eLogger Web API DLL
-register.dll API Manual-

Version 1.0.2 ,2012/10/29

Written by Amber Hsieh

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Contact US

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

Table of contents

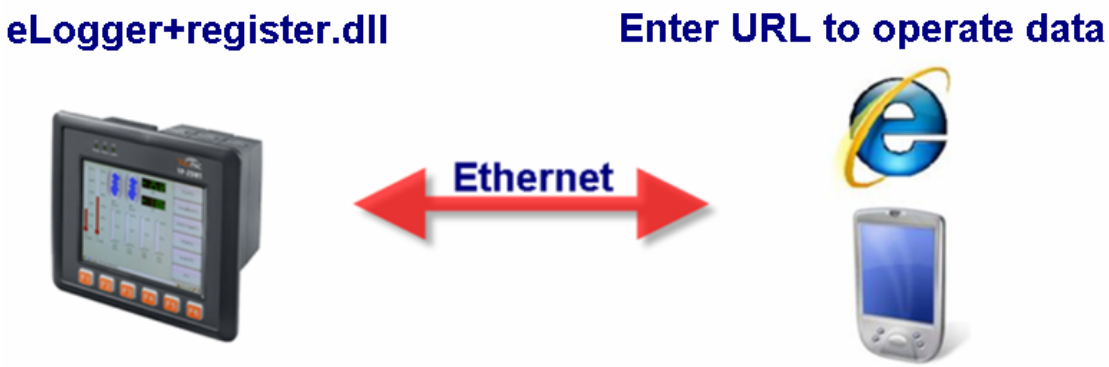
1. Introduction

eLogger is a free of charge and easy-to-use software to implement HMI and data logger applications on the following programmable automation controller (PAC) of ICPDAS:

- WinPAC, ViewPAC (OS: Windows CE .NET 5.0)
- XP8000-CE6 (OS: Windows CE .NET 6.0)
- XP8000 (OS: Windows Embedded Standard 2009)

The web API DLL “register.dll” provided by ICPDAS enables the user to access via the HTTP protocol remote IO modules over the Internet by reading from and writing to the eLogger Shared Memory.

The DLL APIs can be executed by entering the API name with its parameter in the URL of a standard browser. So users can take advantage of “register.dll” to develop web programs or mobile app to do a remote I/O monitoring.



2. register.dll API

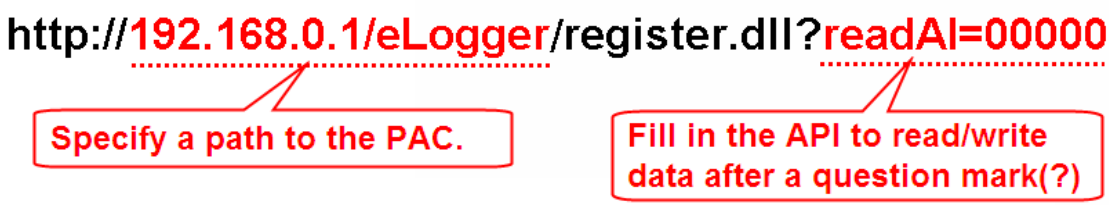
eLogger includes a Memory Address Map. The register.dll provides 24 API for users to read and write eLogger “Shared Memory” data of the specified address. This URL format is：**http://IP address/…/register.dll?Parameter**

Example 1:

If the register.dll is located in the web root folder, you can set the URL as follow：

Example 2:

If the register.dll is located in another directory which is under the web root folder, you can set the URL as follow：



2.1. Read API

Read Reference

The Read APIs enables the user to directly retrieve data values from eLogger Shared Memory.

Function List

Function	Description
readAI=xxxxx	Reads an analog input value (integer) from the specified AI address.
readAI_UInt=xxxxx	Reads an analog input value (unsigned integer) from the specified AI address.
readAI_Long=xxxxx	Reads an analog input value (long integer) from the specified AI address.
readAI_ULong=xxxxx	Reads an analog input value (unsigned long integer) from the specified AI address.
readAI_Float=xxxxx	Reads an analog input value (float) from the specified AI address.
readAI=xxxxx&len=ooo	Reads several analog input values (integer) from consecutive AI addresses.
readAI_UInt=xxxxx&len=ooo	Reads several analog input values (unsigned integer) from consecutive AI addresses.
readAO=xxxxx	Reads an analog output value (integer) from the specified AO address.
readAO_UInt=xxxxx	Reads an analog output value (unsigned integer) from the specified AO address.
readAO_Long=xxxxx	Reads an analog output value (long integer) from the specified AO address.
readAO_ULong=xxxxx	Reads an analog output value (unsigned long) from the specified AO address.
readAO_Float=xxxxx	Reads an analog output value (float) value from the specified AO address.
readAO=xxxxx&len=ooo	Reads several analog output values (integer) from consecutive AO addresses.

readAO_UInt=xxxxx&len=ooo	Reads several analog output values (unsigned integer) from consecutive AO addresses.
readDI=xxxxx	Reads the digital input status from the specified DI address.
readDI=xxxxx&len=ooo	Reads several digital input status from consecutive DI addresses.
readDO=xxxxx	Reads the digital output status from the specified DO address.
readDO=xxxxx&len=ooo	Reads several digital output status from consecutive DO addresses.

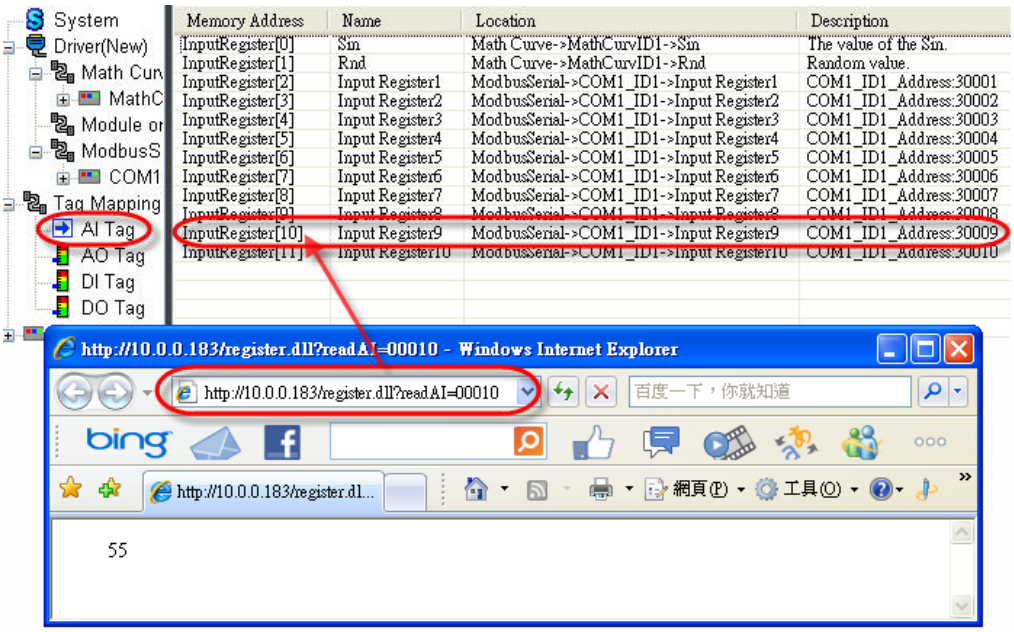
2.1.1 readAI=xxxxx

This function reads an analog input value of “integer” type from a specified AI address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “**readAI”** is case sensitive.

Example:

http://10.0.0.183/register.dll?readAI=00010

Read an analog value from eLogger Shared Memory at address AI 10.



2.1.2 readAI_UInt=xxxxx

This function reads an analog input value of “unsigned integer” type from the specified AI address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAI_UInt=” is case sensitive.

2.1.3 readAI_Long=xxxxx

This function reads an analog input value of “long integer” type from a specified AI address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAI_Long=” is case sensitive.

2.1.4 readAI_ULong=xxxxx

This function reads an analog input value of “unsigned long integer” type from a specified AI address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAI_ULong=” is case sensitive.

2.1.5 readAI_Float=xxxxx

This function reads an analog input value of “float” type from the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAI_Float=” is case sensitive.

2.1.6 readAI=xxxxx&len=ooo

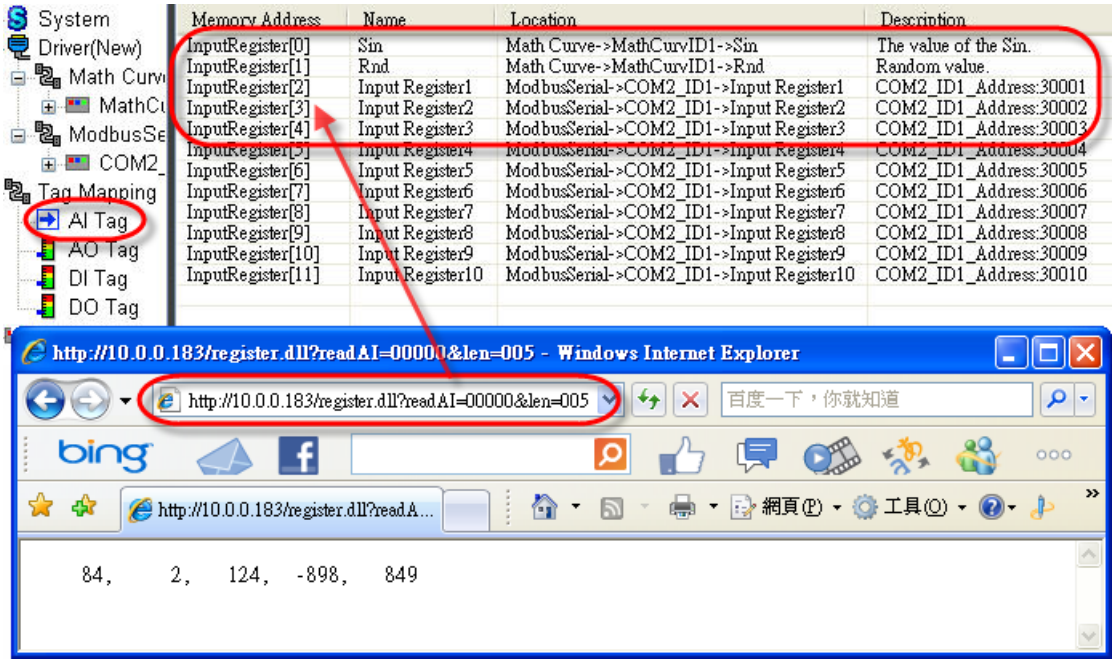
This function reads several analog input values of “integer” type from consecutive AI addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readAI=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as:
“ 32767, 12,-32768, -10”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

Example:

http://10.0.0.183/register.dll?readAI=00000&len=005

Read five consecutive values from the Shared Memory address AI 0~AI 4.



