	RIGHT
LED effect	false
IsTransparent	false

[MAC address] : Device Mac Number registered in the DataBase

[Protocol symbols] : [RPAAdd] Protocol Symbol

[Value expression] : [RPAAdd] Protocol [m0]

[Fig 4.20 [Operand1] LabelValueGetter Properties]

- 3) [+] StaticText Properties is [Text] item [+].
- 4) [Operand2] LabelValueGetter Properties is [Value expression] item [Operand2] [m1] and the others are the same as [Operand1].
- 5) [=] StaticText Properties is [=] for [Text].
- 6) [Result] LabelValueGetter Properties is [AddResult] [m2] for [Value expression] and the rest is

the same as [Operand1].

- 7) Set [Add] DataReceiveButton Properties as follows.

Property Editor - DataReceiveButton	
Property	Value
Receive protocol	r1
Title	Add
Title Font	Dialog, PLAIN, 12
Repeat event ?	true
Repeat period	1000
Transparent backgr...	false
Foreground color	
Background color	
Pressed state image	NONE Clear
Released state ima...	NONE Clear

[Receive protocol] : [RPAAdd] Protocol Symbol

[Repeat event?] : Set to [true] to receive data from the Server repeatedly

[Repeat perion] : Set to [1000] to receive results from the Server every second

[Fig 4.21 [Add] DataReceiveButton Properties]

6. Set Subtracting Results Properties

- Click [Subtract] to display the [RPSubtract] Protocol data.
- Set [Operand1] LabelValueGetter Properties as follows:
 - [Protocol symbols] : [RPSubtract] Protocol Symbol [r2]
 - [Value expression] : [RPSubtract] Protocol [Operand 1] [m0]
 The other properties are the same as [Operand1].
- [-] StaticText Properties is [Text] item [-].
- [Operand2] LabelValueGetter Properties is [Value expression] item [Operand2] [m1]. The other properties are the same as [Operand1]..
- [=] StaticText Properties is [Text] item [=].
- [Result] LabelValueGetter Properties is [Value expression] item [SubtractResult] [m2]
 - The other properties are the same as [Operand1]..
- Set [Subtract] DataReceiveButton Properties as follows:
 - [Receive protocol] : [RPSubtract] Protocol Symbol [r2]
 - [Title] : [Subtract]
 The other properties are the same as [Add] DataReceiveButton.

7. Set Multiplying Results Properties

- Click [Multiply] to display [RPMultiply] Protocol data.
- Set [Operand1] LabelValueGetter Properties as follows:
 - [Protocol symbols] : [RPMultiply] Protocol Symbol [r3]
 - [Value expression] : [RPMultiply] Protocol [Operand 1] [m0]

The other properties are the same as [Operand1].

- 3) Set [*] for [Text] of [*] StaticText Properties.
- 4) Set [Operand2] [m1] in [Value expression] of [Operand2] LabelValueGetter Properties.

The other properties are the same as [Operand1].

- 5) [=] StaticText Properties is [Text] item [=].
- 6) [Result] LabelValueGetter Properties is [Value expression] item [MultiplyResult] [m2]

The other properties are the same as [Operand1].

- 7) Set [Multiply] DataReceiveButton Properties as follows:

[Receive protocol] : [RPMultiply] Protocol Symbol [r3]

[Title] : [Multiply]

The other properties are the same as [Add] DataReceiveButton.

8. Set Dividing Results Properties

- 1) Click [Divide] to display [RPDivide] Protocol data.
- 2) Set [Operand1] LabelValueGetter Properties as follows:
[Protocol symbols] : [RPDivide] Protocol Symbol [r4]
[Value expression] : [RPMultiply] Protocol [Operand1] [m0]
The other properties are the same as [Operand1]..
- 3) Set [/] for [Text] in [/] StaticText Properties.
- 4) Set [Operand2] [m1] in [Value expression] of [Operand2] LabelValueGetter Properties.
The other properties are the same as [Operand1].
- 5) Set [=] for [Text] in [=] StaticText Properties.
- 6) Set [DivideResult] [m2] in [Value expression] of [Result] LabelValueGetter Properties. The other properties are the same as [Operand1].
- 7) Set [Divide] DataReceiveButton Properties as follows:
[Receive protocol] : [RPDivide] Protocol Symbol [r4]
[Title] : [Divide]
The other properties are the same as [Add] DataReceiveButton.

