# EVB8051 User's Manual

Version 2.0


WIZnet

# COPYRIGHT NOTICE



Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr
General Information: info@wiznet.co.kr
Tel: +82-2-547-9709
Fax: +82-2-547-9711

For more information, visit our website at http://www.wiznet.co.kr

# Table of Contents

# Figures



# Tables

# 1. Getting Started

## 1.1 EVB8051 Package

### 1.1.1 Components

The EVB8051 is packaged as shown in <Fig. 1>, and its contents are as shown in <Fig. 2>.


**<Fig. 1: EVB8051 Package>**

**<Fig. 2: Inside of EVB8051 Package>**

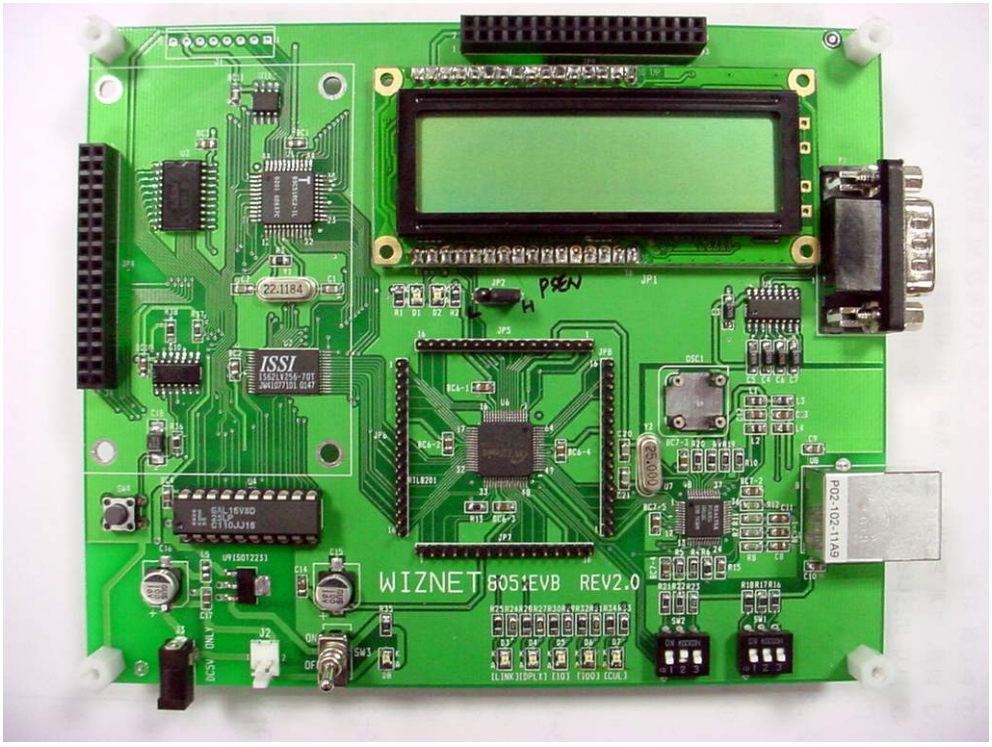
The EVB8051 contains the items described in the table below. Photographs of the items are shown in <Fig. 3> through <Fig. 11>.

**<Table 1: List of Items Contained in the EVB8051>**

| No. | Item | Quantity |
|---|---|---|
| 1 | EVB8051 | 1 |
| 2 | i2Chip W3100A | 5 |
| 3 | User's Manual (EVB8051) | 1 |
| 4 | Power Adaptor (5V) | 1 |
| 5 | Software CD | 1 |
| 6 | UTP Cable | 1 |
| 7 | Serial Cable | 1 |
| 8 | LCD | 1 |


**<Fig. 3: All Items Contained in the EVB8051>**

<Fig. 4> shows the EVB8051 Board.

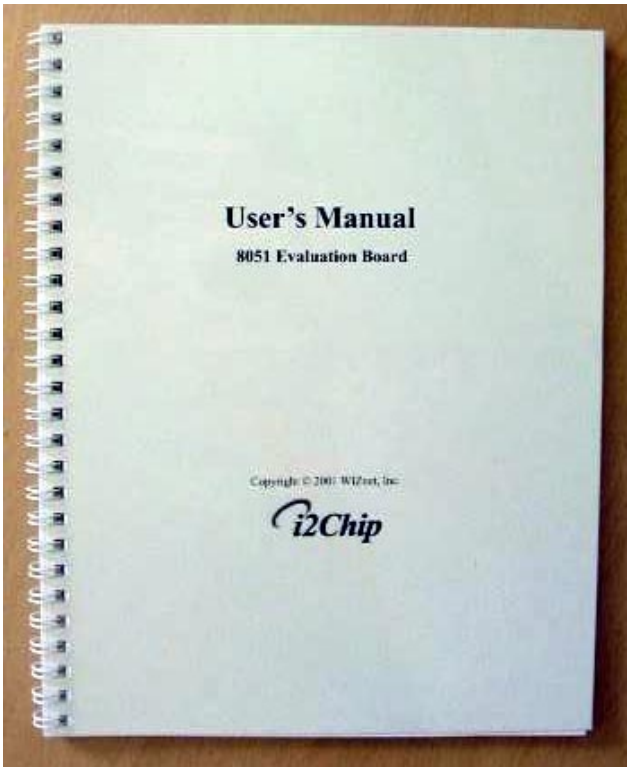



**<Fig. 4: EVB8051>**

<Fig. 5> shows the five W3100A sample chips provided with the EVB8051.


**<Fig. 5: Sample chips of W3100A>**

<Fig. 6> shows the User's Manual of the EVB8051.




**<Fig. 6: User's Manual>**

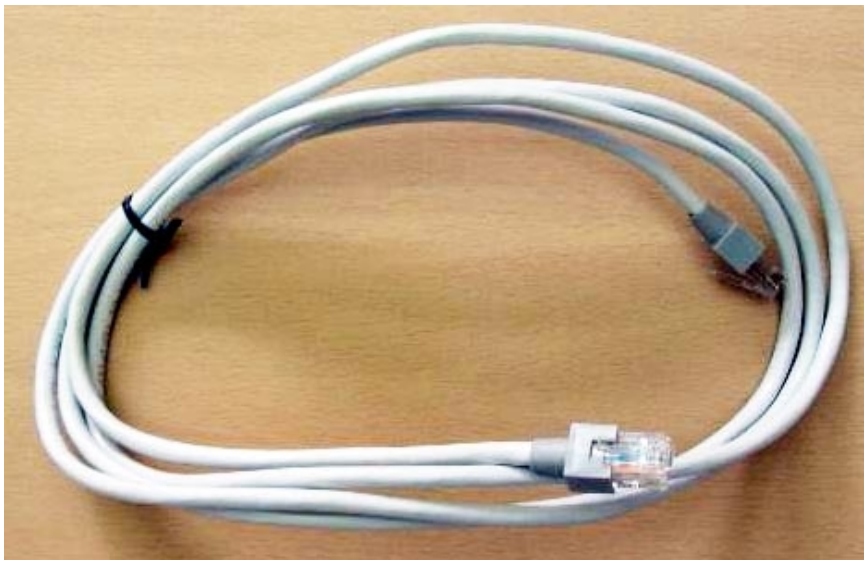
<Fig. 7> shows the 5V Power Adaptor for supplying power to the EVB8051.

**<Fig. 7: Power Adaptor (5V)>**

<Fig. 8> shows the Software CD provided with the EVB8051. It contains Documents, Schematics, Source Code, and Software.

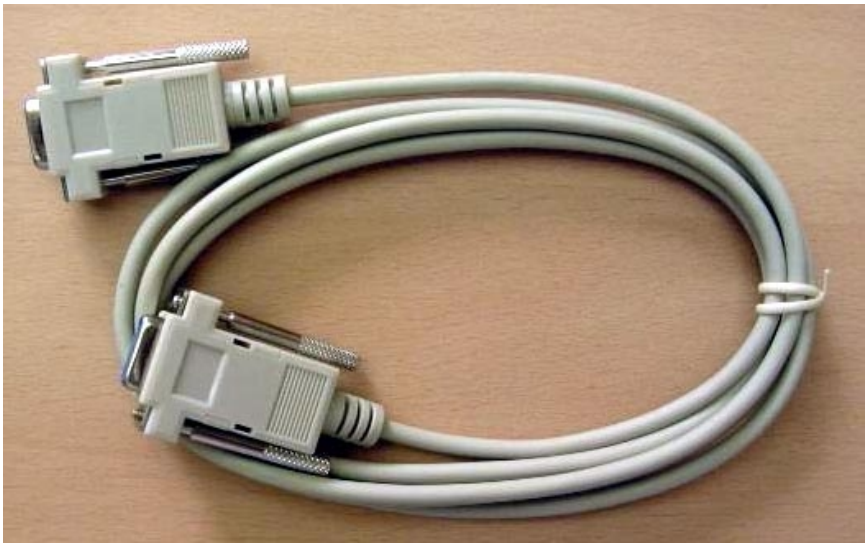

**<Fig. 8: Software CD>**

<Fig. 9> shows the UTP Cable (Twisted Cable) for connecting the EVB8051 directly to the PC.



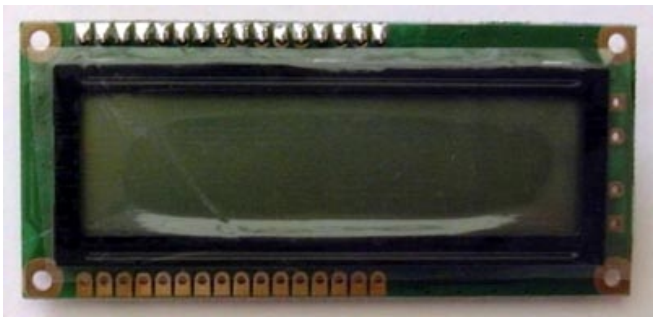
**<Fig. 9: UTP Cable>**

<Fig. 10> shows the Serial Cable (Female-to-Female) for connecting the EVB8051 to the PC. It is used for monitoring and program downloading. Please refer to Appendix B. Specification of serial cable.