2.1.7 readAI_UInt=xxxxx&len=ooo

This function reads several analog input values of “unsigned integer” type from consecutive AI addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readAI_UInt=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as: “ 32767, 12,-32768, -10”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

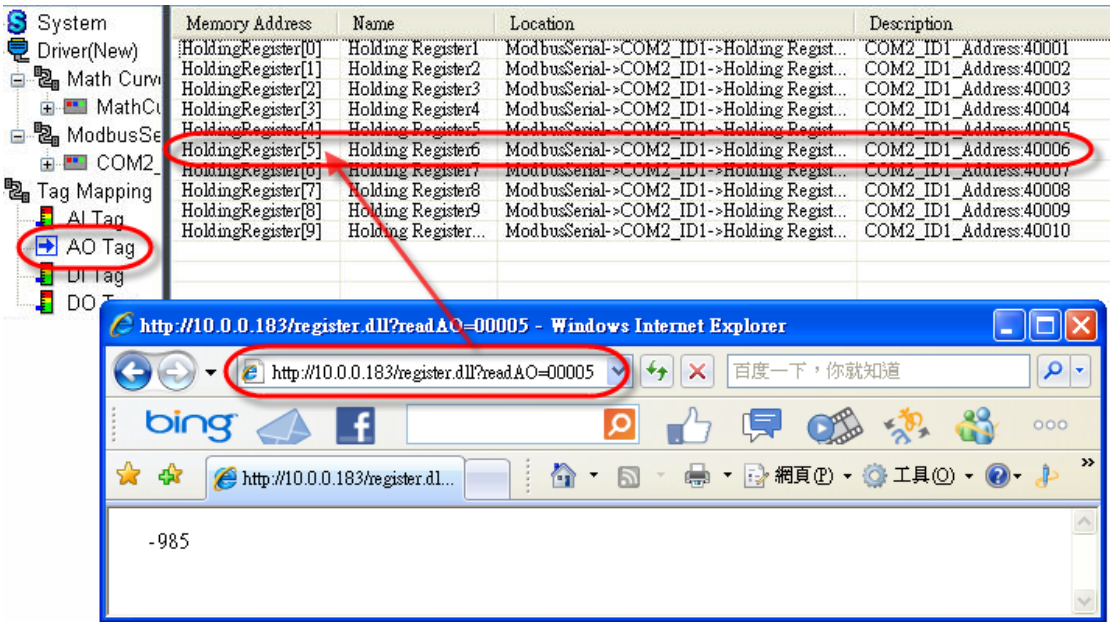
2.1.8 readAO=xxxxx

This function reads an analog output value of “integer” type from a specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAO” is case sensitive.

Example:

http://10.0.0.183/register.dll?readAO=00005

Read an analog value from eLogger Shared Memory at address AO 5.



2.1.9 readAO_UInt=xxxxx

This function reads an analog output value of “unsigned integer” type from the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAO_UInt=” is case sensitive.

2.1.10 readAO_Long=xxxxx

This function reads an analog output value of “long integer” type from the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAO_Long=” is case sensitive.

2.1.11 readAO_ULong=xxxxx

This function reads an analog output value of “unsigned long integer” type from the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAO_ULong=” is case sensitive.

2.1.12 readAO_Float=xxxxx

This function reads an analog output value of “float” type from the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readAO_Float=” is case sensitive.

2.1.13 readAO=xxxxx&len=ooo

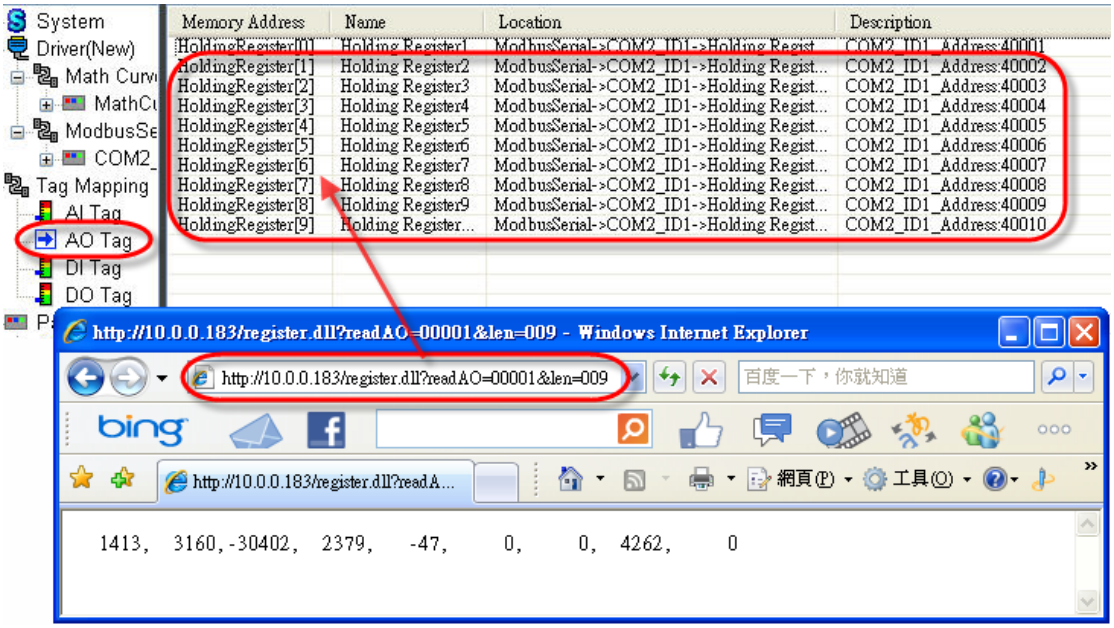
This function reads several analog output values of “integer” type from consecutive AO addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readAO=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as:
“ 32767, 12,-32768, -10”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

Example:

http://10.0.0.183/register.dll?readAO=00001&len=009

Read five consecutive values from Shared Memory address AO 1~AO 9.



2.1.14 readAO_UInt=xxxxx&len=ooo

This function reads several analog output values of “unsigned integer” type from consecutive AO addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readAO_UInt=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as: “ 32767, 12,-32768, -10”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

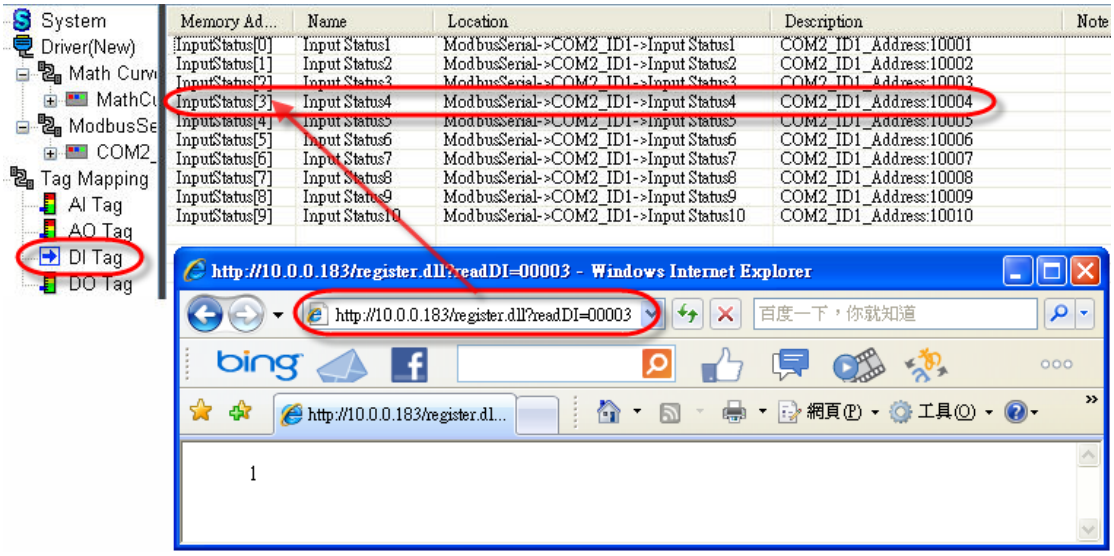
2.1.15 readDI=xxxxx

This function reads the ON/OFF status (1=ON; 0=OFF) from the specified DI address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readDI=” is case sensitive.

Example:

EX：http://10.0.0.183/register.dll?readDI=00003

Read the ON/OFF status from the eLogger Shared Memory address DI 3.



2.1.16 readDI=xxxxx&len=ooo

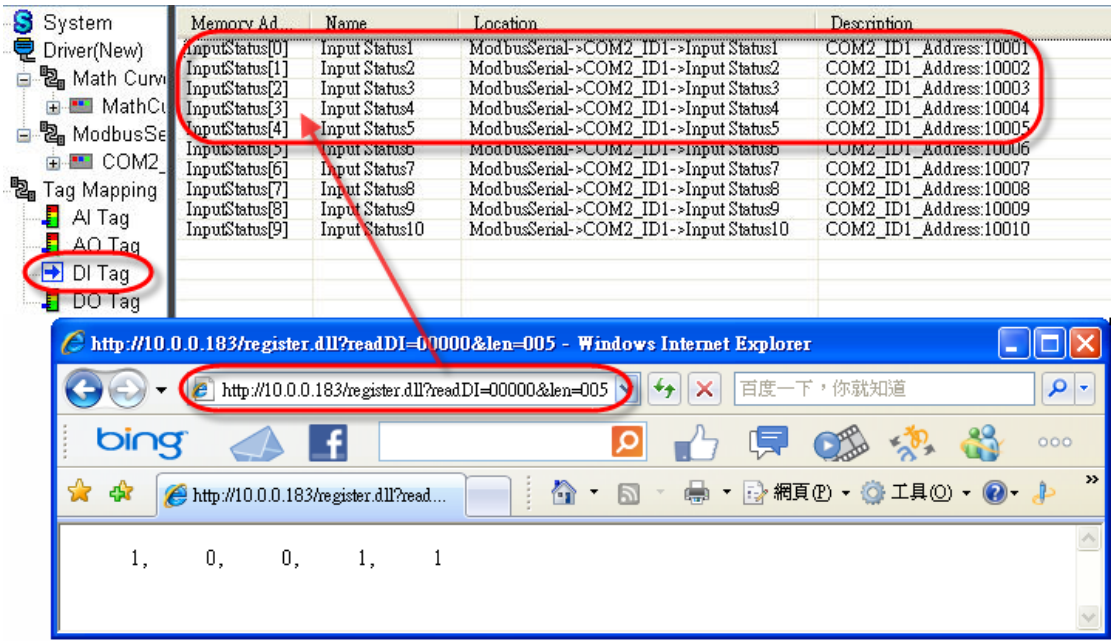
This function reads several ON/OFF status (1=ON; 0=OFF) from consecutive DI addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readDI=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as: “ 1, 1, 0, 1”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

Example:

http://10.0.0.183/register.dll?readDI=00000&len=005

Read five consecutive values from Shared Memory address DI 0~DI 5.