9. Set Dividing Results Properties

- 1) Click [Message] to display [RPMessage] Protocol data.
- 2) Set [Message] TextLabelValueGetter Properties as follows:

Property Editor - TextLabelValueGetter	
Property	Value
Protocol symbols	r5
MAC address	00:00:01
Value expression	v0
Text Font	Arial, PLAIN, 14
ForeGround Color	
BackGround Color	
Horizontal alignment	LEFT
LED effect	false
IsTransparent	false

[MAC address] : Device Mac Number registered in the DataBase

[Protocol symbols]: [RPMessag] Protocol Symbol

[Value expression]: [RPMessag] Protocol [Message] [v0]

[Fig 4.22 [Message] TextLabelValueGetter Properties]

- Set [Message] DataReceiveButton Properties as follows:

[Receive protocol] : [RPMessag] Protocol Symbol [r5]

[Title] : [Message]

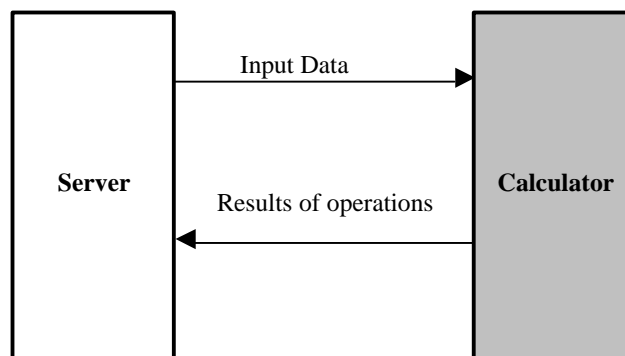
The other properties are the same as [Add] DataReceiveButton.

10. Create UI Application

- Click [File / Save] menu and save UI project as [D:\Manual\ThisIsUI\UICalculator.mdu]
- Select [Depl / Make Application] to open [Create Application] dialog.
- Select [D:\Manual\ThisIsUI] from [Dir] in the [Create Application] dialo and input [127.0.0.1] in the [Default Host Address] section. This will be used later when running the UI Application in the system operating the Server Program. If the UI Application is used in a different system, type the IP Address where the Server Program will run in the [Default Host Address] section.

2.4 Calculator Implementation

The Calculator is a program that returns operation results data received from the Server. As the Calculator becomes a Device from the viewpoint of the Server, the Server likewise becomes the Device from the viewpoint of the Calculator. Input Data becomes the Receive Protocol and the operation results become the Send Protocol. Usually in Server Manager Programs the Send Protocols are input from the UI program made by UI Managers, but the Calculator does not receive Send Protocol operation results from the UI Program. Instead the Calculator utilizes its own Transfer to change the input data into operation results and sends it to the Server.



[Fig 4.23 Project Structure from Calculator Viewpoint]

1. General Property

1) Project Name : TheCalculator

Create Project with name as TheCalculator.

2) Location : D:\Manual\ThisIsCalculator

Create Folder under D:\Manual\ ThisIsCalculator.

3) Daemon Type : Server

Calculator Daemon Type is Server because the Calculator receives input data from the Server (assuming the role of device) and transfers operation results.

4) Port Number : 6001

Refer to (Part II 2. Admin 6) Register Device)

4) DBMS : MS ACCESS

Use MS ACCESS.

5) Class Name : sun.jdbc.odbc.JdbcOdbcDriver

Use JdbcOdbcDriver of Sun Co.

6) URL : Jdbc:Odbc:calculator

Maek ODBC DSN as calculator.

7) User ID, Password : Do not use for this example.

2. Send Protocol

- 1) Formatting data to be sent to the Device is the Send Protocol.
- 2) Define operation results to be sent to Server as Send Protocol.
- 3) Define Send Protocol as SPCalculate.
- 4) The format should be the same as the RPResult (Receive Protocol) of the Server Program.

