

User Manual

BITEK

Script MCU Assembler2 Tool User Manual

Version	0.01
Date	July 24, 2009
Author	Jeffrey Chang
Classification	Confidential
Status	Preliminary

Revision History

Ver.	Date	Author	Remarks
0.01	July 24, 2009	Jeffrey Chang	First Draft.

Copyright © 2005-2009 Beyond Innovation Technology Co., Ltd

All rights are reserved. Reproduction in whole or in parts is prohibited without the prior written consent of the copyright owner.

The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

The software described in this document is furnished under license agreement or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than for the purchaser's personal use, without written permission.

Life Support Applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Beyond Innovation Technology (BiTEK) customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Beyond Innovation Technology for any damages resulting from such improper use or sale.

Disclaimer

Beyond Innovation Technology reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Beyond Innovation Technology assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these product are free from patent, copyright, or mask work right infringement, unless otherwise specified.

CONTENTS

1. INTRODUCTION	1
1.1 Brief description.....	1
1.2 Title and status bar	1
1.3 Function of Buttons.....	4
1.3.1 Assemble.....	4
1.3.2 Download.....	4
1.3.3 EEPROM	4
1.4 Error Message	5
1.5 EEPROM interface.....	5
1.5.1 Erase button	6
2. ASSEMBLY PROGRAMMING RULES	8
2.1 Naming rules.....	8
2.1.1 Symbols.....	8
2.1.2 Labels	8
2.1.3 Comments	8
2.1.4 Decimal numbers	8
2.1.5 Hexadecimal numbers.....	8
2.2 Address control	9
2.3 Symbol definition	9
2.4 Memory Initialization.....	9
2.4.1 Define Byte	9
2.4.2 Define String.....	10
2.4.3 Define Word.....	10
2.5 Instruction Set	10

LIST OF FIGURES

FIGURE 1 ASSEMBLER2 1
FIGURE 2 THE TITLE DISPLAYS THE CURRENT EDITING FILE PATH 2
FIGURE 3 ACTIVE STATUS AND LINE NUMBER 2
FIGURE 4 RESULT OF ASSEMBLING OR VERIFYING 3
FIGURE 5 CODE SIZE AND UTILITY RATIO 3
FIGURE 6 CHECK SUM 3
FIGURE 7 PORT TYPE 3
FIGURE 8 ASSEMBLE BUTTON..... 4
FIGURE 9 DOWNLOAD BUTTON 4
FIGURE 10 EEPROM BUTTON 4
FIGURE 11 ERROR MESSAGE WINDOW 5
FIGURE 12 EEPROM INTERFACE LOOKS..... 6
FIGURE 13 ERASE BUTTON 7

1. Introduction

1.1 Brief description

This manual describes how to program BiT161x device with assembler2 utility. The assembler2 utility contains five files: assembler (BITEK_ASM2_v008.exe) file, WinIo.dll, WinIo.vxd, WinIo.sys and SiUSBXp.dll. These five files must be in the same folder.