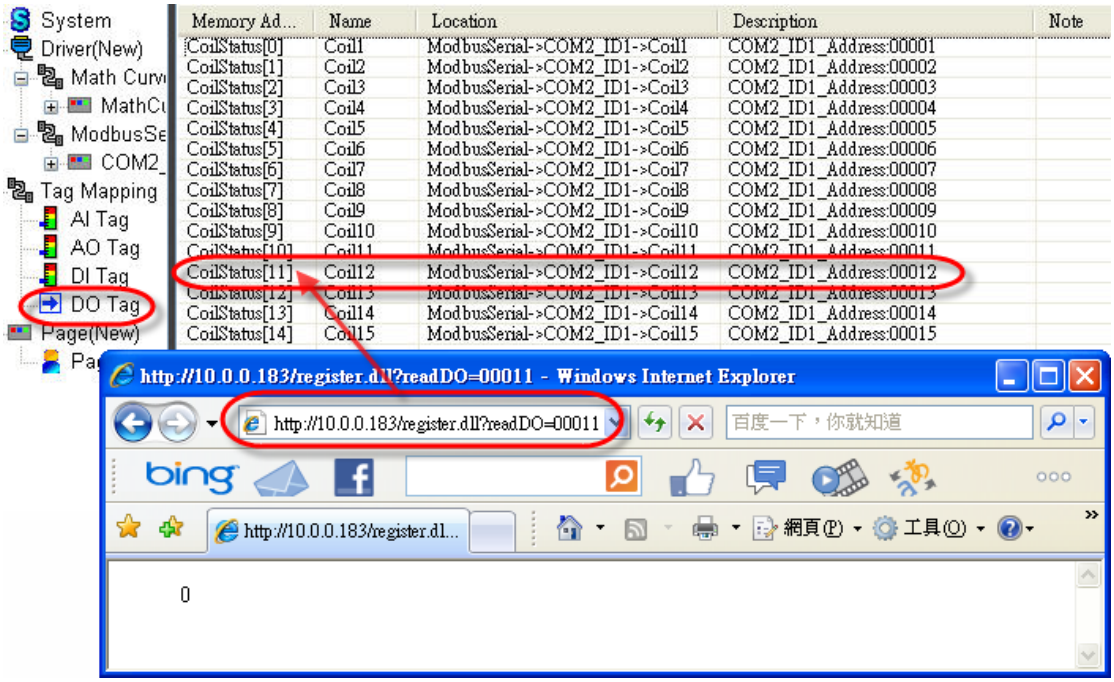
2.1.17 readDO=xxxxx

This function reads the ON/OFF status (1=ON; 0=OFF) from the specified DO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999. Please note that the API name “readDI=” is case sensitive.

Example:

EX：http://10.0.0.183/register.dll?readDO=00011

Read the ON/OFF status from eLogger Shared Memory address DO 11.



2.1.18 readDO=xxxxx&len=ooo

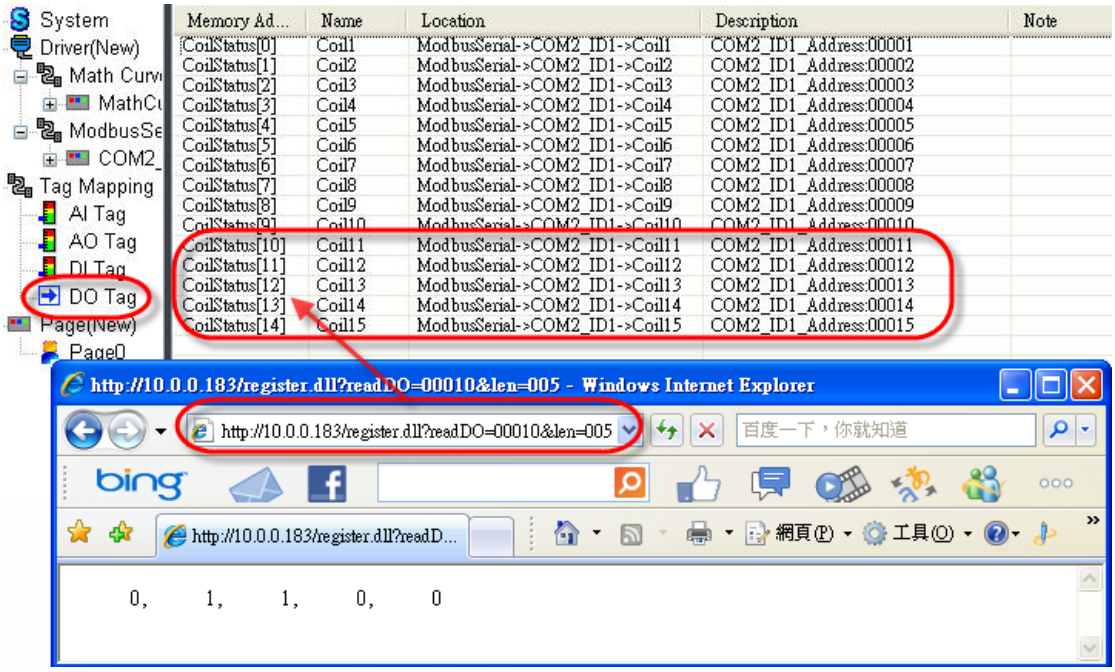
This function reads several ON/OFF status (1=ON; 0=OFF) from consecutive DO addresses. The “x” represents a five-digit start address location in the eLogger Shared Memory, with a valid start range of 00000 to 99999；The “o” represents a three-digit data length, and the maximum length is 256. Please note that the API name “readDO=xxxxx&len=” is case sensitive.

This function returns the retrieved values in a string whereby each value is separated by a comma. The string length of each value is six digits. (If a value is smaller than six digits then the front is filled up with spaces) Such as: “ 1, 1, 0, 1”. The advantage of this approach is that users can distinguish the set of values in the string by either counting the number of digits (each value is six digits long) or by looking for a comma which indicates the beginning of a new value.

Example:

http://10.0.0.183/register.dll?readDO=00010&len=005

Read five consecutive values from Shared Memory address DO10~DO14.



2.2. Write API

Write Reference

Write APIs allows the setting of values in the eLogger Shared Memory.

Function List

Function	Description
writeAO=xxxxx&data=value	Writes an analog output value (integer) to the specified AO address.
writeAO_UInt=xxxxx&data=value	Writes an analog output value (unsigned integer) to the specified AO address.
writeAO_Long=xxxxx&data=value	Writes an analog output value (long integer) to the specified AO address.
writeAO_ULong=xxxxx&data=value	Writes an analog output value (unsigned long integer) to the specified AO address.
writeAO_float=xxxxx&data=value	Writes an analog output value (float) to the specified AO address.
writeDO=xxxxx&data=value	Writes the digital output status to the specified DO address.

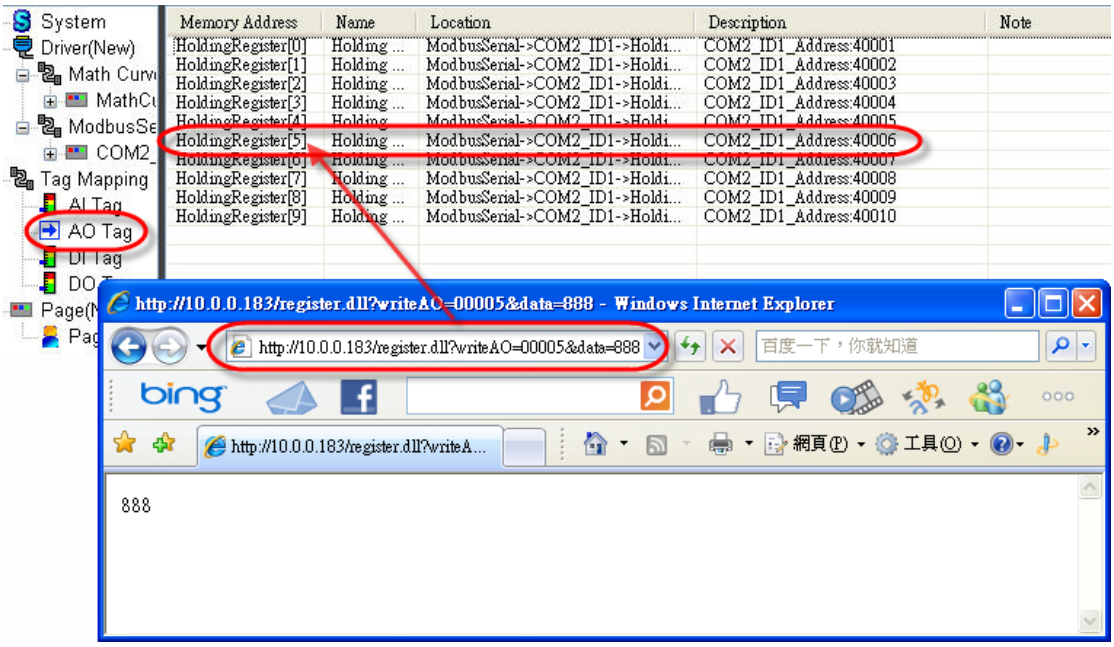
2.2.1 writeAO=xxxxx&data=value

This function writes an analog output value of “integer” type to the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the AO value to be written to the AO address. Please note that the API name “writeAO=xxxxx&data=” is case sensitive.

Example:

http://10.0.0.183/register.dll?writeAO=00005&data=888

Write value “888” to the eLogger Shared Memory address AO 5.



2.2.2 writeAO_UInt=xxxxx&data=value

This function writes an analog output value of “unsigned integer” type to the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the AO value to be written to the AO address. Please note that the API name “writeAO_UInt=xxxxx&data=” is case sensitive.

2.2.3 writeAO_Long=xxxxx&data=value

This function writes an analog output value of “long integer” type to the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the AO value to be written to the AO address. Please note that the API name “writeAO_Long=xxxxx&data=” is case sensitive.