3. Receive Protocol

- 1) Formatting data to be received from the Device(Server) is the Receive Protocol.
- 2) Define data received from Device(Server) as Receive Protocol(RPData).
- 3) The format should be same as the SPData which is the Server Program's Send Protocol.

4. Transfer

- 1) Change the RPData received from the Device(Server) to SPCalculate data to be sent to the Device.
- 2) Instead of receiving Send Protocol data from the UI Program, use this Transfer to conduct operations and switch results to Send Protocol.
- 3) If dividing with 0(Operand2 = 0), make Error Code.

5. Semantics

Conduct operations using Transfer after receiving RPData from the Server and switch the results to Send Protocol(SPCalculator). Define the transfer act as Semantics SECalculate.

6. Create Folder

Create ThisIsCalculator Folder under D:\Manual Folder.

7. Set ODBC DSN

- 1) Select [Start / Settings / Control Panel] to open [Control Panel].
- 2) Open the [ODBC DSN Administrator] by double clicking [ODBC Data Source (32bit)] from the [Control Panel] window.
- 3) Display the list of [System Data Source] by selecting the [System DSN] Tab from the [ODBC DSN Administrator]. (may be registered in the User DSN)
- 4) Click [Add] to open the [CreateNew Data Source] Dialog window.
- 5) Click [OK] after selecting [Driver do Microsoft Access (*.mdb)] from the [Driver Selection] list of the [Create New Data Source] dialog window.

- 6) Input "calculator" in the text box of [Data Source Name] when the [Setup ODBC Microsoft Access] window opens, and Click [Create].
- 7) Select the [D:\Manual\ThisIsCalculator] folder from the [New DataBase] window.
- 8) Click [Ok] after inputting "calculator.mdb" in the [DataBase Name] text box
- 9) Click [Ok] in the [Setup ODBC Microsoft Access] window.

8. Create Project

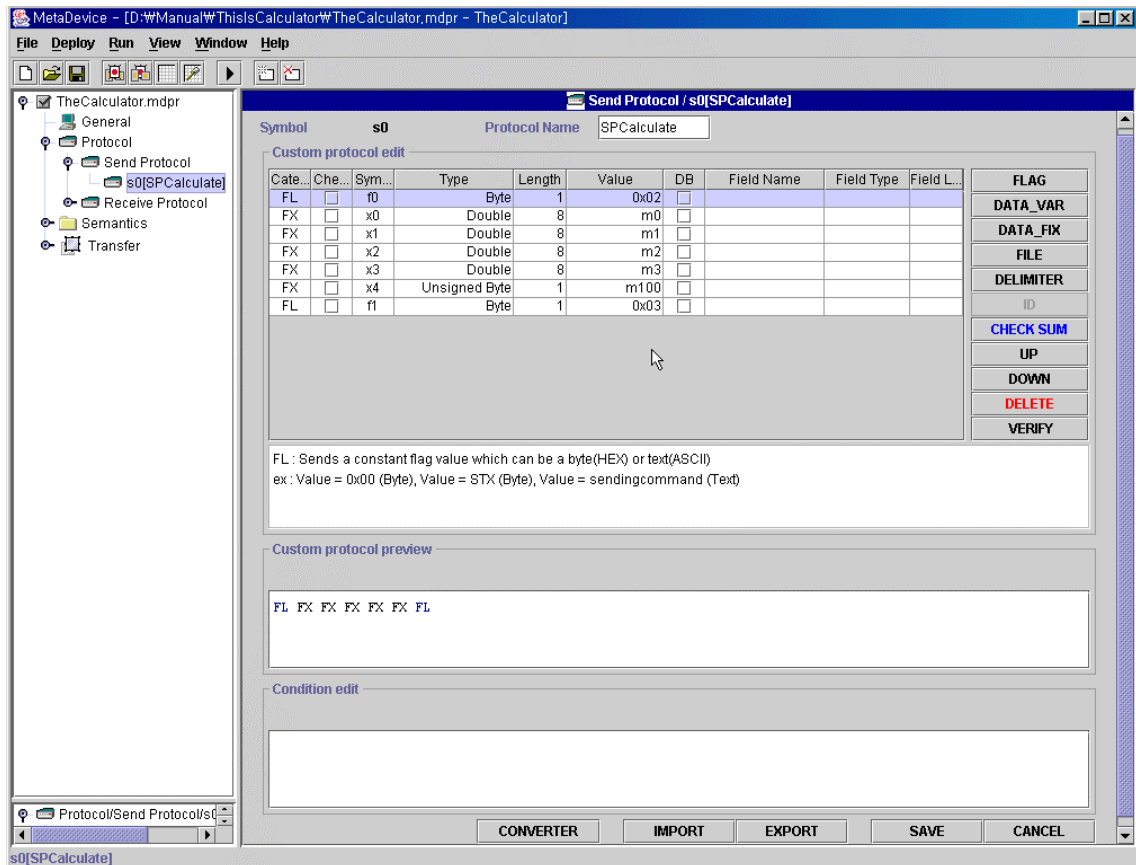
- 1) Run Server Manager.
- 2) Select [File / New] menu to open [New Project] dialog.
- 3) Set [New Project] general properties as follows:
 - Project Name : TheCalculator
 - Location : D:\Manual\ThisIsCalculator
 - Daemon Type : Server (Server radio button)
 - Port Number : 6001
 - DBMS : MS ACCESS (MS ACCESS radio button)
 - Class Name : sun.jdbc.odbc.JdbcOdbcDriver
 - URL : Jdbc:Odbc:calculator
- 4) Click [SAVE] button to create [TheCalculator] Project.