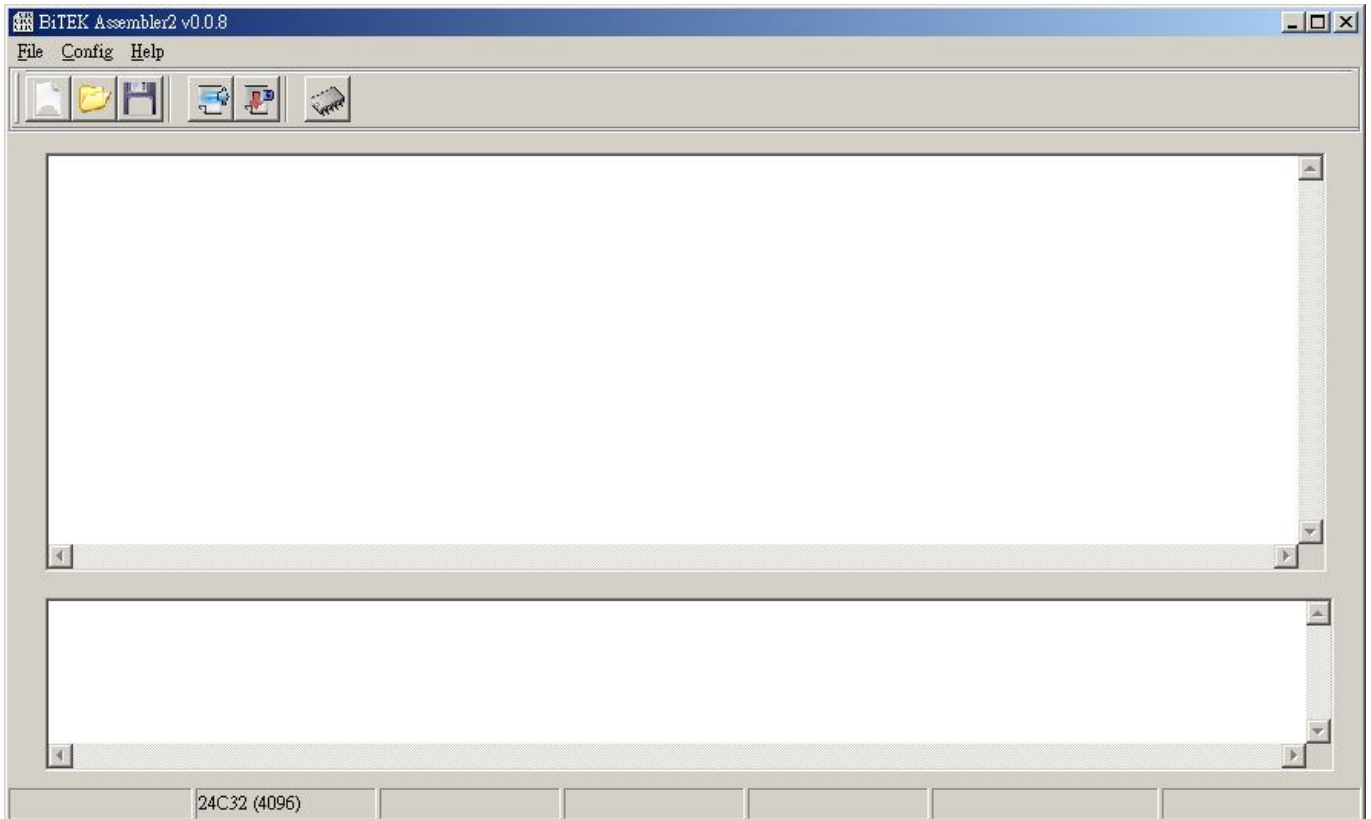


Figure 1 Assembler2

1.2 Title and status bar

When an assembly file is loaded into the assembler utility, the title will display the complete file name.