**<Fig. 10: Serial Cable>**

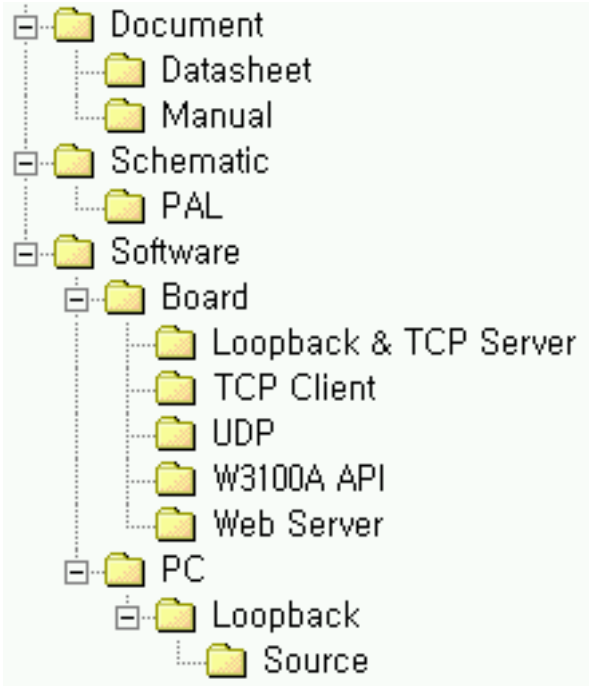
<Fig. 11> shows the LCD provided with the EVB8051. It is useful for testing the functions of the web server and for debugging.



**<Fig. 11: LCD>**

### 1.1.2 Software CD

The EVB8051 is supplied with a Software CD that contains various development tools including Documents, Schematics, Source Code, and Software. <Fig. 12> shows the directory structure of the Software CD.




**<Fig. 12: Directory Structure of the Software CD>**

#### 1.1.2.1 Document

Contains the data sheets of essential parts, including the W3100A data sheet.

#### 1.1.2.2 Schematic

Contains the circuit diagram of the EVB8051. The PAL subdirectory contains the PAL Source that is necessary for interfacing the 8051 MCU and the W3100A.

#### 1.1.2.3 Software

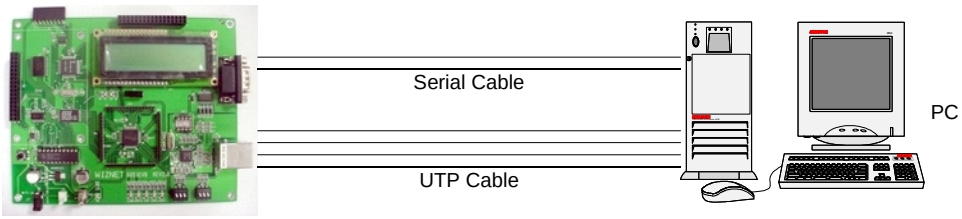
Software is provided for the board and PC applications. The software for the board contains the W3100A API Driver for the 8051 and some samples of application source code. The software for PC applications contains Loopback source code and execution files for Windows.

## 1.2 System Configuration

### 1.2.1 PC Setup

#### 1.2.1.1 Connecting Cables

For testing the functions of the EVB8051 and for application development, the system should be configured as shown in <Fig. 13>. First, the EVB8051 is connected to the PC using the UTP Cable (for data transmission) and the Serial Cable (for monitoring and for program downloading).




**<Fig. 13: System Configuration between EVB8051 and PC>**

#### 1.2.1.2 Network Configuration

For convenience of development, the EVB8051 contains the following default network information:

- IP address: 192.168.0.2
- MAC address: 00-08-DC-00-00-00
- Gateway address: 192.168.0.1
- Subnet Mask: 255.255.255.0

The above information contained in the EVB8051 can be modified at any time to suit the developer's purpose.

First, for testing purposes, set the PC network information as follows:

- IP address: 192.168.0.5
- Gateway address: 192.168.0.1
- Subnet Mask: 255.255.255.0

After the above setup, confirm the operation of the EVB8051 on the PC using the Ping command.

```
C:\> ping 192.168.0.2 –t
```

If the connection has been set up properly, the following message will be displayed on the screen:

```
Pinging 192.168.0.2 with 32 bytes of data:
Reply from 192.168.0.2: bytes=32 time<10ms TTL=128
Reply from 192.168.0.2: bytes=32 time<10ms TTL=128
Reply from 192.168.0.2: bytes=32 time<10ms TTL=128
:
```

If the connection has not been set up properly, the following message will be displayed on the screen:

```
Pinging 192.168.0.2 with 32 bytes of data:
Request timed out.
```

In this case, please refer to Troubleshooting Guide 2.3.1.

#### 1.2.1.3 Program Installation

Since the EVB8051 uses an ATMEL 8051 MCU, you can use the development tools (In-System-Programmer) provided by ATMEL. To download the tools required for development, visit the ATMEL site and download the latest version of the FLIP Software for installation.

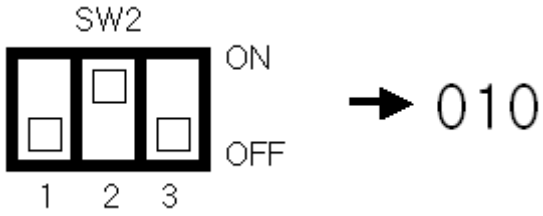
> [8051 – Architecture – Software]
> http://www.atmel.com/atmel/products/prod74.htm
> **FLIP Software** (Download dev_tools3bc6c0cebce3f.zip now. 1.9M, updated Apr 24, 2002)
> FLIP (Flexible In-system Programmer) software v1.6.0. Runs Windows 9x/Me/NT/2000/XP. Supports RS232 or CAN link.

### 1.2.2 Evaluation Board Configuration

#### 1.2.2.1 W3100A mode setting

W3100A provides three modes: non-clocked, clocked, and external clocked mode. DIP switch, SW2 is used for setting all 3-bit values. (For details about each mode value, refer to the explanation for the Mode[2-0] pin on the W3100A Datasheet.) 1 is the MSB (Most Significant Bit), and the bit value is '1' when it is ON or '0' when it is OFF.

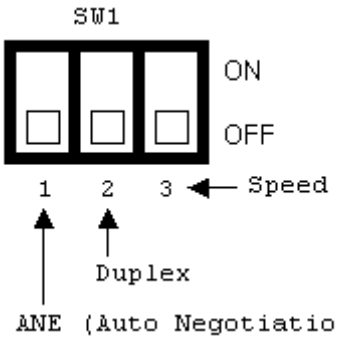
Example: When the mode is set to '010' (Non-clocked mode):




**<Fig. 14: DIP Switch, SW2 for W3100A mode setting>**

#### 1.2.2.2 PHY mode setting

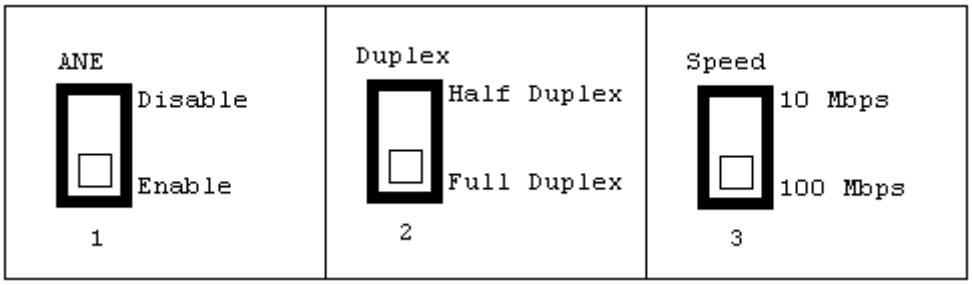
DIP switch SW1 is used for setting the characteristics of the Ethernet PHY chip: negotiation, speed, and duplex.




**<Fig. 15: DIP Switch, SW1 for PHY mode setting>**

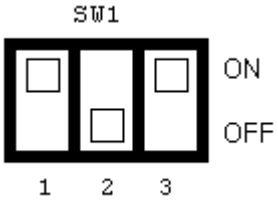
① Switch 1 is used for selecting the ANE (Auto Negotiation) function (Disable/Enable).
② Switch 2 is used for selecting Half or Full Duplex (Half/Full Duplex).
③ Switch 3 is used for selecting the Speed (10/100 Mbps).

The set values for each switch are shown in <Fig. 16>.