9. Create Send Protocol

- 1) Click [Send Protocol] node from the [Send Protocol] node of [Project Pane] in the upper left corner of the screen.
- 2) Click [Append] to open the [New Serial Protocol] dialog window from the [Send Protocol List Panel] of [Content Pane].
- 3) Input [SPCalculate] in the [Protocol Name] text box, click [OK] and create [SPCalculate] Protocol.

10. Edit Send Protocol

- 1) Double click [SPCalculate] Protocol in the [Send Protocol List Panel] to open the [Send Protocol Edit Panel].
- 2) Edit Send Protocol from the [Send Protocol Edit Panel].



[Fig 4.24 Editing Calculator SPCalculate Protocol]

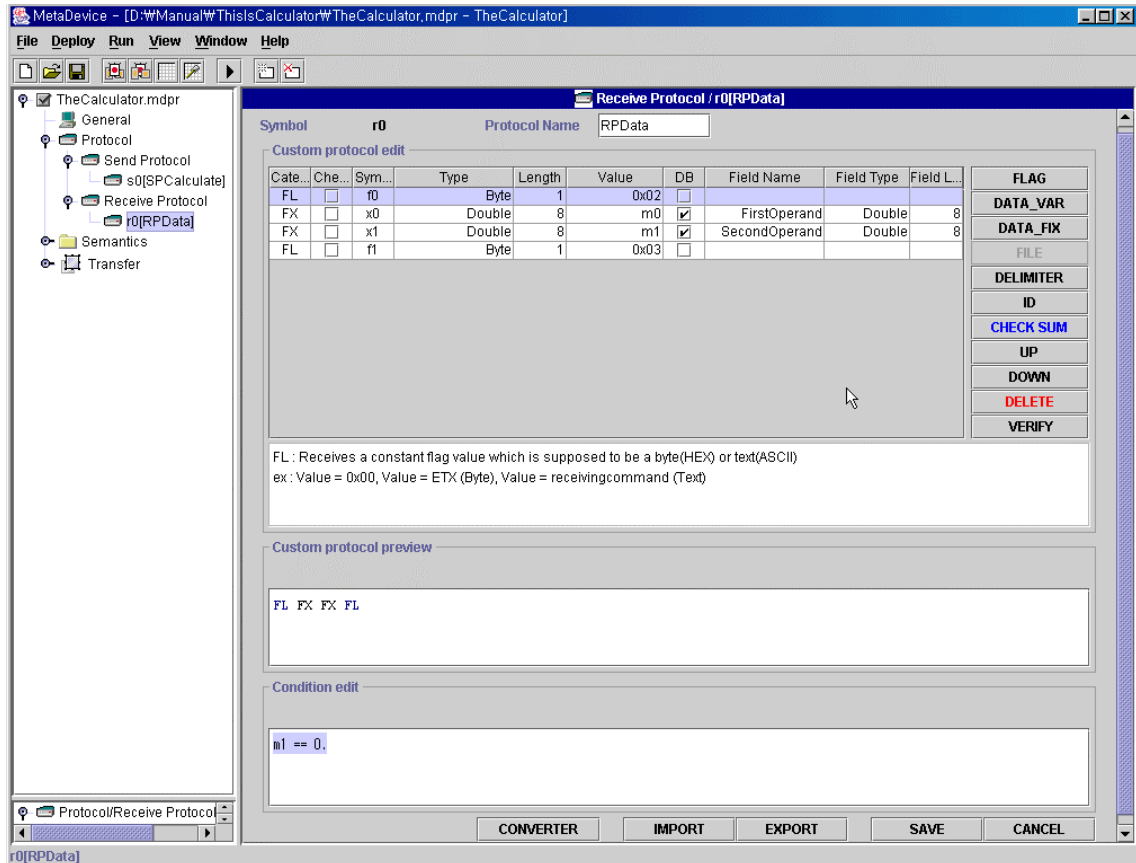
- 3) [SPCalculate] Protocol shows the operation results being sent to the Server.
- 4) It is same format with the Server program's RPResult protocol.