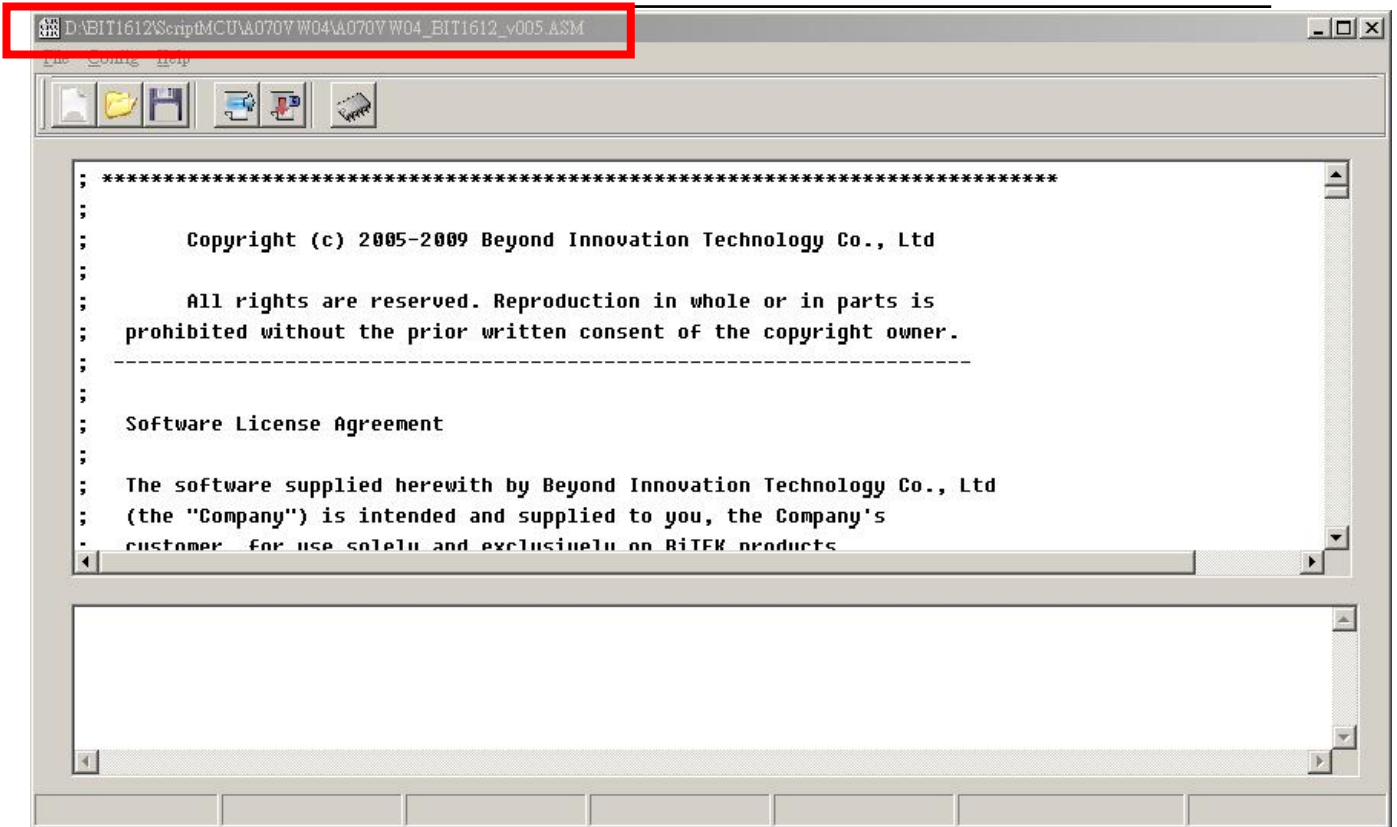


Figure 2 The title displays the current editing file path

The status bar shows the line number of the cursor position, active status, and the result after data verifying.