**<Fig. 16: Meaning of Each DIP Switch>**

Example: When the mode is set to ANE disable, Full Duplex, and 10 Mbps:

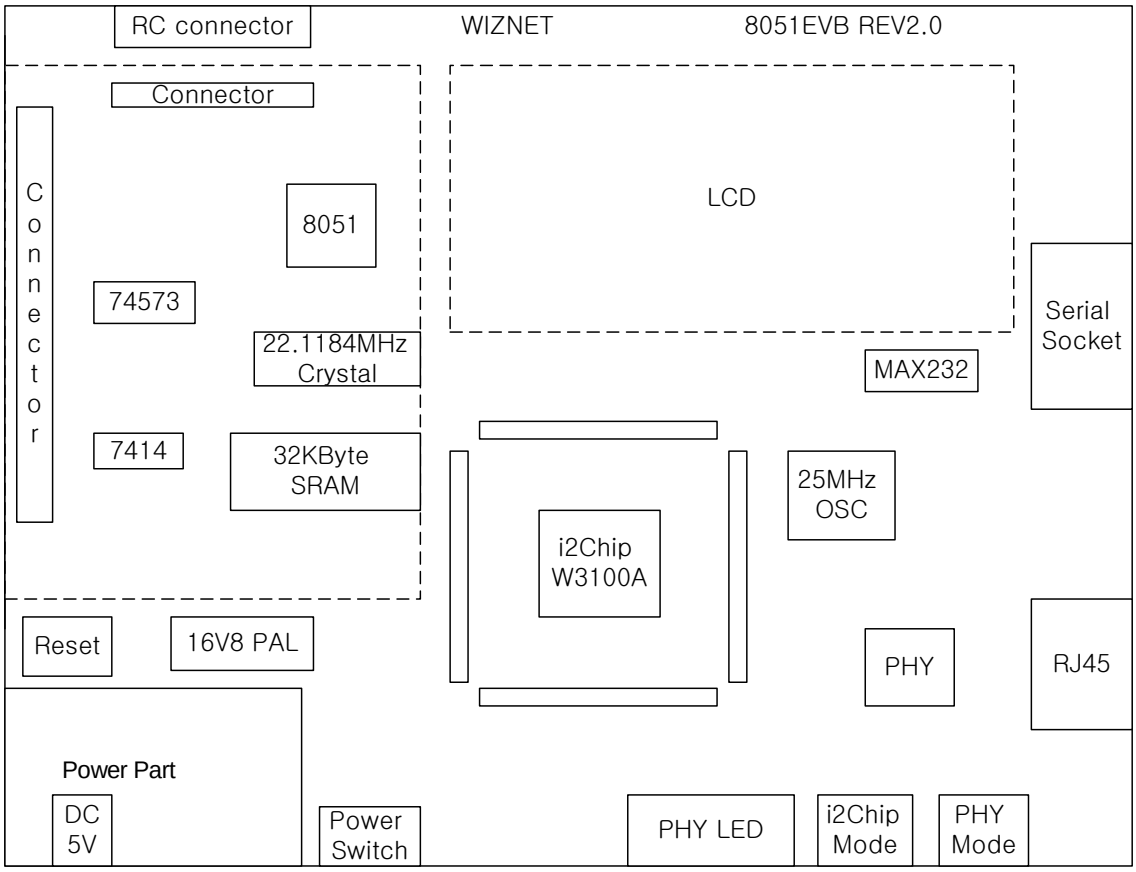



**<Fig. 17: Example setting of SW1>**

# 2. User's Guide

## 2.1 Evaluation Board Layout

<Fig. 18> illustrates the layout of the EVB8051 Board. On the upper left is the processor area that includes the 8051 MCU with 64 Kbytes of flash memory and 32 Kbytes of SRAM. The expansion connector is also located here. On the upper right is the connector and space for connecting the TEXT LCD. The power section is located on the bottom left, which accepts 5V and supplies 5V and 3.3V to the board. On the bottom right are the i2Chip W3100A and the PHY. Also, the PHY LED indicating PHY status and the mode switches for the i2Chip W3100A and the PHY are located here.




**<Fig. 18: Layout of EVB8051>**

## 2.2 Function Testing

### 2.2.1 Loopback Test

The Loopback is the operational mode for measuring the transmission performance of the i2Chip W3100A on the EVB8051. It is used for measuring data transfer speed when the EVB8051 board receives data from the PC and sends it back to the PC.

#### 2.2.1.1 Configuration

Since the EVB8051 board is equipped with default Loopback execution code (TCP Server) in the internal flash memory, its operation can be verified immediately after a network has been set up.

First, install the Axinstall.exe program (located in the "\Software\PC\Loopback\" folder on the CD) on the PC. Once the Axinstall.exe is installed, the Ax1.exe program is created and is required by the PC for Loopback testing. To run the Loopback program loaded on the EVB8051, slide the JP2 Slide Switch on the board to the right.

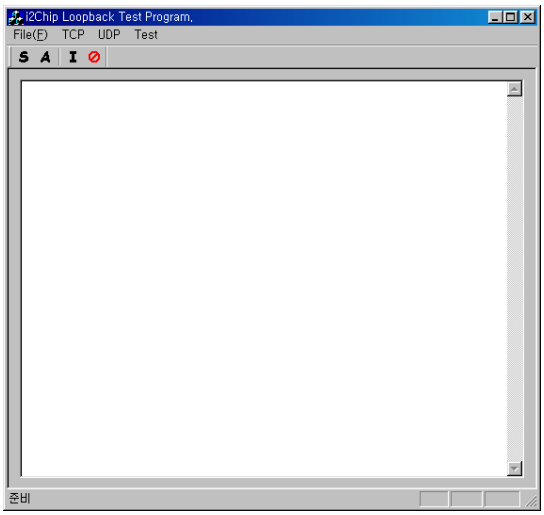
#### 2.2.1.2 Loopback Test

Run the ping command from the PC to the EVB8051 to check network operation.

※ By default, the IP of the EVB8051 is set to 192.168.0.2.

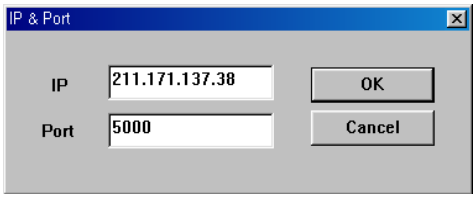
```
C:\> ping 192.168.0.2 –t
```

On the PC, run the AX1.exe program for connection setup. The screen will look like <Fig. 19>.




**<Fig. 19: Running of AX1.exe Program>**

From the 'TCP' menu of the AX1 program, select 'Connect' to display the dialog box as shown in <Fig. 20>. Enter the IP address assigned to the EVB8051 (192.168.0.2) and the Port (5000) number, and try the connection.

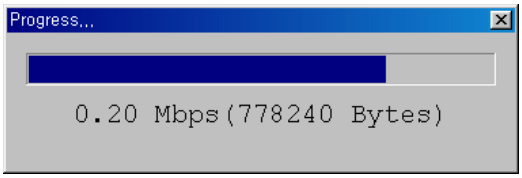



**<Fig. 20: Input of the Connection Information>**

Once a connection is set up between the EVB8051 and the computer, a box with the 'Connected' message appears as shown in <Fig. 21>.

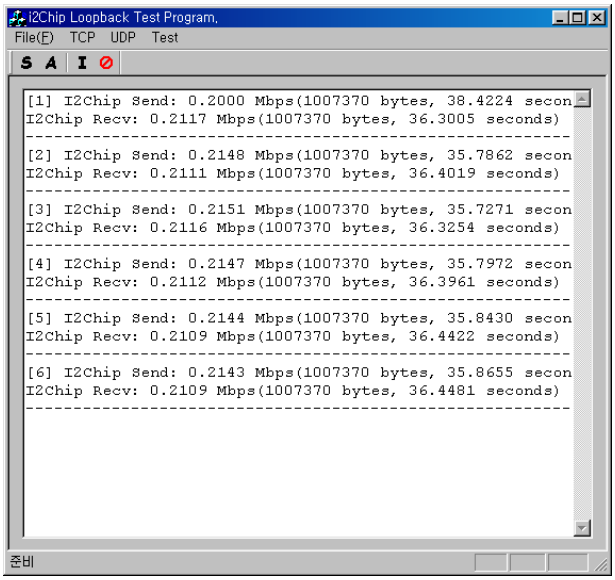

**<Fig. 21: Connection Setup Complete>**

After the connection setup, select 'Send' from the 'File' menu. The dialog box for file transfer appears. Select a file to start the loop back test. Refer to <Fig. 22>.




**<Fig. 22: File Transfer>**

※ You can perform the Loopback test successively using the 'A(uto)' command or the 'I(teration)' command. Make sure to perform the 'S(end)' command before the 'A(uto)' command or the 'I(teration)' command. Refer to <Fig. 23>.




**<Fig. 23: Loopback Test in Succession>**

※ If the program does not run properly, try downloading the loopback program from the Software CD (\Software\Board\Loopback & TCP Server\loopback.hex) into the EVB8051 again.

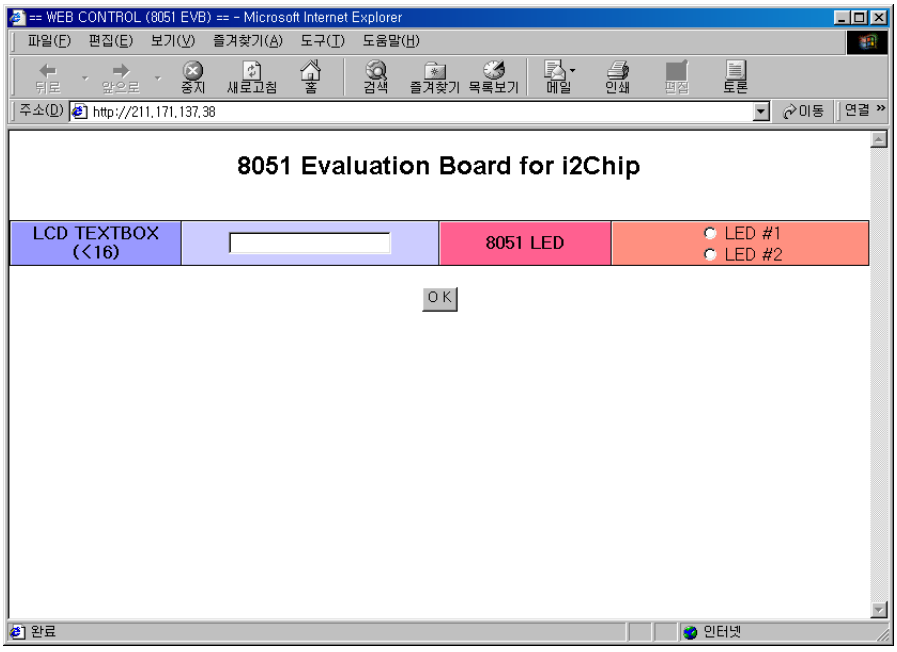
## 2.2.2 Web Server Test

### 2.2.2.1 Outline

The EVB8051 provides the source code to control the equipment through the web and is available for developing applications that require web server functions.