2.2.4 writeAO_ULong=xxxxx&data=value

This function writes an analog output value of “unsigned long integer” type to the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the AO value to be written to the AO address. Please note that the API name “writeAO_ULong=xxxxx&data=” is case sensitive.

2.2.5 writeAO_Float=xxxxx&data=value

This function writes an analog output value of “float” type to the specified AO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the AO value to be written to the AO address. Please note that the API name “writeAO_Float=xxxxx&data=” is case sensitive.

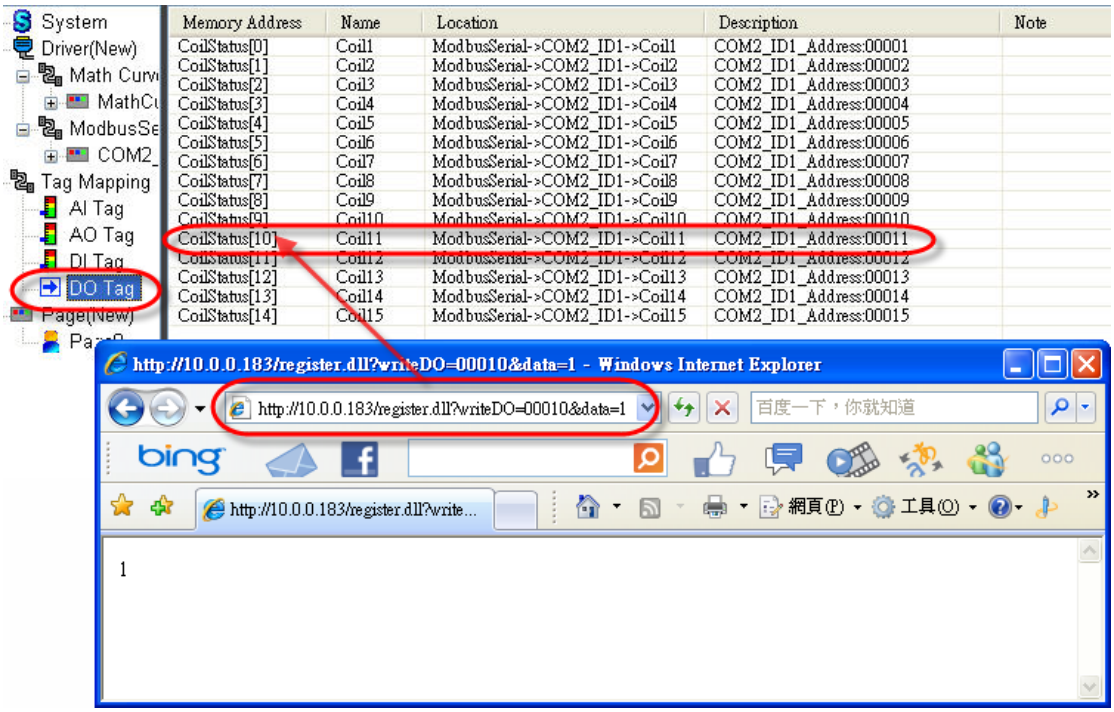
2.2.6 writeDO=xxxxx&data=value

This function writes the ON/OFF status (1=ON; 0=OFF) to a specified DO address. The “x” represents a five-digit address location in the eLogger Shared Memory, with a valid range of 00000 to 99999; the “value” represents the DO status (1=ON; 0=OFF) to write to the DO address. Please note that the API name “writeDO=xxxxx&data=” is case sensitive.

Example:

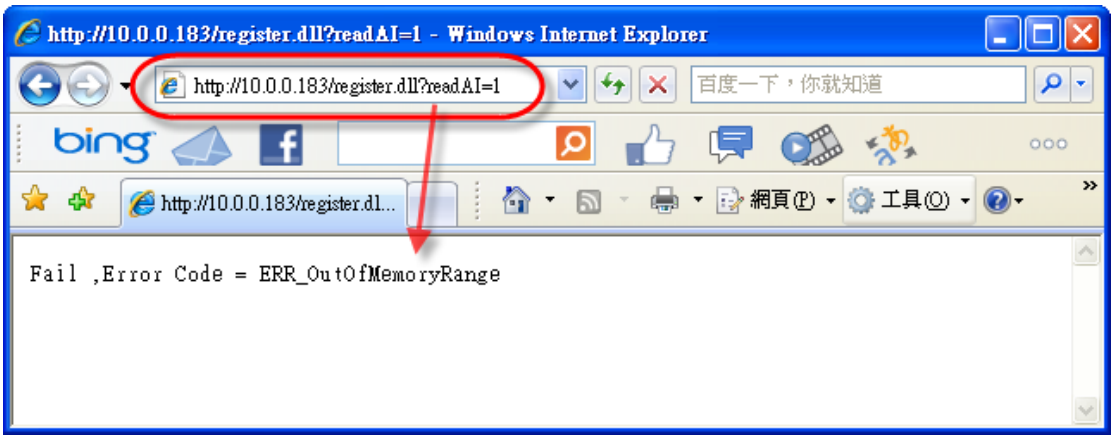
EX：http://10.0.0.183/register.dll?writeDO=00010&data=1

Write value 1 to the eLogger Shared Memory address DO 10, and the status is ON.



2.3. Error Code Reference

When users sent a wrong or invalid URL command, the register.dll will return an error string. An error string example is shown in the following figure: ”Fail, Error Code: ERR_OutOfMemoryRange”



The error codes are described in the table.

Error Code	Description
ERR_OutOfMemoryRange	The address is invalid, incompatible formats.
ERR_WrongCommand	The parameter is invalid. (Misspelled or not case sensitive)
ERR_WriteDataNotNumber	The value written to the eLogger Shared Memory not a number.
ERR_WriteDataNotInterger	The value written to the eLogger Shared Memory not an integer.
ERR_WriteDataNotUInterger	The value written to the eLogger Shared Memory not an unsigned integer.
ERR_WriteDataNotLong	The value written to the eLogger Shared Memory not a long integer.
ERR_WriteDataNotULong	The value written to the eLogger Shared Memory not an unsigned long integer.
ERR_Unknown	Unknown error.

3. How To Configure register.dll

ICPDAS provide for each operation system a different register.dll different version. Therefore make sure to select the correct version corresponding to your OS. The eLogger setup file can be downloaded from ICP DAS website. After unzipping the setup file, copy the appropriate runtime directory to your PAC, and follow the steps below to configure the register.dll.

The table below displays for the different OS supported by the PAC series the corresponding eLogger runtime file name and register.dll.

OS	eLogger runtime file	register.dll
CE5	For_WinPAC	 register.dll 1.0.2.0 eLogger Web API CE5
CE6	For_XP8000CE6	 register.dll 1.0.2.0 eLogger Web API CE6
XPE	For_XP8000WES	 register.dll 1.0.2.0 eLogger Web API XPE

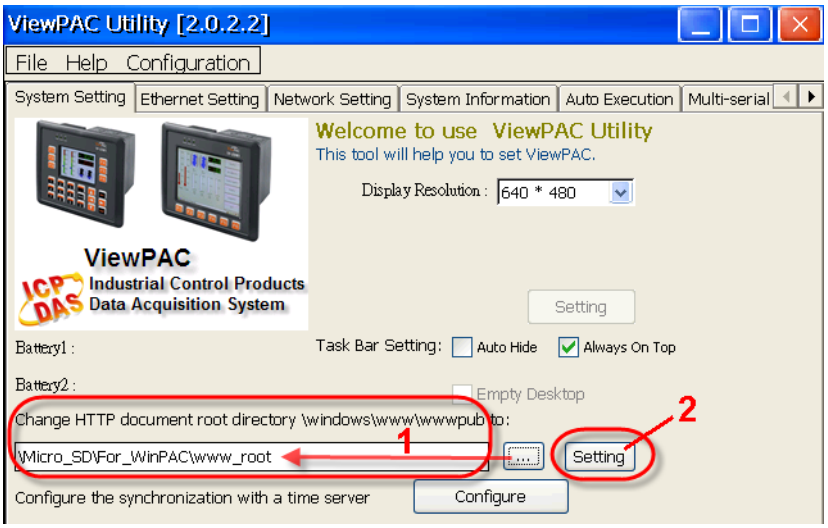
3.1. Windows CE5 based PACs

Step 1: Copy the eLogger runtime folder ”For_WinPAC” to any folder of the PAC.

Example：\Micro_SD\For_WinPAC

Step 2: Open the folder “\Micro_SD\For_WinPAC\system_disk\icpdas\system” and copy the file” SharedMemory.dll” to the PAC’s system folder “\System_Disk\icpdas\system” to allow multiple programs to read and write eLogger Shared Memory.

Step 3: Execute “ViewPAC Utility” on desktop, switch to “System Setting” tab, click button to change the web root directory to the following directory: \Micro_SD\For_WinPAC\www_root, and then click the “Setting” button to finish the setting. Now the register.dll can be used by the network.



3.2. Windows CE6 based PACs

Step 1: Copy the eLogger runtime folder ”For_XP8000CE6” to the any folder on the PAC.

Example：System_Disk\For_XP8000CE6

Step 2: Open the folder：

"\System_Disk\For_XP8000CE6\system_disk\icpdas\system and copy the file “SharedMemory_CE6.dll” to the PAC’s system folder \System_Disk\ICPDAS\SYSTEM to allow multiple programs to read and write eLogger Shared Memory.

Step 3: Open the folder \System_Disk\For_XP8000CE6\www_root and copy the file “register.dll” to the PAC’s web root directory (Note) \System_Disk\www. The register.dll can now be accessed via internet.

Note：Start the “XPAC Utility” on the desktop, and switch to “Network” tab to confirm that the web root directory is \System_Disk\www. If not then click the “ …” button to change to the correct directory, and then click the “Apply” button to finish the setting.

3.3. Windows Embedded Standard 2009 PACs

Step 1: Copy the eLogger runtime folder ”For_XP8000WES” to the any folder on the PAC.