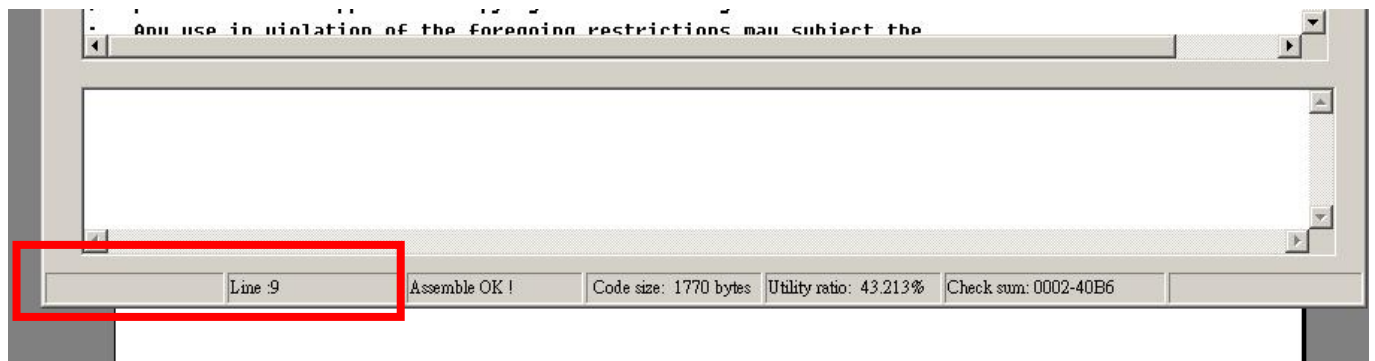


Figure 3 Active status and line number

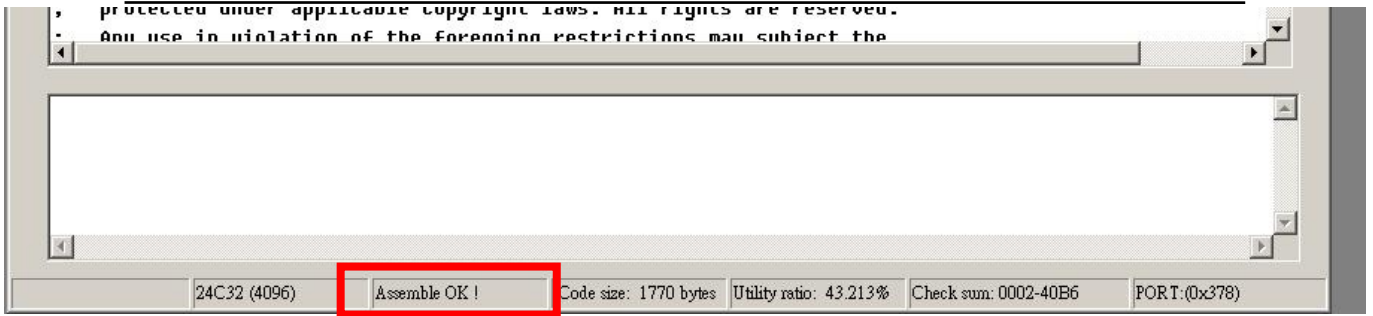


Figure 4 Result of assembling or verifying

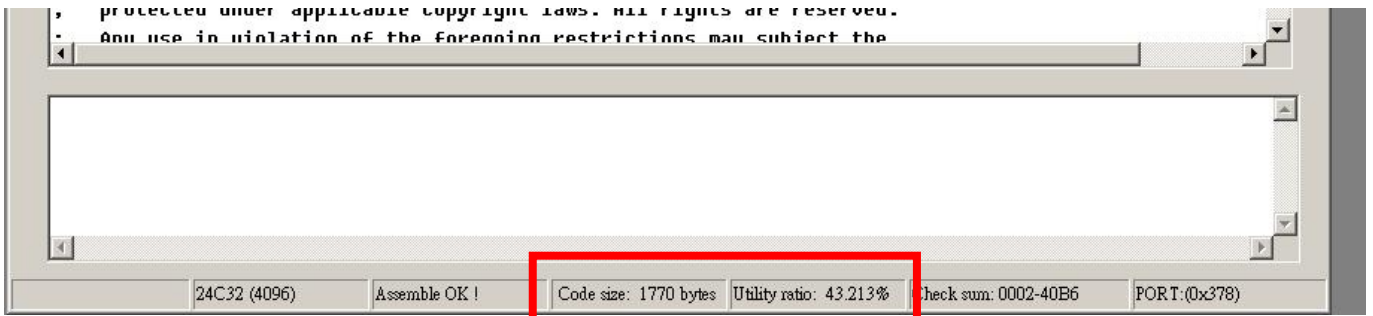


Figure 5 Code size and utility ratio

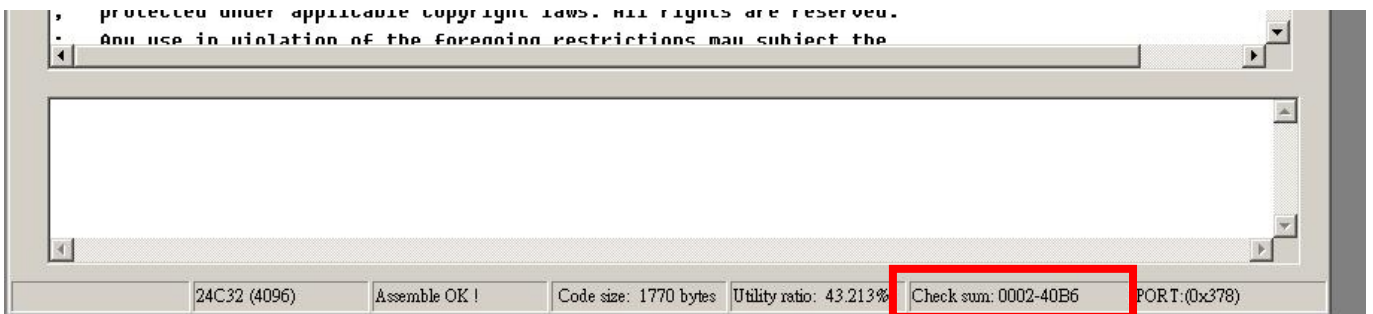


Figure 6 Check sum

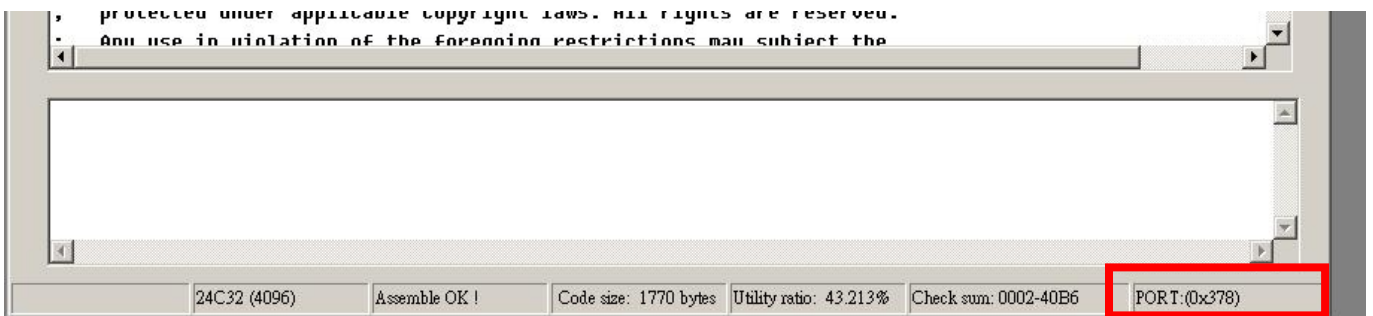


Figure 7 Port type

1.3 Function of Buttons

1.3.1 Assemble