### 2.2.2.2 Testing Procedure

The test method for the web server is the same as for the Loopback test.
Download the web server program "\Software\Board\Web Server\webserv.hex" from the Software CD to the EVB8051 for testing.
Run the ping command to the EVB8051 to check network operation. By default, the IP address of the EVB8051 is set to 192.168.0.2.
If the Ping command works properly, run the web browser on the PC and enter the IP address of the EVB8051 (192.168.0.2) in the URL window to attempt to access the web server of the EVB8051.
If the EVB8051 is running in web server mode, the starting screen of the web page will look like <Fig. 20>.




**<Fig. 24: Starting Screen for the Demo Web Page of the EVB8051>**

#### 2.2.2.3 Functions of the Demo Web Page

(1) LCD Character Display

Entering characters in the LCD Text Box on the demo page will display the characters on the LCD of the EVB8051. (The LCD should be installed separately on the EVB8051.)

(2) LED Remote Control

It controls the LEDs on the board through the web. In actual applications, it can be used for controlling other devices than the LED in remote places through the web. Selecting LED#1 and LED#2 in <Fig. 20> will turn the LEDs (D1, D2) on the EVB8051 ON/OFF.

## 2.3 Troubleshooting Guide

### 2.3.1 Ping

When you cannot reach EVB8051 by Ping command,

Step 1. Did you connect correctly between test PC and EVB8051 with UTP cable?

Step 2. Did you change your test PC's network environment (IP address, Gateway, Subnet)?

If no, you should change it first as follows:

IP address: 192.168.0.5

Gateway address: 192.168.0.1

Subnet Mask: 255.255.255.0

Step 3. Whether EVB8051's Link LED is ON?

If off, you'd better check whether the UTP cable works correctly.

### 2.3.2 Misc.

#### 2.3.2.1 When the screen remains blank with the power on after a connection is made

Step 1. Check the connection condition of the serial cable.

Step 2. Check if the COM Port numbers of the PC and terminal coincide.

Step 3. Check the terminal configuration.

# 3. Programmer's Guide

## 3.1 API Function

### 3.1.1 Type of Functions

(1) Internal Function: Used inside the driver function

(2) API Function: Used in applications

| Function Name | void Int0(void) interrupt 0 |
|---|---|
| Arguments | None |
| Return value | None |
| Description | Interrupt handling function of the W3100A.<br>Stores the status information that each function waits for in the global variable S_STATUS for transfer.<br>S_STATUS stores the interrupt status value for each channel. |
| Category | Internal Function |

| Function Name | void ISR_ESTABLISHED(SOCKET s) |
|---|---|
| Arguments | s: Channel number |
| Return value | None |
| Description | Established connection interrupt handling function.<br>Called upon connection establishment, and may be inserted in user code if needed by the programmer. |
| Category | Internal Function |

| Function Name | void ISR_CLOSED(SOCKET s) |
|---|---|
| Arguments | s: Channel number |
| Return value | None |
| Description | Closed connection interrupt handling function.<br>Called upon connection closure, and may be inserted in user code if needed by the programmer. |
| Category | Internal Function |

| Function Name | void ISR_RX(SOCKET s) |
|---|---|
| Arguments | s: Channel number |
| Return value | None |
| Description | Received data interrupt handling function.<br>Called upon receiving data, and may be inserted in user code if needed by the programmer. |
| Category | Internal Function |

| Function Name | void initW3100A(void) |
|---|---|
| Arguments | None |
| Return value | None |
| Description | W3100A initialization function.<br>Function for S/W resetting of the W3100A.<br>Sets the initial SEQ# to be used for TCP communication. |
| Category | API Function |

| Function Name | void sysinit(void) |
|---|---|
| Arguments | None |
| Return value | None |
| Description | W3100A initialization function.<br>Sets the source MAC, source IP, gateway, and subnet mask to be used by the W3100A to the designated values.<br>May be called when setting the concerned register to modify network information and reflect it on the W3100A. |
| Category | API Function |

| Function Name | void setsubmask(u_char * addr) |
|---|---|
| Arguments | addr: Pointer having the value for setting up the subnet mask |
| Return value | None |
| Description | Subnet mask setup function |
| Category | API Function |

| Function Name | void setgateway(u_char * addr) |
|---|---|
| Arguments | addr: Pointer having the value for setting up the gateway IP |
| Return value | None |
| Description | Gateway IP setup function |
| Category | API Function |

| Function Name | void setIP(u_char * addr) |
|---|---|
| Arguments | addr: Pointer having the value for setting up the source IP address |
| Return value | None |
| Description | W3100A IP address setup function |
| Category | API Function |

| Function Name | void setMACAddr(u_char * addr) |
|---|---|
| Arguments | addr: Pointer having the value for setting up the MAC address |
| Return value | None |
| Description | MAC address setup function |
| Category | API Function |

| Function Name | void settimeout(u_char * val) |
|---|---|
| Arguments | val: Pointer having the value for setting up the timeout.<br>Upper 2 bytes have the initial timeout value, while the last 1 byte has the number of retransmissions until timeout. |
| Return value | None |
| Description | TCP timeout setup function.<br>Used for adjusting the TCP retransmission time.<br>A timeout interrupt takes place when retransmission is attempted for establishing the connection or for data transfer beyond the set value. |
| Category | API Function |

| Function Name | void setINTMask(u_char mask) |
|---|---|
| Arguments | mask: Value of the mask to be set (‘1’ refers to interrupt enable) |
| Return value | None |
| Description | Interrupt mask setup function.<br>Enables/disables the concerned interrupt. |
| Category | API Function |

| Function Name | void setbroadcast(SOCKET s) |
|---|---|
| Arguments | s: Channel number |
| Return value | None |
| Description | Broadcast data transfer enable setup function<br>Enables/disables broadcasting data transfer in UDP or IP RAW mode. |
| Category | API Function |

| Function Name | **void setTOS(SOCKET s, u_char tos)** |
|---|---|
| **Arguments** | s: Channel number<br>tos: Value to be set for the TOS field of the IP header |
| **Return value** | None |
| **Description** | Handles protocol setup function in IP RAW mode |
| **Category** | API Function |

| Function Name | **char socket(SOCKET s, u_char protocol, u_int port, u_char flag)** |
|---|---|
| **Arguments** | s: Channel number<br>protocol: Protocol designated for the channel<br>SOCK_STREAM(0x01) -> TCP<br>SOCK_DGRAM(0x02) -> UDP<br>SOCK_IPL_RAW(0x03) -> IP Layer RAW<br>SOCK_MACL_RAW(0x04) -> MAC Layer RAW<br>port: Source port designated for the channel<br>flag: Options designated for the channel<br>SOCKOPT_BROADCAST(0x80) -> '1' refers to broadcast data transfer in UDP mode<br>SOCKOPT_NDTIMEOUT(0x40) -> '1' refers to use of only the register that designates the timeout value<br>SOCKOPT_NDACK(0x20) -> '1' refers to the delayed ACK not to be used<br>SOCKOPT_SWS(0x10) -> '1' refers to the silly window syndrome to be used |
| **Return value** | Channel number if succeeded, or –1 if failed. |
| **Description** | Initialization of the channel.<br>Initializes the designated channel and waits for completion of W3100A handling. |
| **Category** | API Function |

| Function Name | **char connect(SOCKET s, u_char * addr, u_int port)** |
|---|---|
| **Arguments** | s: Channel number<br>addr: Destination IP address<br>port: Destination port number |
| **Return value** | 1 if connection is established, or –1 if connection fails. |
| **Description** | Sets the connection to the designated peer.<br>Establishes a connection with a peer on the designated channel and waits until the connection is established. (TCP client mode) |
| **Category** | API Function |

| Function Name | **char listen(SOCKET s, u_char * addr, u_int * port)** |
|---|---|
| **Arguments** | s: Channel number<br>addr: Peer IP address at the time of connection establishment<br>port: Peer Port number at the time of connection establishment |
| **Return value** | 1 if connection is established, or –1 if connection fails. |
| **Description** | Waits for connection with a peer. (Blocking Mode)<br>The designated channel waits for connection by a peer. (TCP Server mode) |
| **Category** | API Function |

| Function Name | char NBlisten(SOCKET s) |
|---|---|
| **Arguments** | s: Channel number |
| **Return value** | 1 |
| **Description** | Waits for connection with a peer. (Non-blocking Mode)<br>The designated channel waits for connection by a peer. (TCP Server mode) |
| **Category** | API Function |

| Function Name | void initseqnum(SOCKET s) |
|---|---|
| **Arguments** | s: Channel number |
| **Return value** | None |
| **Description** | Generates random values for the initial SEQ# to be used for establishing a TCP connection.<br>This function may be added to the code for generating random numbers for assigning a random number to initial SEQ# used in TCP.<br>In an actual internet environment, the initial SEQ# must be a random number. (A fixed number is used for EVB/DK.) |
| **Category** | API Function |

| Function Name | u_int send(SOCKET s, u_char * buf, u_int len) |
|---|---|
| **Arguments** | s: Channel number<br>buf: Pointer indicating the data to be sent<br>len: Size of the data to be sent |
| **Return value** | Sent data size |
| **Description** | Function for sending TCP data.<br>Composed of the send()and send_in() functions.<br>The send() function is an application I/F function. It continues to call the send_in() function to complete the sending of the data up to the size of the data to be sent when the application is called. The send_in() function receives the return value (the size of the data sent), calculates the size of the data to be sent, and calls the send_in() function again if there is any data left to be sent. |
| **Category** | API Function |