11. Create Receive Protocol

- 1) Click [Receive Protocol] from the [Project] node of [Project Pane] in the upper left corner of the screen.
- 2) Click [Append] in the [Receive Protocol List Panel] from the [Content Pane] to open the [New Serial Protocol] dialog window.
- 3) Input [RPData] in the [Protocol Name] text box. Click [OK] to create the [RPData] Protocol.

12. Edit Receive Protocol

- 1) Double click [RPData] Protocol from the [Receive Protocol List Panel] to open the [Receive Protocol Edit Panel].
- 2) Edit Receive Protocol as follows from the [Receive Protocol Edit Panel].



[Fig 4.25 Editing Calculator RPData Protocol]

- 3) The [RPData] Protocol expresses the input data received from the Server and conducts operations using this data. The [TRCalculate] Transfer will be made to transfer the calculation results to the [SPCalculate] Protocol which will in turn be sent to the Server.
- 4) The format of [RPData] Protocol is the same as that of the [SPData] Protocol of the Server program.
- 5) Check the [DB] edit field and type [Field Name], [Field Type] and [Field Length] edit fields in the [Protocol Edit Table] to store and input operands.
- 6) Input [m1 == 0.] in the [DB Storage Condition Edit] edit window and set as to allow storage only when [RPData] [m1] value is 0, or in other words, when Operand2 is 0. If no conditions are specified, all [RPData] Protocol data will be stored.