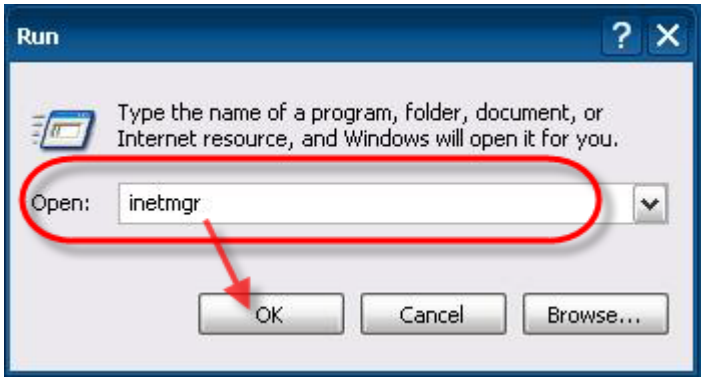
Example：T:\For_XP8000WES

Step 2: Open the folder：” T:\For_XP8000WES \windows” and copy the file “SharedMemory_XP.dll” to the PAC’s system folder ” C:\windows” to allow multiple programs to read and write eLogger Shared Memory.

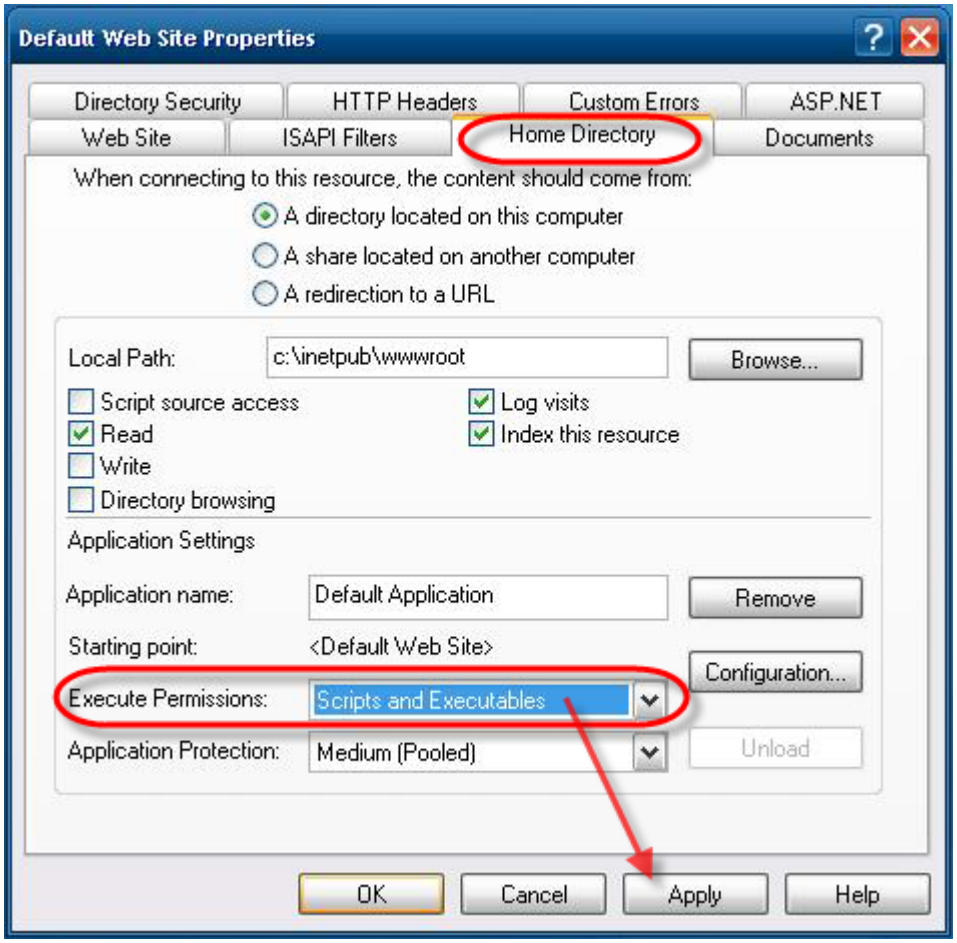
Step 3: Open the folder “T:\For_XP8000WES\www_root” and copy the file “register.dll” to the PAC’s web root directory “C:\inetpub\wwwroot”.

Step 4: Enable the “Scripts and Executes” rights of the Web directory. So that users can operate the register.dll by network.

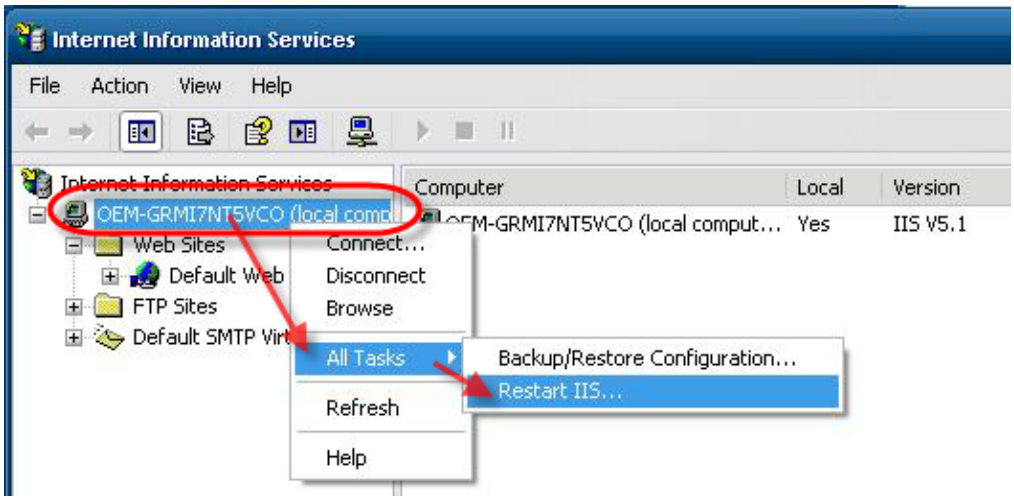
a. Start → Run → insert the command “inetmgr” to open IIS.



b. Right click ”Default Web Site” to select ”Properties” → on the ”Home Directory” tab, select the ”Scripts and Executes” check box for “Execute permissions” → click the “Apply” button → click the “OK” button.



c. In the IIS manager, right click the local computer, point to “All Tasks”, then click “Restart IIS”



4. How To Develop Android App

<Setup and Tutorials>

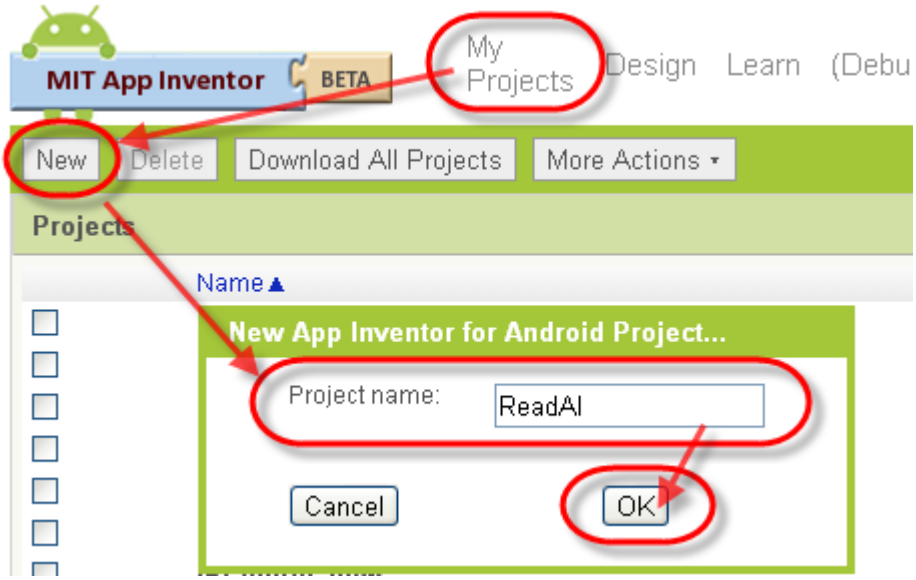
MIT App Inventor website

App Inventor teaching website

Example1：Read data from the eLogger Shared Memory AI 1

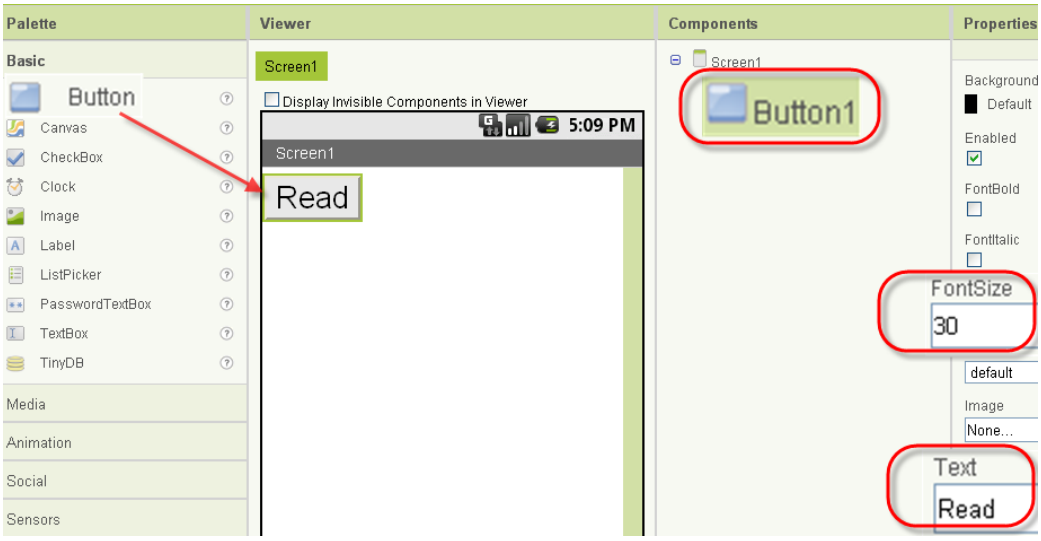
Step 1: Build a new project

Login Gmail → open The Designer → move to “My Projects” page and build a new project named “ReadAI”.