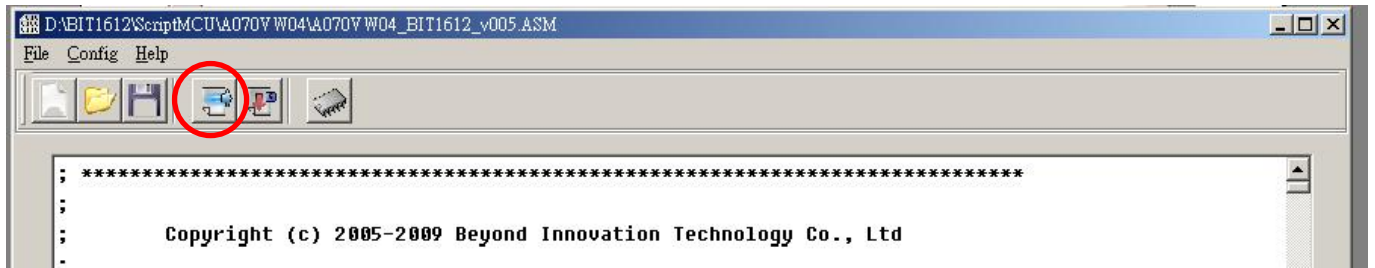


Figure 8 Assemble button

1.3.2 Download



Figure 9 Download button

1.3.3 EEPROM



Figure 10 EEPROM button

1.4 Error Message

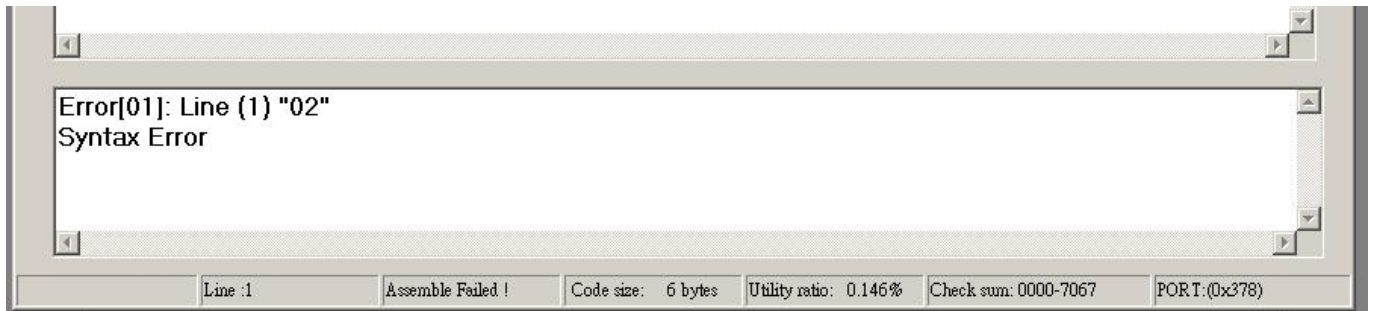


Figure 11 Error message window

1.5 EEPROM interface

The EEPROM is a simple I²C interface EEPROM read/write utility. When a cell is selected and modified, it is necessary to press <Enter> key to update the content.