| Function Name | u_int send_in(SOCKET s, u_char * buf, u_int len) |
|---|---|
| Arguments | s: Channel number<br>buf: Pointer indicating the data to be sent<br>len: Size of the data to be sent |
| Return value | Sent data size |
| Description | Internal function for sending TCP data.<br>Called by the send() function for TCP transmission.<br>It first calculates the free transmit buffer size and compares it with the size of the data to be transmitted to determine the transmission size.<br>After calculating the data size, it copies data from TX_WR_PTR.<br>It waits if there is a previous send command in process.<br>When the send command is cleared, it updates the TX_WR_PTR up to the size to be transmitted and performs the send command. |
| Category | Internal Function |

| Function Name | u_int recv(SOCKET s, u_char * buf, u_int len) |
|---|---|
| Arguments | s: Channel number<br>buf: Pointer where the data to be received is copied<br>len: Size of the data to be received |
| Return value | Received data size |
| Description | TCP data receiving function.<br>The recv() function is an application I/F function. It continues to wait for as much data as the application wants to receive. |
| Category | API Function |

| Function Name | u_int sendto(SOCKET s, const u_char * buf, u_int len, u_char * addr, u_int port) |
|---|---|
| Arguments | s: Channel number<br>buf: Pointer indicating the data to send<br>len: Size of the data to send<br>addr: Destination IP address |
| Return value | Sent data size |
| Description | UDP data sending function.<br>Composed of the sendto()and sendto_in() functions.<br>The send() function is an application I/F function. It continues to call the send_in() function to complete the sending of the data up to the size of the data to be sent when the application is called.<br>Unlike TCP transmission, it designates the destination address and the port. |
| Category | API Function |

| Function Name | u_int sendto_in(SOCKET s, const u_char * buf, u_int len) |
|---|---|
| Arguments | s: Channel number<br>buf: Pointer indicating the data to send<br>len: Size of the data to send |
| Return value | Sent data size |
| Description | UDP data sending function.<br>An internal function that is the same as the send_in() function of the TCP. |
| Category | Internal Function |

| Function Name | u_int recvfrom(SOCKET s, u_char * buf, u_int len, u_char * addr, u_int * port) |
|---|---|
| Arguments | s: Channel number<br>buf: Pointer where the data to be received is copied<br>len: Size of the data to be received<br>addr: Peer IP address for receiving<br>port: Peer port number for sending |
| Return value | Received data size |
| Description | UDP data receiving function.<br>Function for receiving UDP and IP layer RAW mode data, and handling the data header. |
| Category | API Function |

| Function Name | char close(SOCKET s) |
|---|---|
| Arguments | s: Channel number |
| Return value | 1 |
| Description | Channel closing function.<br>Function for closing the connection of the designated channel. |
| Category | API Function |

| Function Name | u_int select(SOCKET s, u_char func) |
|---|---|
| Arguments | s: Channel number<br>func: SEL_CONTROL(0x00) -> return socket status<br>SEL_SEND(0x01) -> return free transmit buffer size<br>SEL_RECV(0x02) -> return data size in receive buffer |
| Return value | Socket status or free transmit buffer size or received data size |
| Description | Function handling the channel socket information. |
| Category | API Function |

| Function Name | u_int read_data(SOCKET s, u_char * src, u_char * dst, u_int len) |
|---|---|
| Arguments | s: Channel number<br>src: Receive buffer pointer of the W3100A<br>dst: System buffer pointer<br>len: Data size to be copied |
| Return value | Copied data size |
| Description | Copies the receive buffer data of the W3100A to the system buffer.<br>It is called from the recv()or recvfrom() function. |
| Category | Internal Function |

| Function Name | u_int write_data(SOCKET s, u_char * src, u_char * dst, u_int len) |
|---|---|
| Arguments | s: Channel number<br>src: System buffer pointer<br>dst: Transmit buffer pointer of the W3100A<br>len: Data size to be copied |
| Return value | Copied data size |
| Description | Copies the system buffer data to the transmit buffer of the W3100A.<br>It is called from the send_in()or sendto_in() function. |
| Category | Internal Function |

| Function Name | void wait_10ms(int cnt) |
|---|---|
| Arguments | cnt: count |
| Return value | None |
| Description | Designates the delay.<br>Waits for 10 milliseconds. |
| Category | Internal Function |

| Function Name | void wait_1ms(int cnt) |
|---|---|
| Arguments | cnt: count |
| Return value | None |
| Description | Designates the delay.<br>Waits for 1 millisecond. |
| Category | Internal Function |

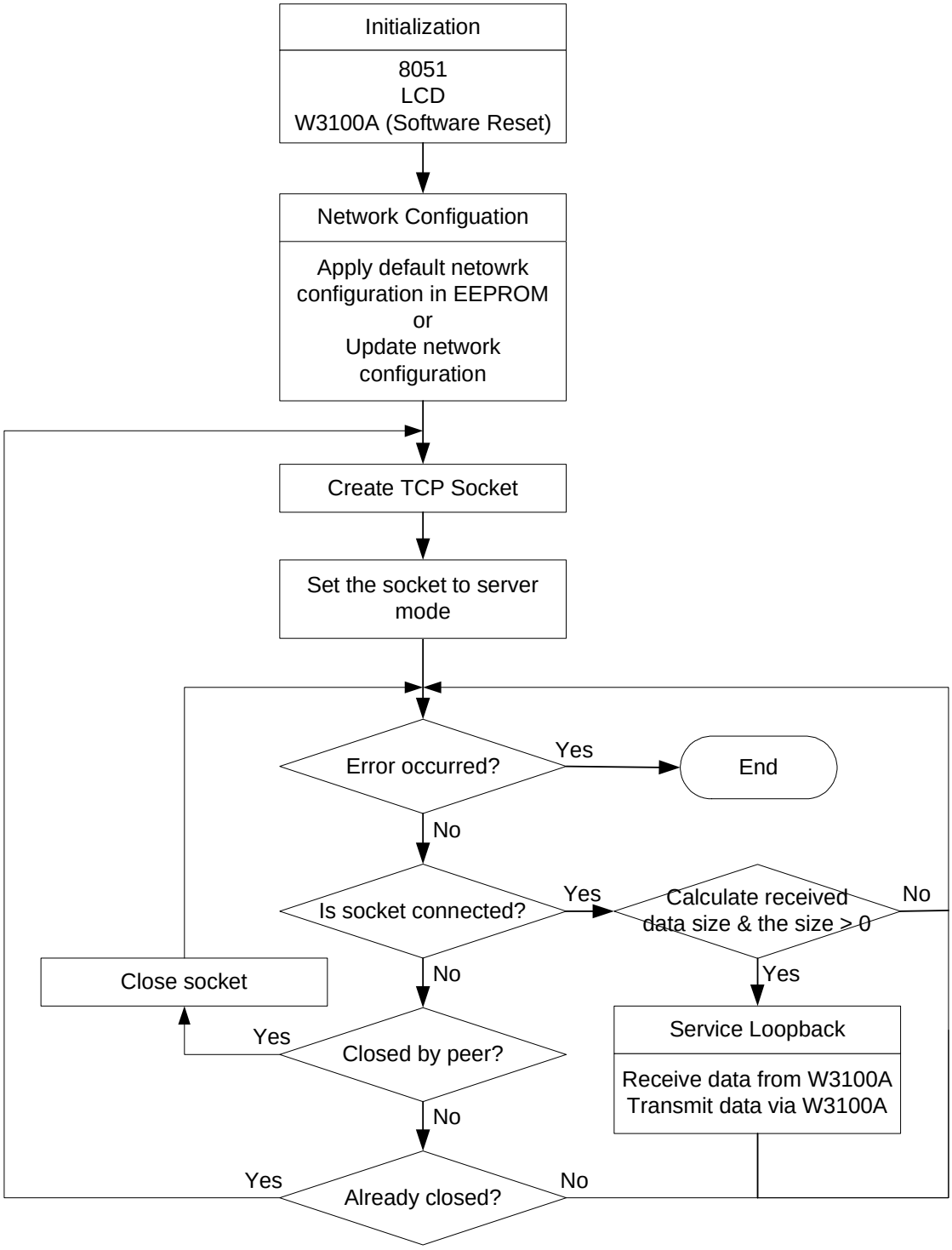
| Function Name | void wait_1us(int cnt) |
|---|---|
| Arguments | cnt: count |
| Return value | None |
| Description | Designates the delay.<br>Waits for 1 millisecond. |
| Category | Internal Function |