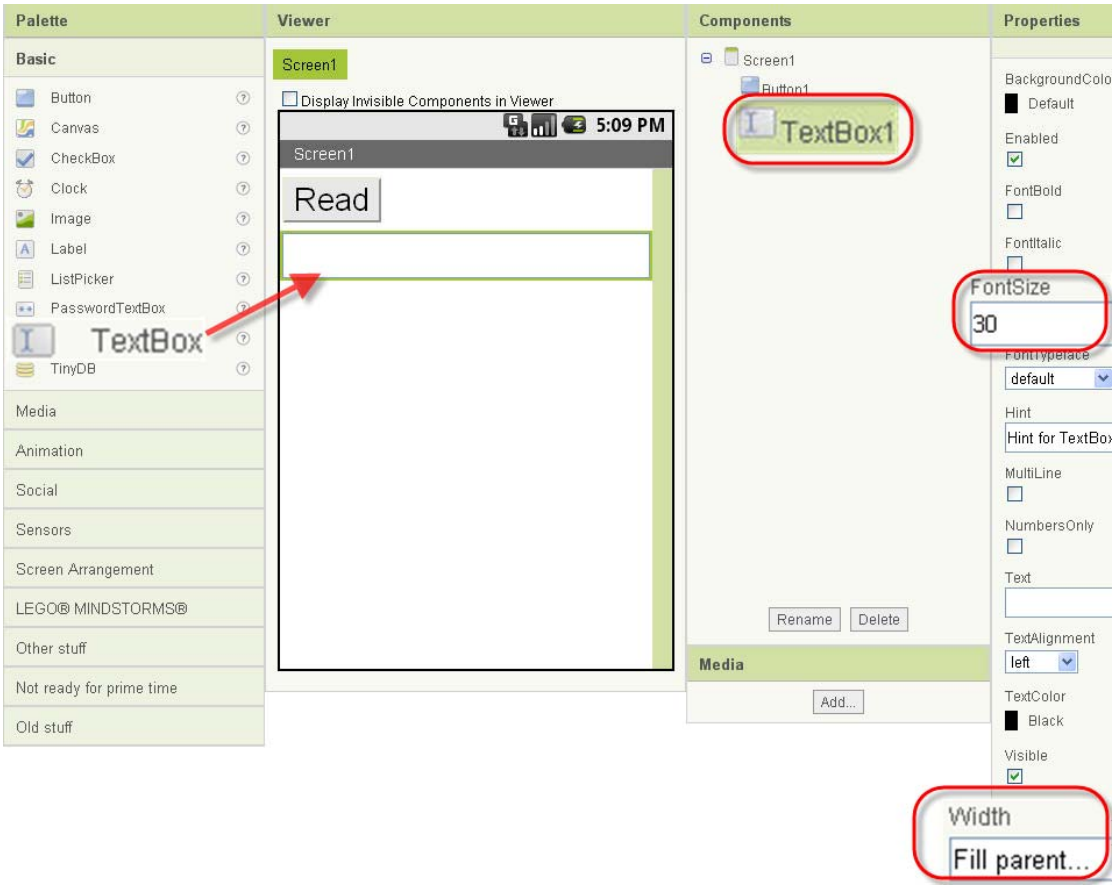


Step 2: The Designer creates the User Interface

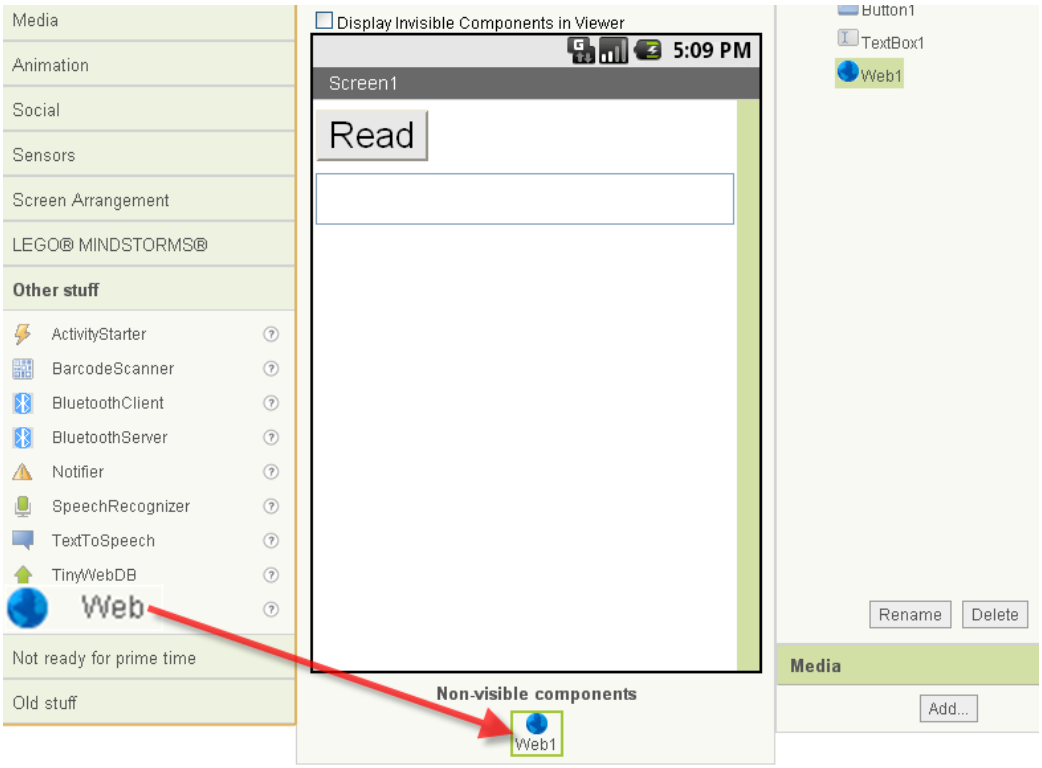
2.1 Select the “Basic” tab, and drag a “Button” component on “Screen1”. Change the button’s font size (30) and text (Read) in the “Properties” panel.



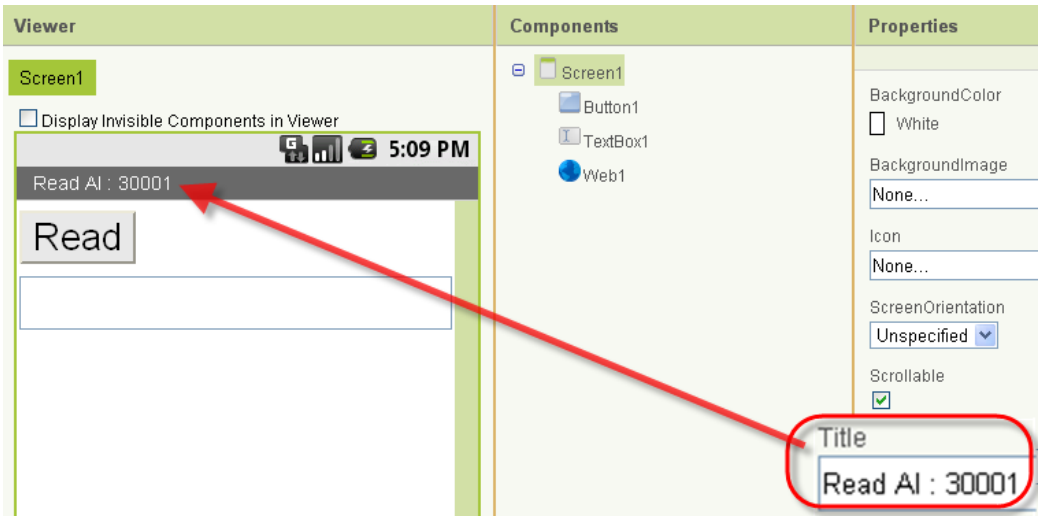
2.2 Repeat the drag-and-drop for the “TextBox”, and change its font size (30) and width (fill parent).



2.3 Select the “Other stuff” tab, and drag a “Web” component on “Screen1”.



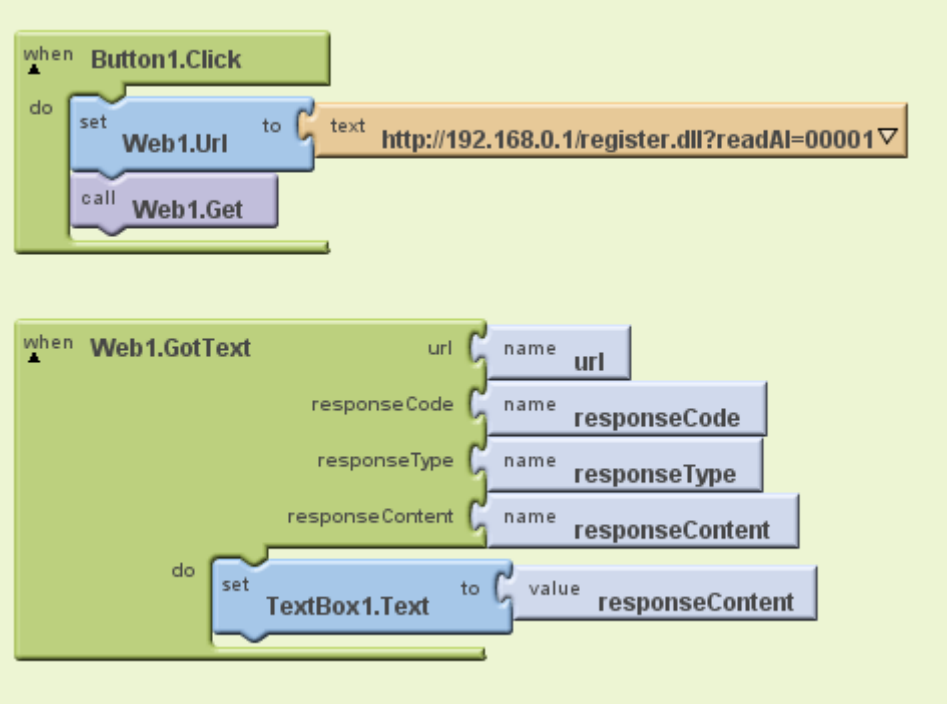
2.4 Change the Screen1’s title (Read AI :30001).



Step 3: The Blocks Editor

3.1 Click on the "Open the Blocks Editor" button to download the “AppInventorForAndroidCodeblocks.jnlp” file, and open it to enter the “Blocks Editor”.

3.2 On the “Blocks Editor” window, place your blocks and it should look like the snapshot below.



3.2.1 Block description：When the Button1 is clicked, send a web request to read AI 1 data of eLogger Shared Memory.

1. My Blocks → Button1 → Button1.Click event block

Perform the contained actions when the button1 is clicked.

2. My Blocks → Web1 → set Web1.Url to event block

Specify the URL to request.