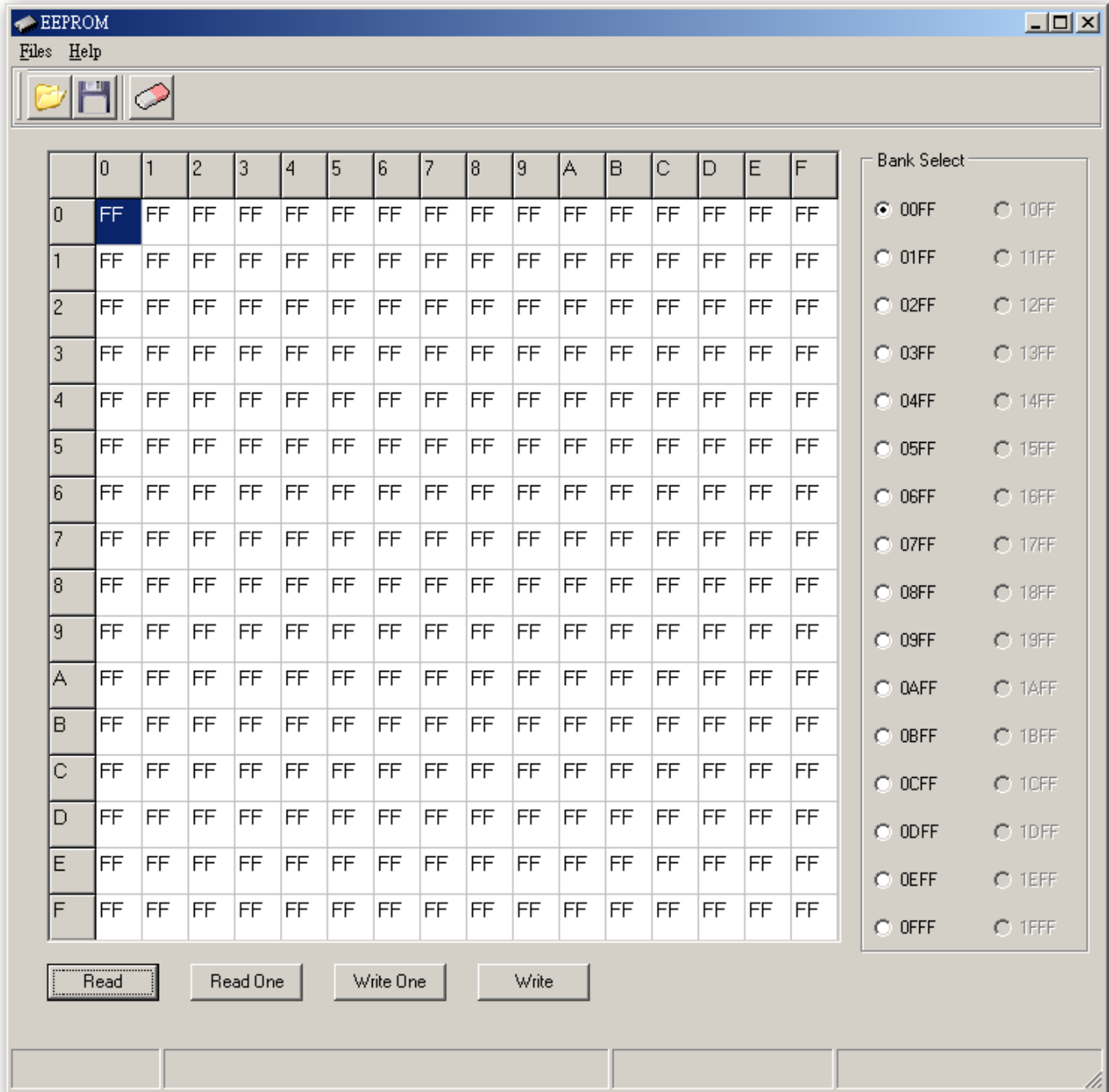


Figure 12 EEPROM interface looks

1.5.1 Erase button

The ERASE button will set all the EEPROM content to default value (0x07).

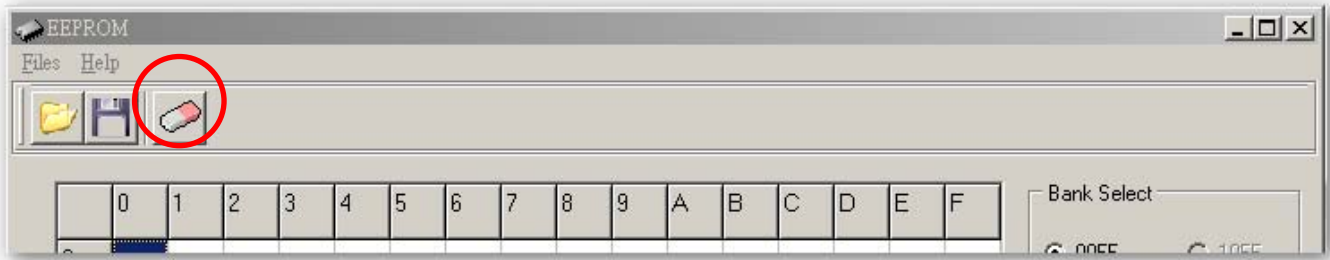


Figure 13 Erase button

2. Assembly programming rules

2.1 Naming rules

2.1.1 Symbols

Symbols includes EQU Names and Labels are composed of the following list :

A – Z, a – z, 0 – 9, and _.