## 3.2 Sample Source Codes

### 3.2.1 Loopback & TCP Server

#### 3.2.1.1 Source Codes : \Software\Board\Loopback & TCP Server\

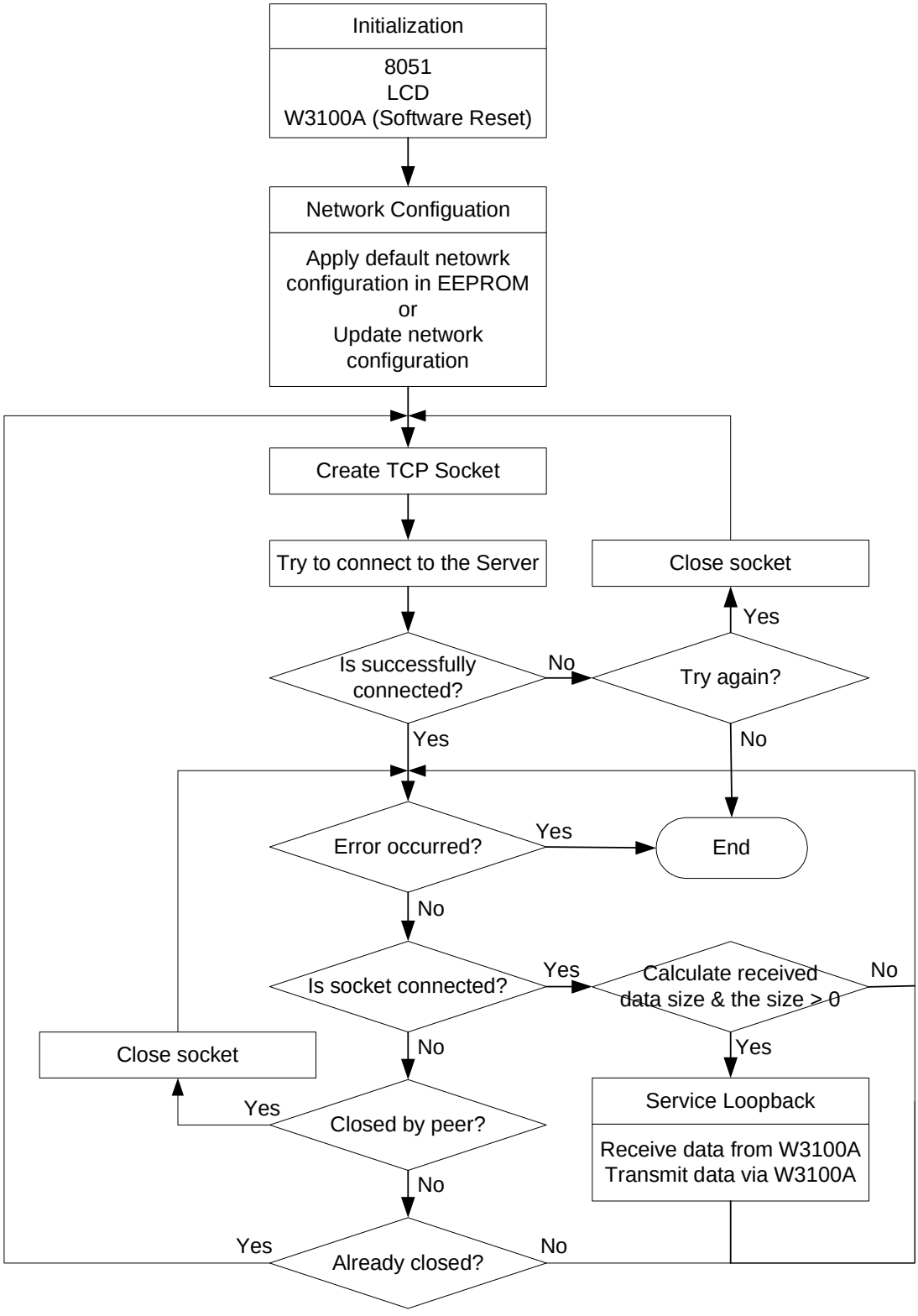
#### 3.2.1.2 Flow Diagram



### 3.2.2 TCP Client

#### 3.2.2.1 Source Codes : \Software\Board\TCP Client\

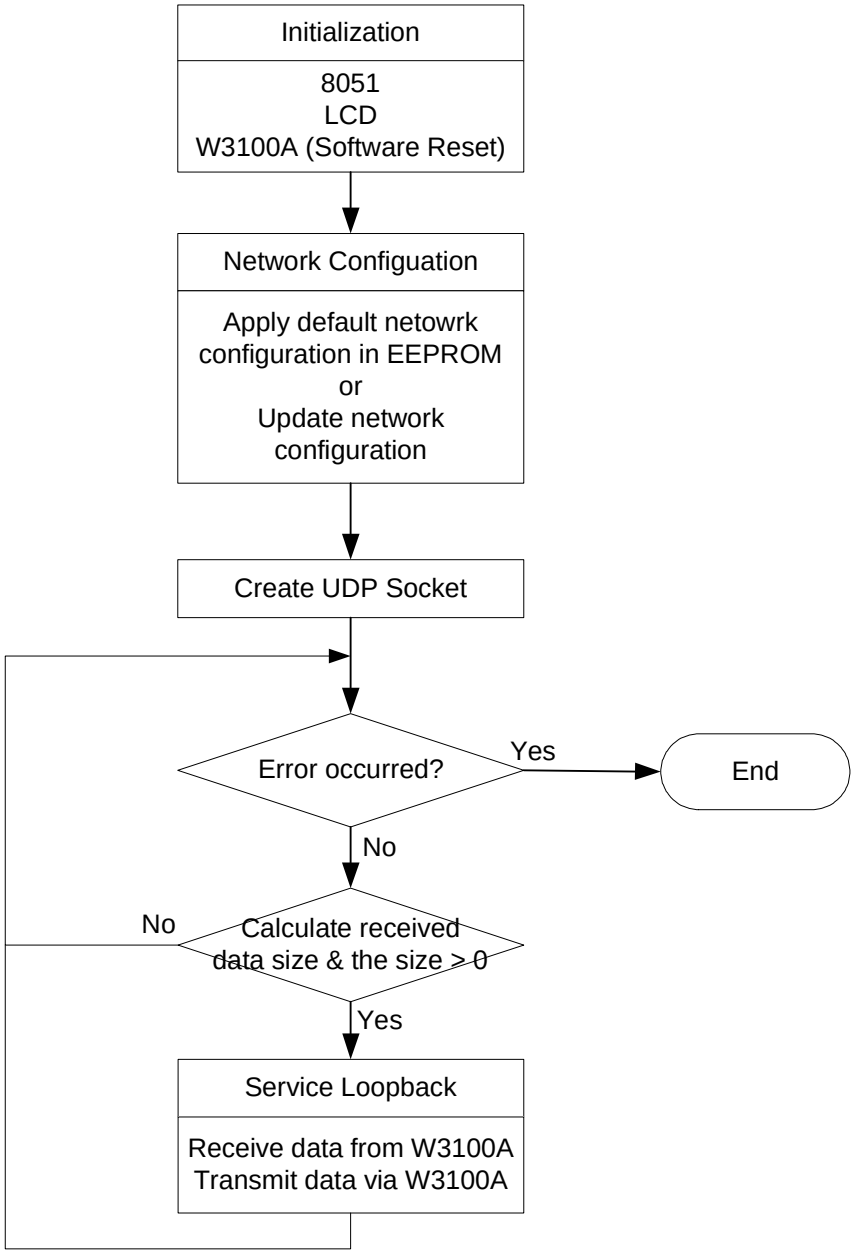
#### 3.2.2.2 Flow Diagram



### 3.2.3 UDP

#### 3.2.3.1 Source Codes : \Software\Board\UDP\

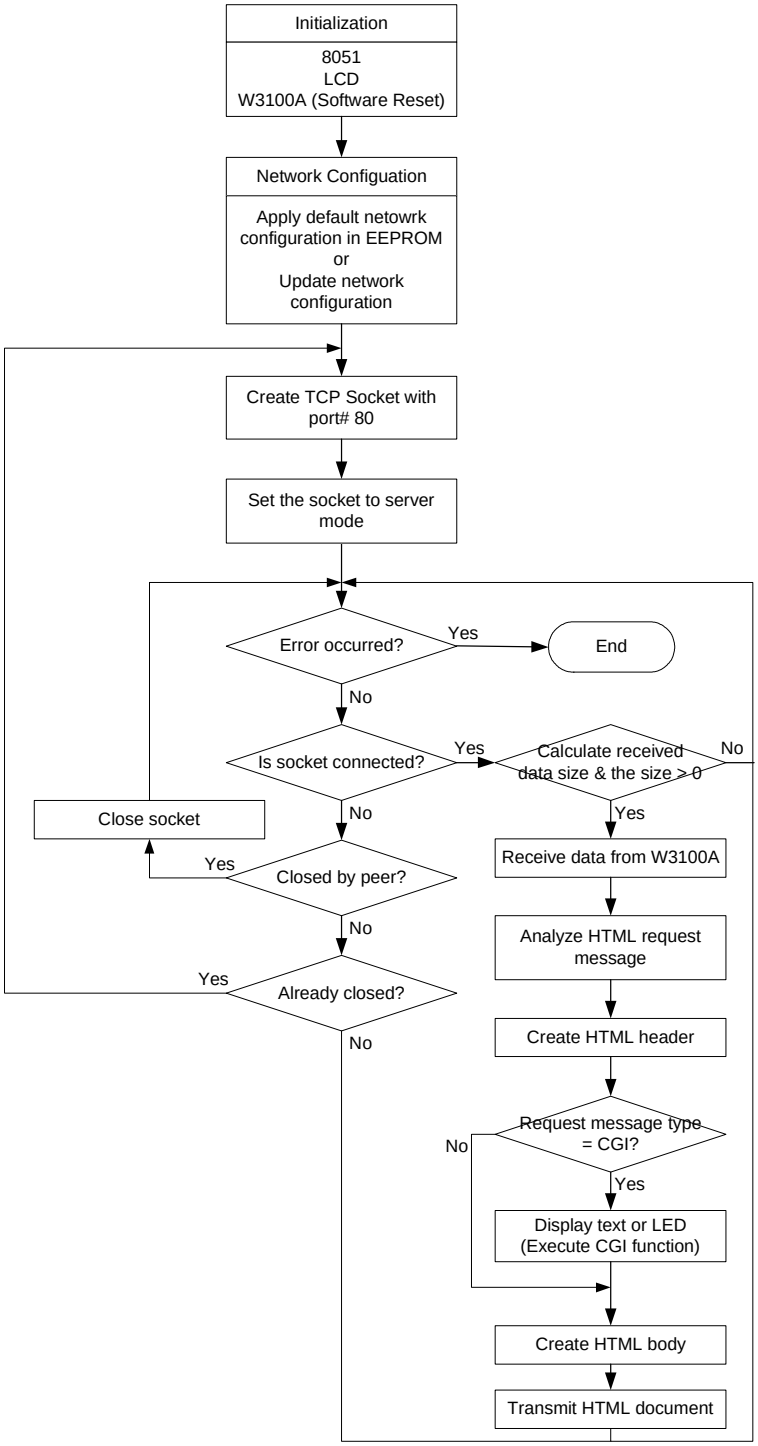
#### 3.2.3.2 Flow Diagram



### 3.2.4 Web Server

#### 3.2.4.1 Source Codes : \Software\Board\Web Server\

#### 3.2.4.2 Flow Diagram

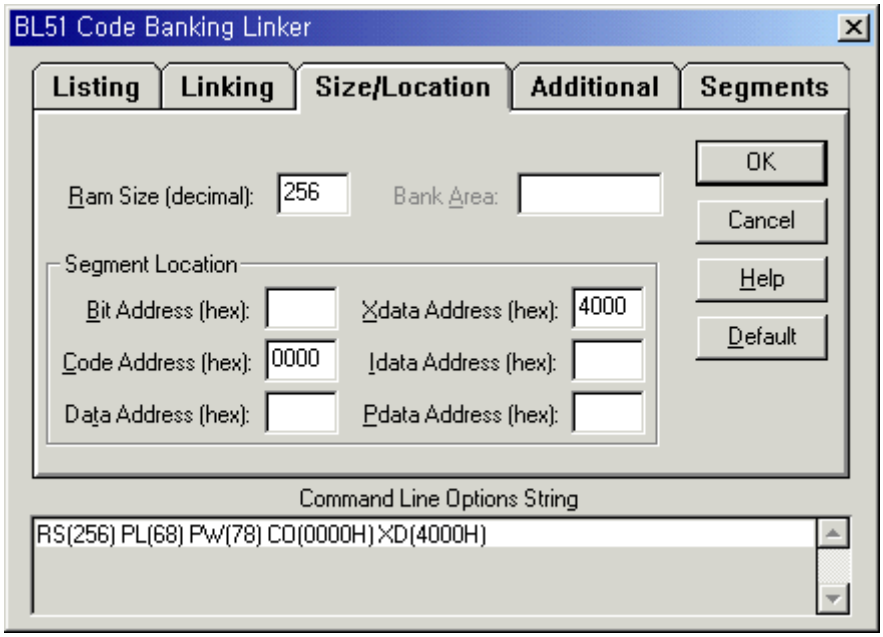


## 3.3 Application Development Procedure

### 3.3.1 Program Developing Procedure (based on the KEIL compiler)

#### 3.3.1.1 Configuration

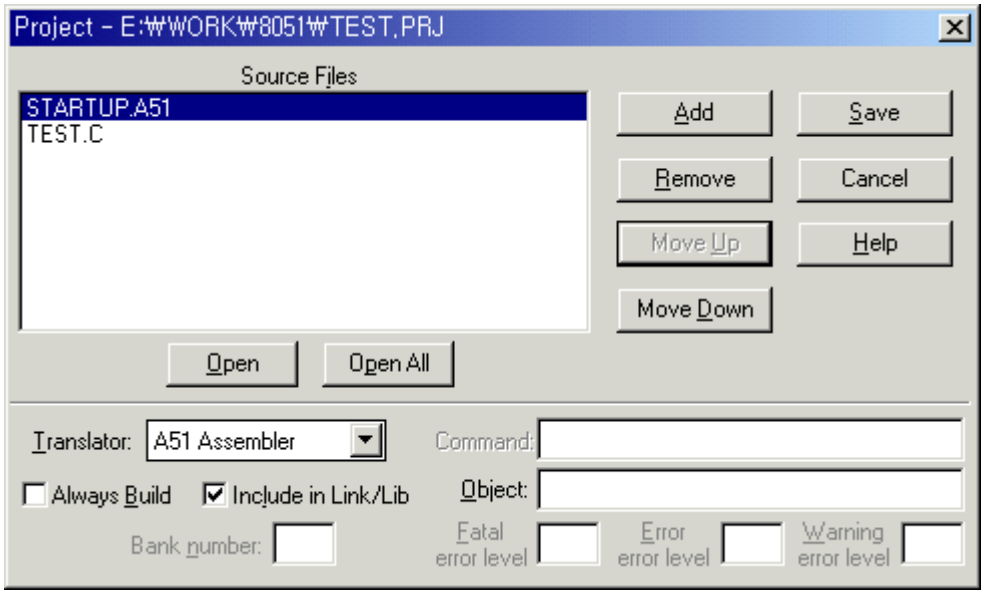
1. Run uVision-51.