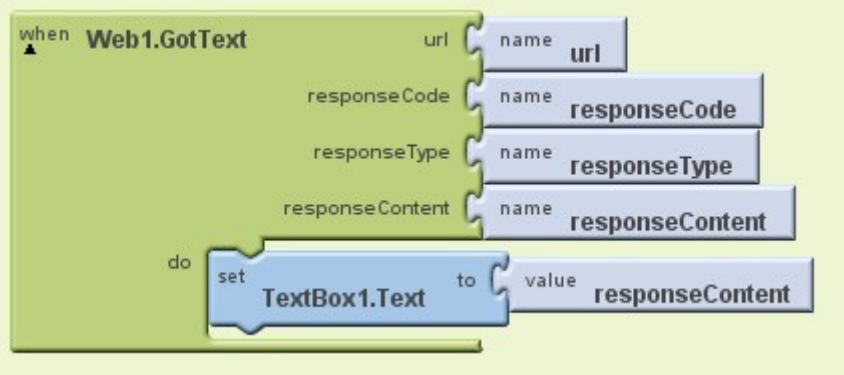
3. Built-In → Text → text parameter

The "text" block indicates the URL. Click on "text" of "text" block and change it to http://192.168.0.1/register.dll?readAI=00001 (the IP address is the PAC's IP address that you want to connect to.)

4. My Blocks → Web1 → call Web1.Get command block

Make the web request.

3.2.2 Block description：When the response to the web request arrives, the Web.GotText event is raised with four parameters. Change the content of "TextBox1" to display the received data which exists in "responseContent" parameter.



1. My Blocks → Web1 → Web1.GotText event block

Specify what to do when the reply comes back from the web.

2. My Blocks → TextBox1 → set TextBox1.Text to event block

Display the result on the screen.

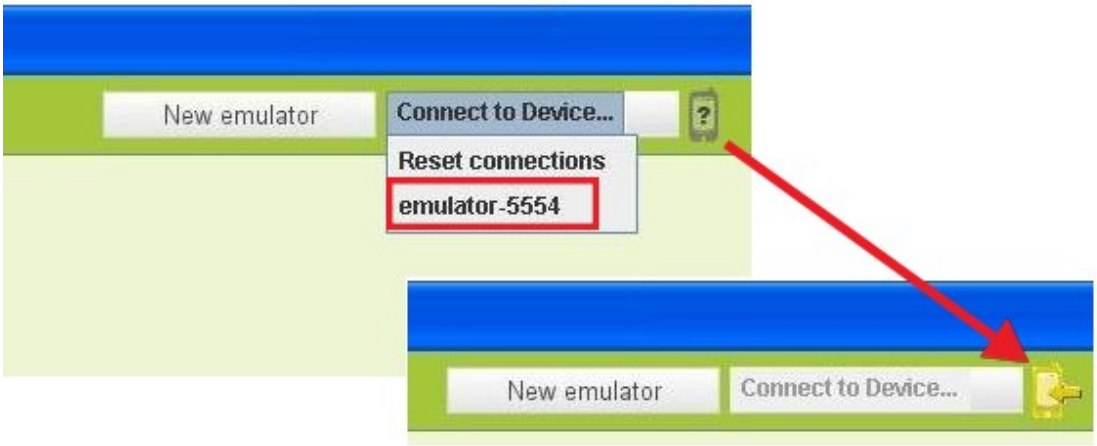
3. My Blocks → My Definitions → responseContent value

The value returned from the web.

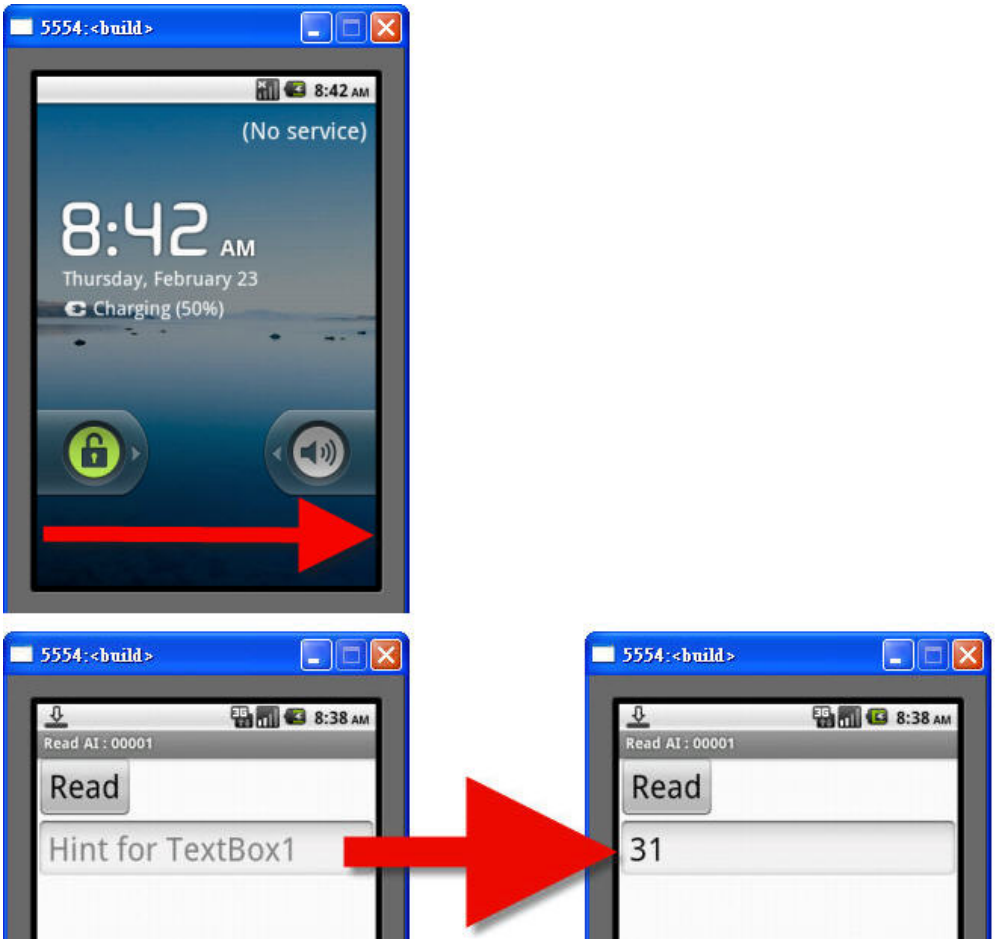
Step 4: To see your app in action

4.1 Click on “New emulator” from the Block Editor window and it will launch an emulator for you. (If you have a phone connected, ignore 4.1.)

4.2 Click on “Connect to Device…”, select "Emulator 5554" or your connected device name , and it will load the app on the specified device.



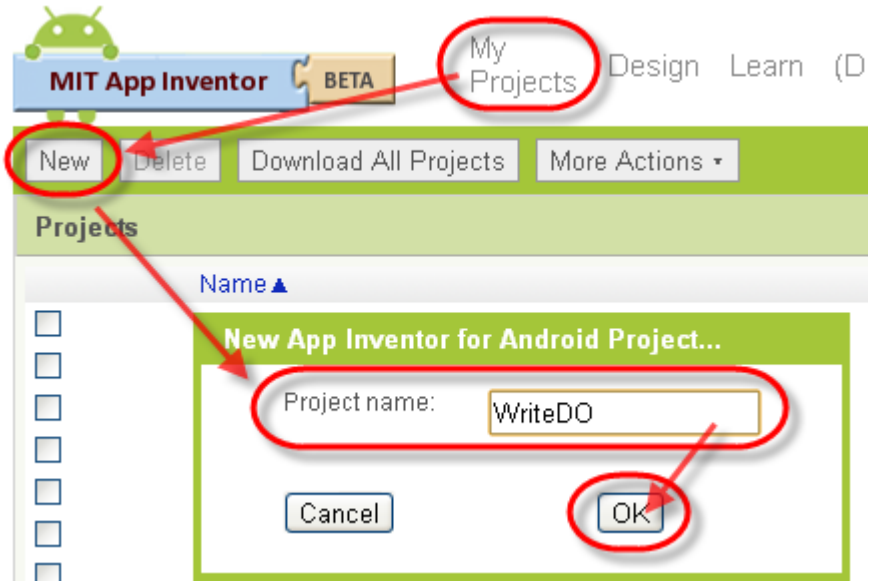
4.3 Unlock the device, and the app will be loaded for a while. Click the “Read” button to read the data from eLogger Shared Memory AI 1.



Example2：Write the data to eLogger Shared Memory DO 0

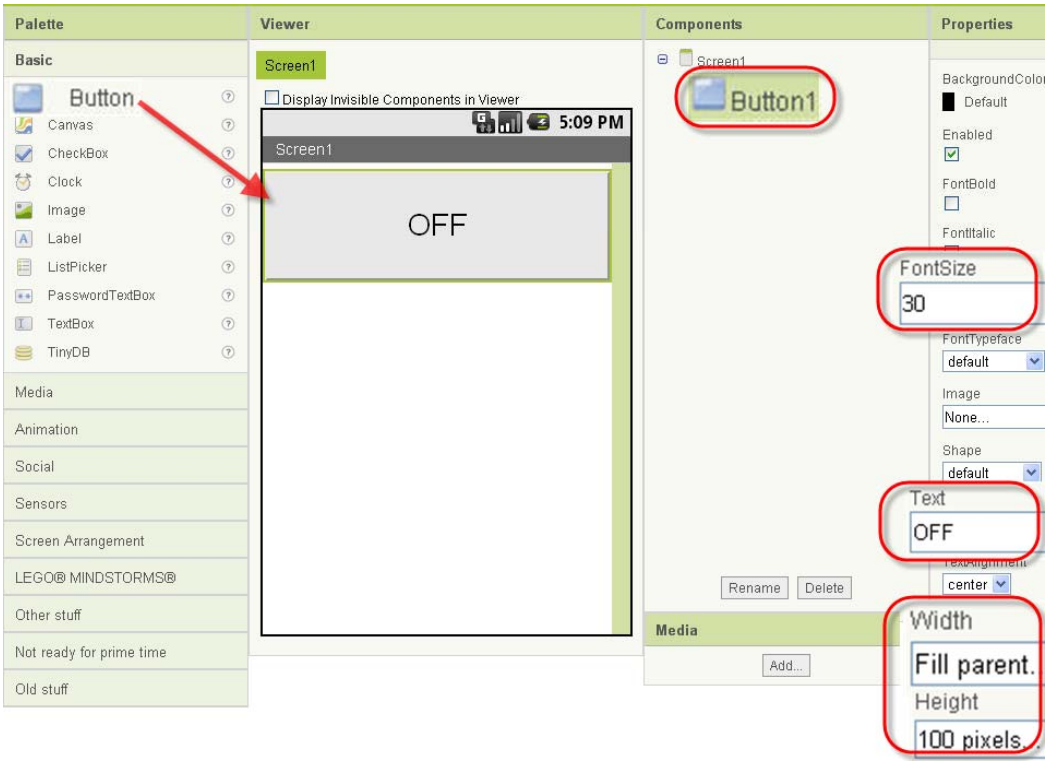
Step 1: Build a new project

Login Gmail → open The Designer → move to “My Projects” page and build a new project named “WriteDO”.