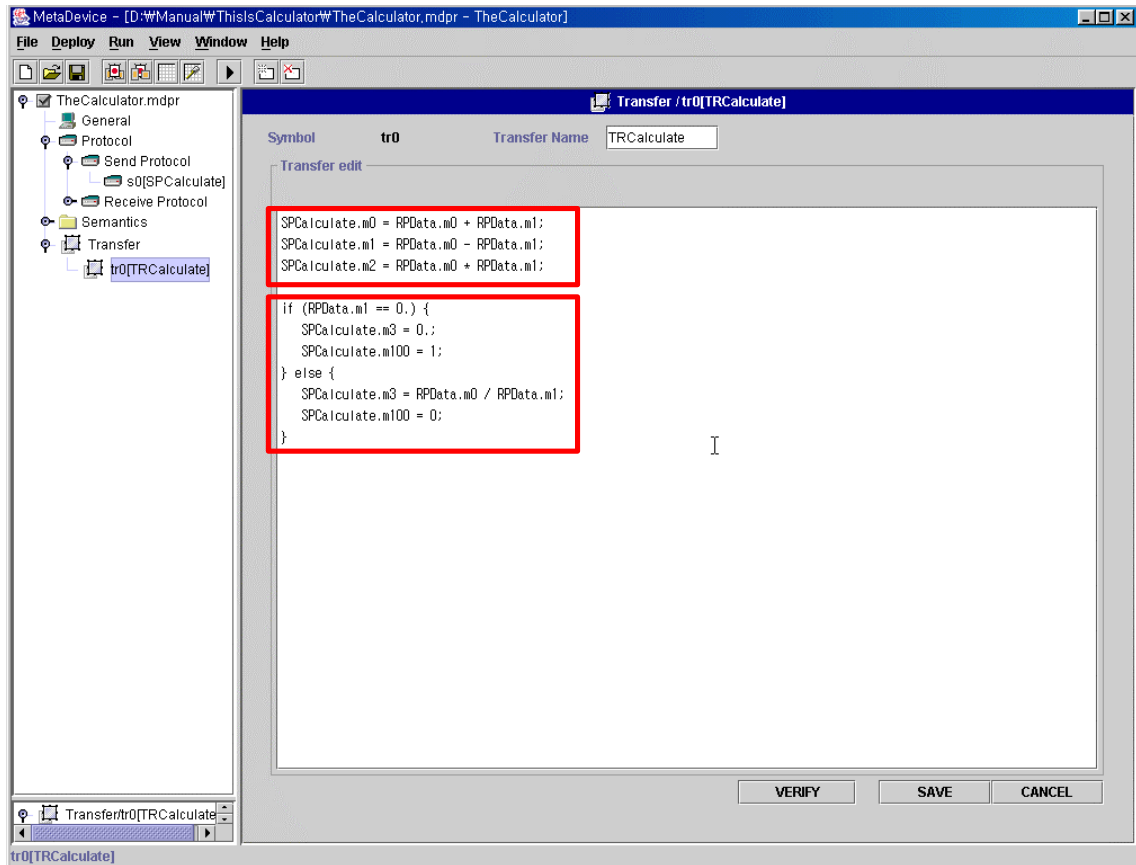
13. Create Transfer

- 1) Click [Transfer] from the [Project Pane] in the upper left corner of the screen.
- 2) Click [Append] from the [Transfer List Panel] in the [ContentPane] to open the [New transfer] dialog.
- 3) Input [TRCalculate] in the [Transfer Name] text box of the [New transfer] dialog and click [OK]

to create [TRCalculate] Transfer.

14. Edit Transfer

- 1) Doubleclick [TRCalculate] Transfer in the [Transfer List Panel] to open [Transfer Edit Panel].
- 2) Edit as follows from the [Transfer Edit Panel].



[Fig 4.26 Editing Calculator TRCalculate Transfer]

- 3) The [TRResult] Transfer is used to transfer input data to operation results to be sent to the Server.
- 4) In the first part Operand1 of [RPData] and adding, subtracting, multiplying results of Operand2 is converted to [SPCalculate].
- 5) In the second part input 0 to the division results and 1 for the error code when Operand2 of [RPData] is 0. When Operand2 is not 0, input the actual division results and 0 for the error code. [m100], Error code of [RPData], is converted to a message displaying the [RPMMessage] Protocol of the Server according to the Server Transfer [TRResult], or errors in dividing with 0.