2. In Options->BL51 Code Banking Linker, set the Xdata Address arbitrarily in the SRAM area and the Code Address to 0. (Refer to the Memory Map.)




**<Fig. 25: uVision-51>**

#### 3.3.1.2 Making a New Project

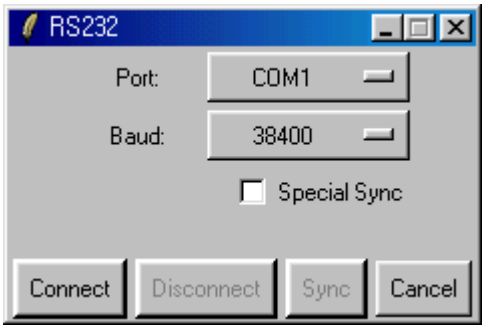
1. Make a new project and add startup.a51 and necessary sources.
   (Startup.a51 file does not need to be modified.)
2. Modify or develop the program.
3. Compile it




**<Fig. 26: Making a new project>**

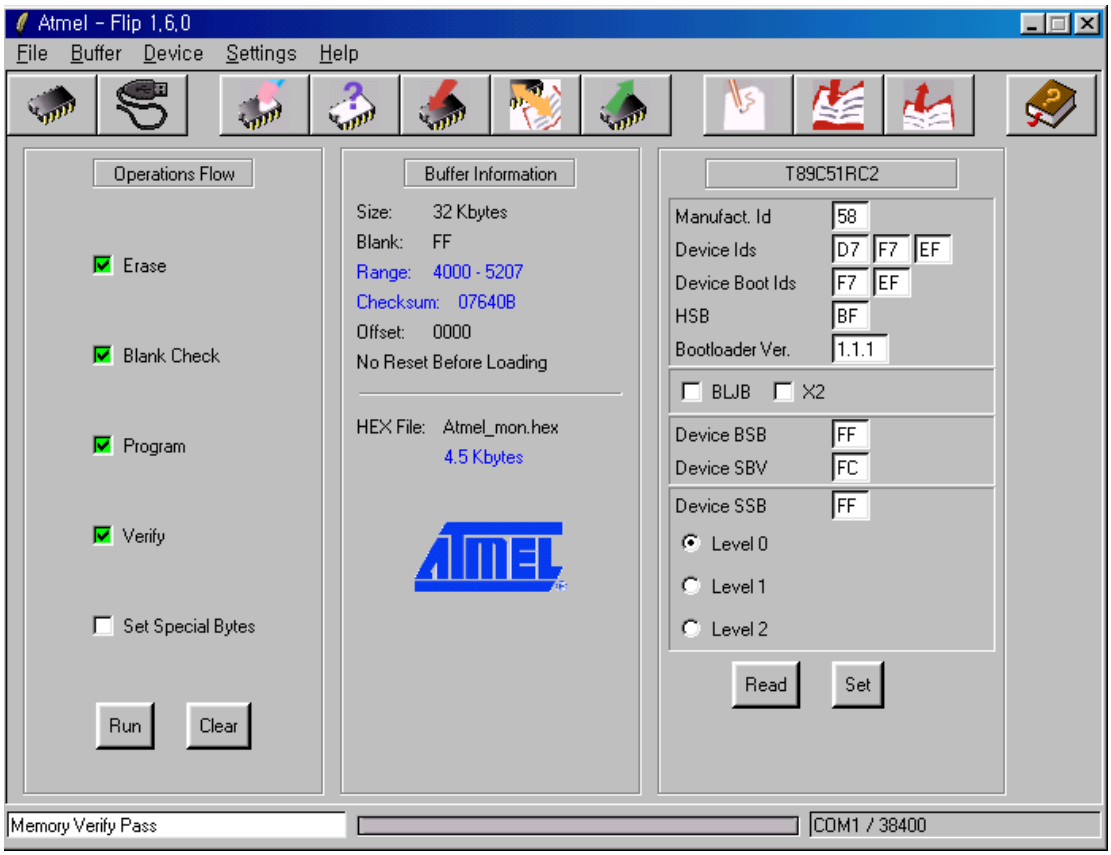
### 3.3.2 Program Downloading and Running Procedure (based on Flip by ATMEL)

1. Connect the 8051EVB and the COM port of the PC with the serial cable.
2. Slide the switch on the JP2 to the left and turn on the power.
3. Run Flip, the ISP program of ATMEL, and select T89C51RD2 as the device.
   (Device>Select… => T89C51RD2 )
4. Select Setting>Communication>RS232 and click the Connect button.