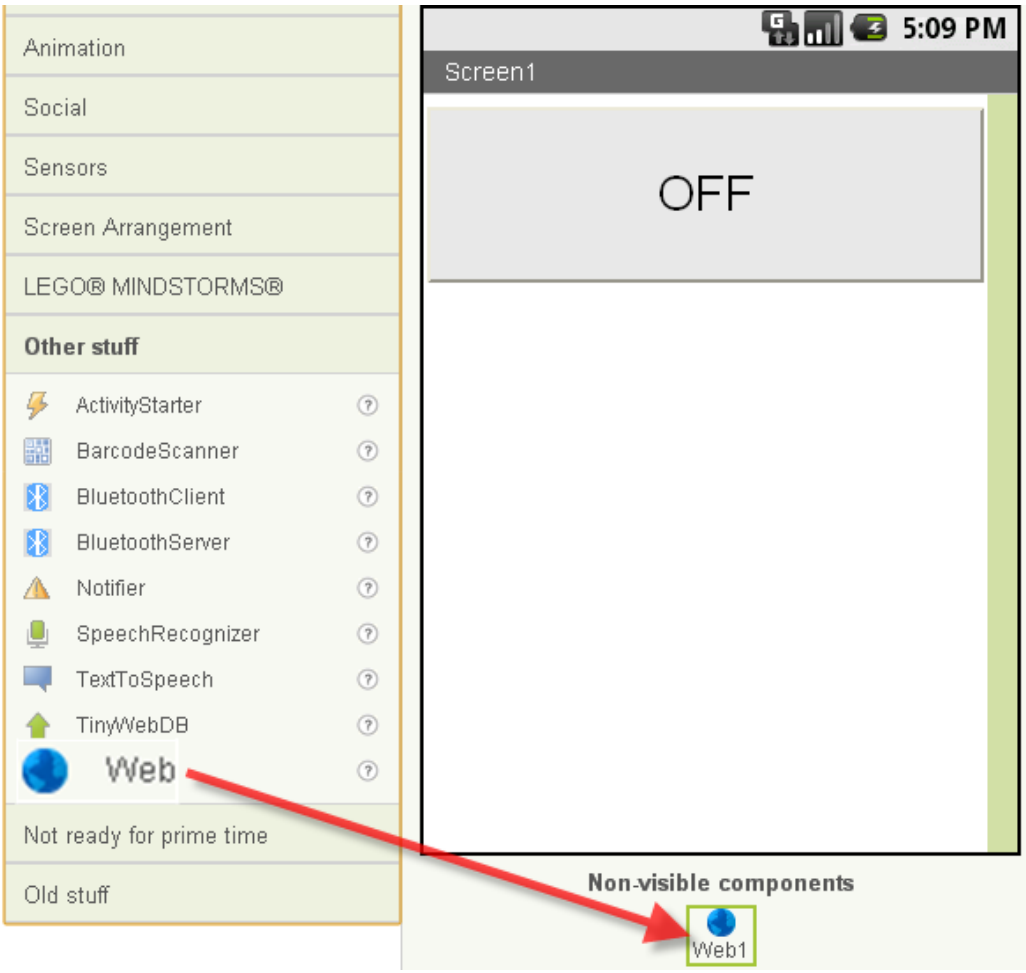


Step 2: The Designer creates the User Interface

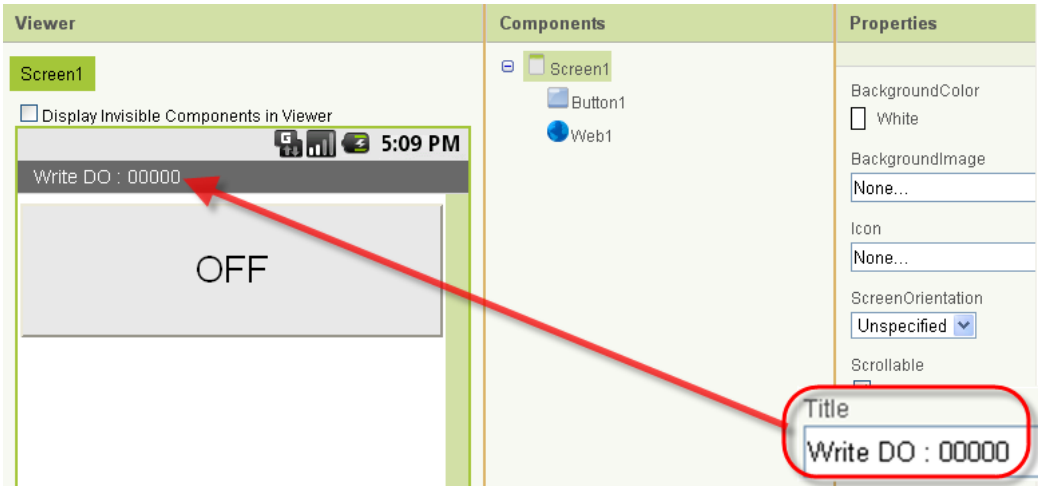
2.1 Select the “Basic” tab, and drag a “Button” component on “Screen1”. Change button’s font size, Width, Height and Text in “Properties” panel.



2.2 Select the “Other stuff” tab, and drag a “Web” component on “Screen1”.



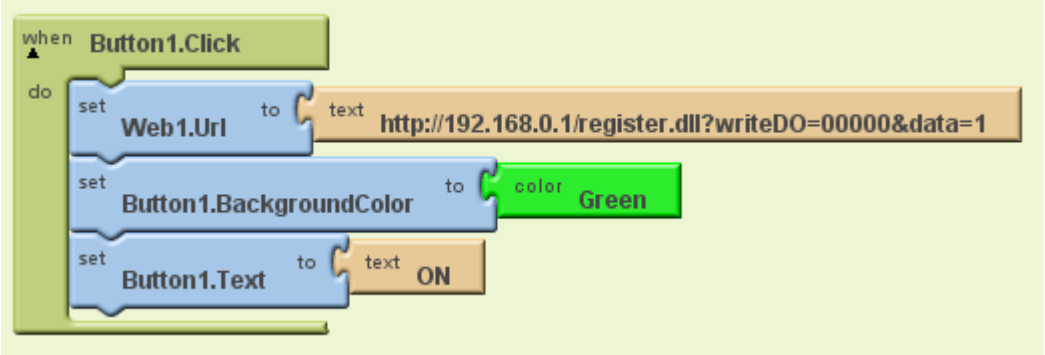
2.3 Change the Screen1’s title (Write DO：00000).



Step 3: The Blocks Editor

3.1 Click on the "Open the Blocks Editor" button to download the “AppInventorForAndroidCodeblocks.jnlp” file, and open it to enter the “Blocks Editor”.

3.2 On the “Blocks Editor” window, place your blocks and it should look like the snapshot below.



3.2.1 Block description：When the Button1 is clicked, send the web request to write data 1 to eLogger Shared Memory DO 0 and change the Button1 ‘s text and background color.

1. My Blocks → Button1 → Button1.Click event block

Perform the contained actions when the Button1 is clicked.

2. My Blocks → Web1 → set Web1.Url event block

Specify the URL to request.

3. Built-in → Text → text parameter

The “text” block indicates the URL. Click on “text” of “text” block and change it to http://192.168.0.1/register.dll?writeDO=00000&data=1 (the IP address is the PAC’s IP address that you want to connect to.)

4. My Blocks → Button1 → set Button1.BackgroundColor to

Display the background color of Button1.

5. Built-In → Colors → color Green value

Indicate green color.

6. My Blocks → Button1 --> set Button1.Text to event block

Set the Button1’s display text.

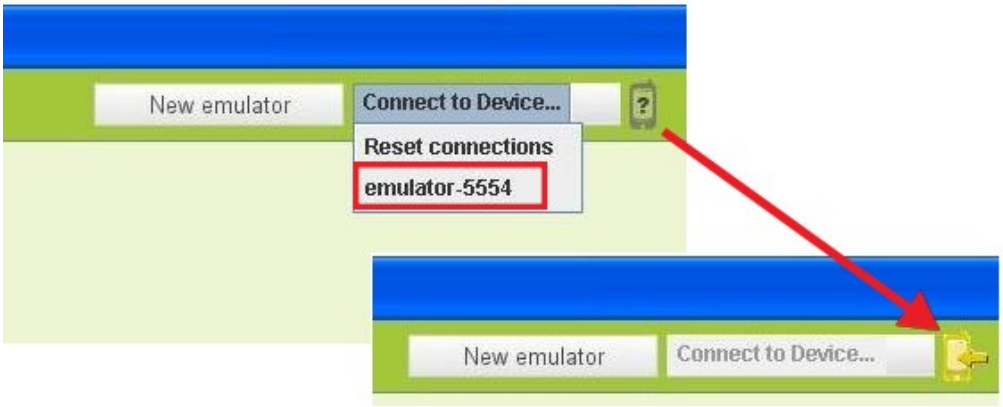
7. My Blocks → Web1 → Call Web1.Get command block

Make the web request.

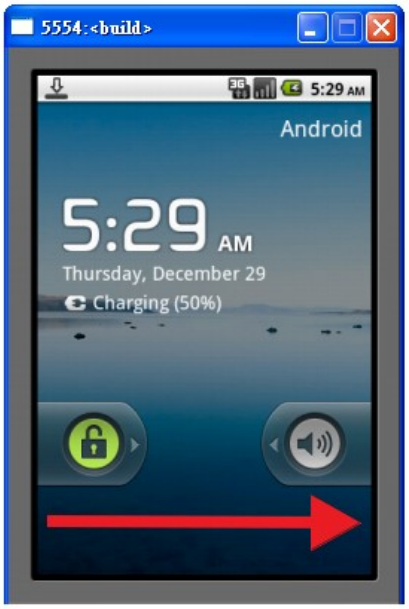
Step 4: To see your app in action

4.1 Click on “New emulator” from the Block Editor window and it will launch an emulator for you. (If you have a phone connected, ignore 4.1)

4.2 Click on “Connect to Device…”, select "Emulator 5554" or your connected device name , and it will load the app on the specified device.



4.3 Unlock the device, and the app will be loaded for a while. Click the button to write the data 1 to eLogger Shared Memory DO 0.



Appendix A. Revision History

Revision	Date	Description
1.0.1	2012/03/06	Initial issue.
1.0.2	2012/10/29	Add the new function description.