A symbol name can start with these characters except the digits 0-9.

2.1.2 Labels

A label is a symbol but ended in a colon character (:). When a label is defined, it must be the first text field in a line. Only space, tab and comments are allowed to exist after a label.

Example :

```
LABEL_1:      MOV B, A      wrong !
```

```
-----
LABEL_2:
                MOV B, A      correct !
-----
```

2.1.3 Comments

All characters that come after ‘;’ are comments.

Example :

```
      ; This is a comment sample.
LABEL_1:                ;This is another
```

2.1.4 Decimal numbers

Decimal numbers can be written as following format :

3, 127, 3d and 3D.

2.1.5 Hexadecimal numbers

Hexadecimal numbers must start with ‘0’ and end in ‘H’ or ‘h’ :

0FFh, 0EAH and 0235H.

2.2 Address control

Name : ORG

Format :

ORG argument

Argument : decimal numbers \ hexadecimal numbers \ symbols

Example :

```
START EQU 0000H
ORG START
ORG 200
ORG 0300H
```

2.3 Symbol definition

Name : EQU

Format :

symbol-name **EQU** argument

Argument : decimal and hexadecimal numbers

Example :

```
STARTU EQU 000H
```

Name : EQUH

Format :

symbol-name **EQUH** argument

Argument : decimal \ hexadecimal numbers and symbols

Example :

```
TEST EQU 01234H
TEST1 EQUH TEST (TEST1 = 012H)
```

Name : EQUH

Format :

symbol-name **EQUH** argument

Argument : decimal \ hexadecimal numbers and symbols

Example :

```
TEST EQU 01234H
TEST1 EQUH TEST (TEST1 = 034H)
```

2.4 Memory Initialization

2.4.1 Define Byte

Name : DB

Format :

DB number, number, . . . , number
DB symbol, symbol, . . . , symbol
DB number, symbol, . . . , symbol

.

DB number

Note: numbers are 8-bit byte values

2.4.2 Define String

Name: DS

Format:

DS "expression"

Note: expression is a string between " and "

2.4.3 Define Word

Name: DW

Format:

DW Symbol-name

Note: DW reserve a 2-byte long space in memory

2.5 Instruction Set

Identifier	Description
A	Accumulator (8-bit)
B	B register (8-bit)
C	Carry flag
Z	Zero flag
PC	Program counter (11-bit)
Eaddr	EEPROM address (11-bit)
Raddr	Internal Register Set address (11-bit)
Iaddr	I ² C sub-address (8-bit)
REG_ADDR	ADDR register (11-bit)
REG_NUM	NUM register (8-bit)
REG_CNT	CNT register (8-bit)

ADD	0	0	0	0	0	0	0	0	0	1
A ← A + B										C, Z

AND	0	0	0	0	1	0	0	0	0	1
A ← A and B										

CALL Eaddr	1	1	1	1	1	E10	E9	E8	2	
	E7	E6	E5	E4	E3	E2	E1	E0		
RET_ADDR ← PC + 1, PC ← Eaddr										

CLR	0	0	0	0	0	1	0	0	1
A ← 0									Z
COMP	0	0	0	0	0	1	0	1	1
A - B(not update A)									C, Z
DEC	0	0	0	0	0	0	1	1	1
A ← A - 1									C, Z
DELAY NUM	0	0	1	0	0	0	1	0	2
	N7	N6	N5	N4	N3	N2	N1	N0	2
DELAY dec hex symbol XCLKs									
FILL R[Raddr], Num, Cnt	0	0	1	1	1	R10	R9	R8	4
	R7	R6	R5	R4	R3	R2	R1	R0	
	N7	N6	N5	N4	N3	N2	N1	N0	
	C7	C6	C5	C4	C3	C2	C1	C0	
For Idx = 0 to Cnt do R[Raddr + Idx] ← Num									
FILLR	1	1	0	1	1	1	0	0	1
For Idx = 0 to REG_CNT do R[REG_ADDR + Idx] ← REG_NUM									
HALT	0	0	0	0	0	1	1	1	1
Halt program and stay at standby mode									
INC	0	0	0	0	0	0	1	0	1
A ← A + 1									C, Z
JB Eaddr, Bit	1	0	B2	B1	B0	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
if (A[Bit] == 1) PC ← Eaddr else PC ← PC + 1									