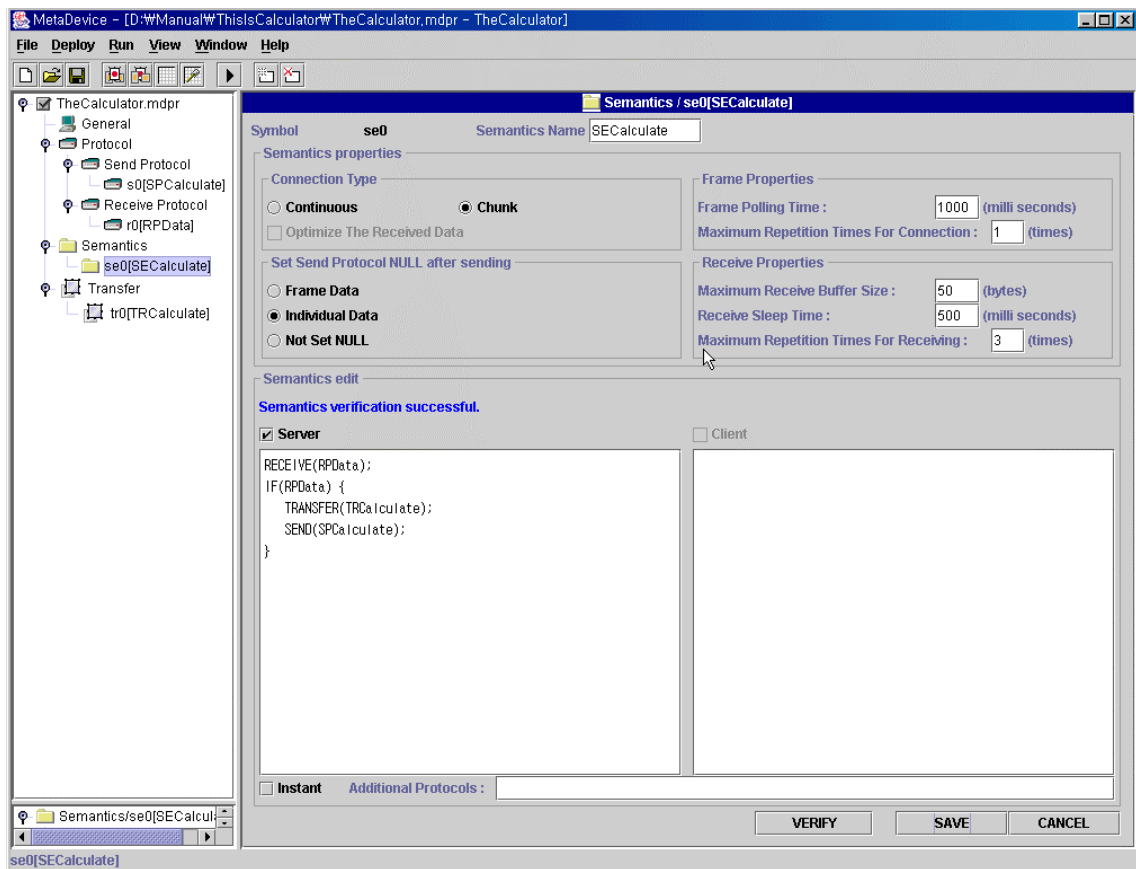
15. Creating Semantics

- 1) Click the [Semantics] node of [Project Pane] in the upper left corner of the screen.

- 2) Click [Append] in the [Semantics List Panel] from [Content Panel] in the upper right corner of the screen to open the [New semantics] dialog.
- 3) Input [SECalculate] in the [Semantics Name] text box of [New semantics] and click [OK] and to create [SECalculate] Semantics.

16. Editing Semantics

- 1) Doubleclick [SECalculate] Semantics from the [Semantics List Panel] to open [Semantics Edit Panel].
- 2) Edit as follows from the [Semantics Edit Panel].



[Fig 4.27 Edit Calculator SECalculate Semantics]

- 3) Set [Connection Type] as [Chunk]. This means that the Calculator sends [SPCalculate] to the Server and then disconnects.
- 4) Set [Set Send Protocol NULL after sending] as [Individual Data]. This means that after being sent to the Server, the [SPCalculate] of the Calculator is set as NULL.
- 5) Set [Frame Polling Time] as [1000]. Because the Daemon Type of the Calculator is Server, this property does not have any effect.
- 6) Set [Maximum Repetition Times For Connection] as 1. Because the Daemon Type of the

Calculator is Server, this property does not have any effect.