**<Fig. 27: Setting RS232>**

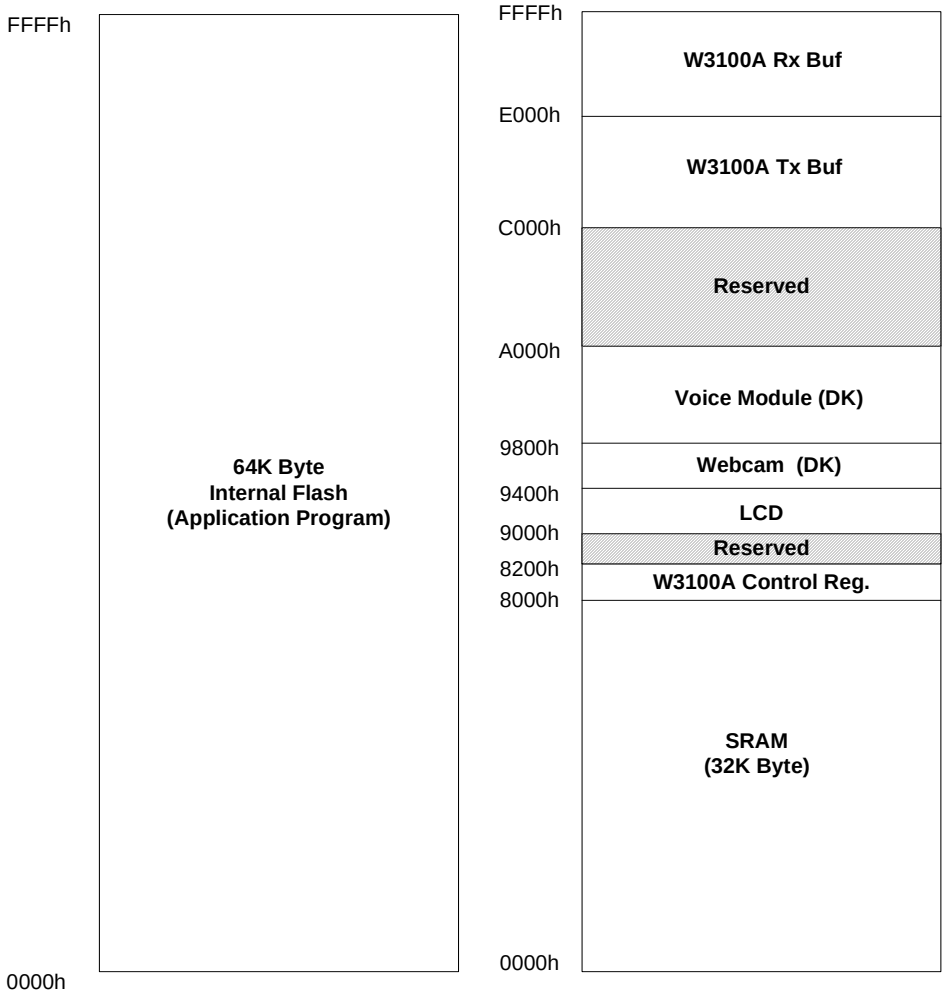
5. Execute File > Load HEX… to load the file to be downloaded.
6. Press the Run button to reprogram the internal flash memory of the 8051 in accordance with the Operation Flow.
7. Turn off the power, slide the switch on JP2 to the right, and turn on the power back to run the program that was downloaded in step 6.




**<Fig. 28: FLIP by ATMEL>**

### 3.3.3 Memory Map

**ATMEL Version (EVB)**




**<Fig. 29: Memory of EVB8051>**

# 4. Hardware Designer's Guide

## 4.1 EVB8051 Schematic

Please refer to schematics in Software CD (\Schematic\).

## 4.2 PAL

```
library ieee;
use ieee.std_logic_1164.all;

entity test is
        port(
          Addr          : in std_logic_vector(5 downto 0);
          nPSEN         : in std_logic;
          nRD           : in std_logic;
          nWR           : in std_logic;
          nEA           : in std_logic;
          nROMCS        : out std_logic;
          nRAMCS        : out std_logic;
          nCS_I2CHIP : out std_logic;
          LCDCS                 : out std_logic;
          nCS_VM        : out std_logic;
          nCS_CAM       : out std_logic;
          nROMRD        : out std_logic;
          nRAMRD        : out std_logic
        );

ATTRIBUTE pin_numbers of test:ENTITY IS
"Addr(5):6 "
& "Addr(4):5 "
& "Addr(3):4 "
& "Addr(2):3 "
& "Addr(1):2 "
& "Addr(0):1 "
& "nPSEN:9 "
& "nRD:7 "
& "nWR:8 "
& "nEA:11 "
& "nROMCS:12 "
& "nRAMCS:13 "
& "nCS_I2CHIP:15 "
& "LCDCS:14 "
& "nCS_VM:16 "
& "nCS_CAM:19 "
& "nROMRD:17 "
& "nRAMRD:18 ";
end test;

architecture arch_test of test is
begin
   nROMRD <= nPSEN;
```

```
nRAMRD <= nRD;

-- nROMCS    (0x0000 - 0x7fff) : External ROM
process(Addr, nPSEN)
begin
   if (((Addr >= "000000") and (Addr < "100000")) and (nPSEN = '0')) then
      nROMCS <= '0';
         else
      nROMCS <= '1';
         end if;
end process;

--nRAMCS (0x0000 - 0x7fff) :
process(Addr, nPSEN)
begin
   if (((Addr >= "000000") and (Addr < "100000")) and (nPSEN = '1')) then
      nRAMCS <= '0';
         else
      nRAMCS <= '1';
         end if;
end process;


--LCDCS (0x9000 - 0x93ff)
process(Addr, nRD, nWR)
begin
   if (((Addr >= "100100") and (Addr < "100101")) and (nRD = '0' or nWR = '0')) then
      LCDCS <= '1';
         else
      LCDCS <= '0';
         end if;
end process;

-- CAM (0x9400 - 0x9800)
process(Addr)
begin
   if ((Addr >= "100101") and (Addr < "100110")) then
            nCS_CAM <= '0';
         else
            nCS_CAM <= '1';
         end if;
end process;

  -- VM (0x9800 - 0xA000)
process(Addr)
begin
   if ((Addr >= "100110") and (Addr < "101000")) then
            nCS_VM <= '0';
         else
            nCS_VM <= '1';
         end if;
end process;
```

```
-- W3100A (0x8000 - 0x9000, 0xC000 - 0x10000)
process(Addr, nRD, nWR)
   begin
      if (((Addr >= "100000") and (Addr < "100100")) or (Addr >= "110000")) and (nRD = '0' or nWR =
'0')then
               nCS_I2CHIP <= '0';
            else
               nCS_I2CHIP <= '1';
            end if;
   end process;
end arch_test;
```

## 4.3 Parts List

Please refer to part list in Software CD (\Schematic\).

# Appendix A. Quick Testing Procedure

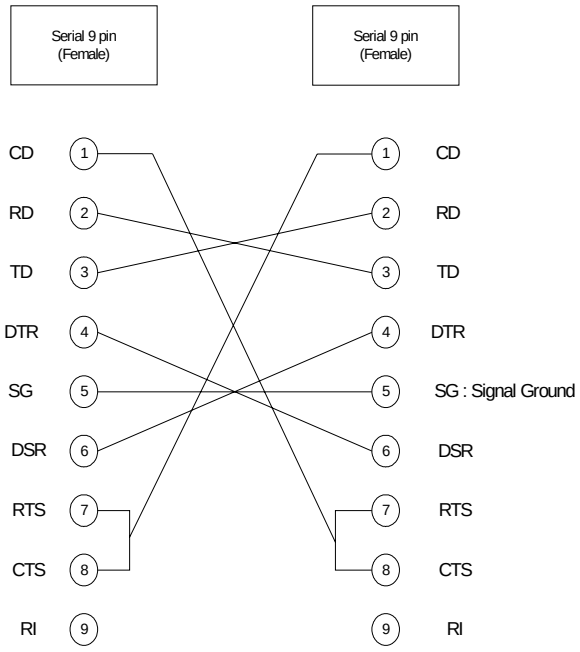
## A.1 Loopback Test

Step 1. Check whether EVB8051 is connected to PC correctly with UTP cable.
Step 2. Slide JP2 on EVB8051 to the right and turn on the power of EVB8051.
Step 3. Ping to EVB8051 (192.168.0.2) on the PC whether EVB8051 is connected to the PC correctly.
Step 4. Install Axinstall.exe (only for the first time) and run Ax1.exe on the PC.
Step 5. Select TCP\Connect menu and enter EVB8051's IP address and Port number (5000).
Step 6. Select File\Send menu and select a file to transfer.

## A.2 Web Server Test

Step 1. Connect the EVB8051 and the COM port of the PC with the serial cable.
Step 2. Slide the switch on the JP2 to the left and turn on the power.
Step 3. Run Flip and select T89C51RD2 as the device in Device\Select … menu.
Step 4. Select Setting\Communication\RS232 menu and click the 'Connect' button.
Step 5. Select File\Load HEX… to load webserv.hex to be downloaded.
Step 6. Press the 'Run' button to reprogram the internal flash memory of the 8051 in accordance with the Operation Flow.
Step 7. Turn off the power of EVB8051 and slide the switch on JP2 to the right, and turn on the power back to run the program that was downloaded in step 6.
Step 8. Run web browser on the PC and enter URL, "http://192.168.0.2"

# Appendix B. Specification of Serial Cables




TD (Transmit Data) : Serial Data Output (TXD)
RD (Receive Data) : Serial Data Input (RXD)
CTS (Clear to Send) : This line indicates that the Modem is ready to exchange data.
DCD(Data Carrier Detect) : When the modem detects a "Carrier" from the modem at the other end of the phone line, this Line becomes active.
DSR (Data Set Ready) : This tells the UART that the modem is ready to establish a link.
DTR (Data Terminal Ready) : This is the opposite to DSR. This tells the Modem that the UART is ready to link.
RTS (Request To Send) : This line informs the Modem that the UART is ready to exchange data.
RI (Ring Indicator) : Goes active when modem detects a ringing signal from the PSTN.