JC Eaddr	1	1	1	0	0	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
<pre>if (C == 1) PC ← Eaddr else PC ← PC + 1</pre>									
JMP Eaddr	1	1	1	1	X	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
PC ← Eaddr									
JMPR REG_ADDR	1	1	0	1	1	1	1	1	1
PC ← REG_ADDR									
JZ EADDR	1	1	1	0	1	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
<pre>if (Z == 1) PC ← Eaddr else PC ← PC + 1</pre>									
MOV A, Num	0	0	1	0	0	0	0	0	2
	N7	N6	N5	N4	N3	N2	N1	N0	
A ← Num									Z
MOV A, E[Eaddr]	1	1	0	0	1	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
A ← E[Eaddr]									Z
MOV A, I[Iaddr]	1	1	0	1	1	0	0	0	2
	I7	I6	I5	I4	I3	I2	I1	I0	
A ← I[Iaddr]									Z
MOV A, R[Raddr]	0	0	0	1	0	R10	R9	R8	2
	R7	R6	R5	R4	R3	R2	R1	R0	
A ← R[Raddr]									Z
MOV B, A	0	0	0	0	0	1	1	0	1
B ← A									

MOV B, Num	0	0	1	0	0	0	0	1	2
	N7	N6	N5	N4	N3	N2	N1	N0	
B ← Num									

MOV E[Eaddr], A	1	1	0	0	0	E10	E9	E8	2
	E7	E6	E5	E4	E3	E2	E1	E0	
E[Eaddr] ← A									

MOV E[Eaddr], Num	0	0	1	0	1	E10	E9	E8	3
	E7	E6	E5	E4	E3	E2	E1	E0	
	N7	N6	N5	N4	N3	N2	N1	N0	
E[Eaddr] ← Num									

MOV I[Iaddr], A	1	1	0	1	0	0	0	0	2
	I7	I6	I5	I4	I3	I2	I1	I0	
I[Iaddr] ← A									

MOV R[Raddr], A	0	0	0	1	1	R10	R9	R8	2
	R7	R6	R5	R4	R3	R2	R1	R0	
R[Raddr] ← A									

MOV R[Raddr], Num	0	0	1	1	0	R10	R9	R8	3
	R7	R6	R5	R4	R3	R2	R1	R0	
	N7	N6	N5	N4	N3	N2	N1	N0	
R[Raddr] ← Num									

MOV R[Raddr], E[Eaddr], Cnt	0	1	E10	E9	E8	R10	R9	R8	4
	E7	E6	E5	E4	E3	E2	E1	E0	
	R7	R6	R5	R4	R3	R2	R1	R0	
	N7	N6	N5	N4	N3	N2	N1	N0	
For Idx = 0 to Cnt do R[Raddr + Idx] ← E[Eaddr + Idx]									

MOV S, Num	1	1	0	1	0	1	0	0	4
	N23	N22	N21	N20	N19	N18	N17	N16	
	N15	N14	N13	N12	N11	N10	N9	N8	
	N7	N6	N5	N4	N3	N2	N1	N0	
SPI ← Num (up to $2^{24}-1$)									
NOT	0	0	0	0	1	0	1	1	1
A ← ~A									Z
NOTC	0	0	1	0	0	1	0	0	1
C ← ~C									C
NOTZ	0	0	1	0	0	1	0	1	1
Z ← ~Z									Z
OR	0	0	0	0	1	0	0	1	1
A ← A or B									Z
RET	0	0	1	0	0	1	1	0	1
PC ← RET_ADDR									
SHL	0	0	0	0	1	1	0	1	1
A ← A << 1									Z
SHR	0	0	0	0	1	1	0	0	1
A ← A >> 1									Z
SUB	0	0	0	0	0	0	0	1	1
A ← A - B									C, Z
SWAP	0	0	0	0	1	1	1	0	1
A ↔ B									Z
TABLE	1	1	0	1	1	1	0	1	1
A ← E[REG_ADDR]									Z
XOR	0	0	0	0	1	0	1	0	1
A ← A xor B									Z

※symbol|dec|hex means the argument accepts symbols or decimal numbers or hexadecimal numbers.