- 7) Set [Maximum Receive Buffer Size] as 50, larger than the [RPData]. This means that when the Calculator receives [RPData] from the Server, a 50 byte buffer will be created.
- 8) Set [Receive Sleep Time] as [500], [Maximum Repetition Times For Receiving] as [3]. When the Calculator tries to receive [RPData] but cannot, the Calculator will wait $1.5(500 \times 3)$ Sec, Set an additional 0.5 sec after receiving data to check if there are any additional transfers. (Note : If [Maximum Repetition Time For Receiving] is set as [1], the Server Program will not be able to communicate. Therefore, select a price larger than 1.)
- 9) Edit [Server Semantics] as below in [Server Semantics Edit].

```

RECEIVE(RPData);           The Calculator tries to receive the [RPData].
IF(RPData) {               The Calculator checks if received data is [RPData].
    TRANSFER(TRCalculate);   The Calculator transfers [RPData] to [SPCalculate].
    SEND(SPCalculate);       The Calculator sends [SPCalculate] to the Server.
}

```

17. Admin works

- 1) Save the project file – TheCalculator.mdpr by clicking [File / Save] menu of ServerManager.
- 2) Run the Admin Module and then open the saved project file, TheCalculator.mdpr by clicking [File/Open] menu.
- 3) Create the Admin Tables by clicking [Deploy / Create Admin Tables] menu.
- 4) Input [admin] in [User ID] and [admin] in [PassWd] and click [Login]. Create Protocol tables by clicking [Deploy / Create Protocol Tables] menu.
- 5) There is no need to register a user because the Calculator will not communicate with any user directly.
- 6) Registering Device

Because The Server is a Device from the viewpoint of the Calculator, insert the Server information to the hd table. Select the [Device] node of the [Admin Tables] node in the structure pane(Table work window) and then the [Device] panel will open from the content pane. Open the [Device Register] dialog box by clicking [Insert] in the [Device] pane. Append a new record by clicking [Append] from the [Device Register] dialog box. The [MAC Number]s are [00], [00], [01], the [Current IP] is [192.168.0.32], the [Port No] is [6001], the [Semantics] is [SECalculate] and the [Device Code] is [-1]. The [Current IP] signifies the IP address of the system where the Server will run. Register the Device record to the hd table by clicking the [Register] button in the dialog box.

- 7) Selecting User Device Relation

It is needless to specify relations because the Calculator will not communicate with any user

directly.

- 8) Refer to 2 Server Manager/Admin "2. Admin Works".

18. Create the server program

- 1) Create the Calculator server program by clicking the [Deploy / Make Server] menu in the Server Manager.
- 2) The Calculator server program will be located at D:\Manual\ThisIsCalculator\Server folder.

2.5 Sample Test

It is high time to test whether the Server, the UI Application and the Calculator program will work properly as designed. Because the Calculator must run at other systems than the system where the Server runs, the Calculator program needs to be ported to the system where it will actually run. By running the implemented programs, you will be able to know the results of what you did with the Server Manager and UI Manager.

1. Porting the Calculator

- 1) Create a folder in the system where the [Current IP] was registered during Admin Work. In this sample the [C:\ThisIsCalculator] folder will be created at IP [192.168.18.10].)
- 2) Copy the Server folder, calculator.mdb and TheCalculator.mdpr under [D:\Manual\ThisIsCalculator] to the [C:\ThisIsCalculator] folder of the system where the Calculator will be run.
- 3) Set the ODB DSN in the system where the Calculator will run by referring to 2.4 Implementation Calculator 7. Each step is the same except for selecting [C:\ThisIsCalculator\calculator.mdb] copied above in the [DataBase Name] text box of procedure 8).

2. Running the Calculator

Run [TheCalculator.bat] below in the [C:\ThisIsCalculator\Server] folder of the system where the Calculator will run by doubleclicking the file in Explorer or running the [TheCalculator] below [C:\ThisIsCalculator\Server] in DOS.

3. Running the Server

Run [TheServer.bat] under [D:\Manual\ThisIsServer\Server] folder.

4. Running the UI Application

Run the [MyApplication.bat] under [D:\Manual\ThisIsUI] folder and then the [Log In] dialog box of [MyApplication-MetaDevice _UI_Manager] program will open. Input [jrkim] for the User ID and

Password which were registered in Admin work while implementing the Server program. Click [Submit] to run [MyApplication-MetaDevice _UI_Manager] UI Application.

5. Test

Input many kinds of m0, m1 and click [Send]. Click [Add], [Subtract], [Multiply], [Divide] and [Message] and to receive data results from the Server and display them every second. If m1 is 0, the Message will be "ERROR:Do_not_divide_by_0".