GENTOS

BM-GENTOS-V2.0

BM

Brief Manual of GENTOS

GENTOS: CoreRiver's Total Solutions for Embedded System Development

V2.0

August 2005

- CoreRiver Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice.
- Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
- The CoreRiver Semiconductor products listed in this document are intended for usage in general electronics applications. These CoreRiver Semiconductor products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.

Contents

[PART I : Introduction]

[PART II : GENTOS S/W]

[PART III : GENSYS52 MDS for MiDAS 1.0]

[PART IV : GENSYS52 MDS for MiDAS 1.1]

[PART V : GenICE52 System for MiDAS2.0/2.1]

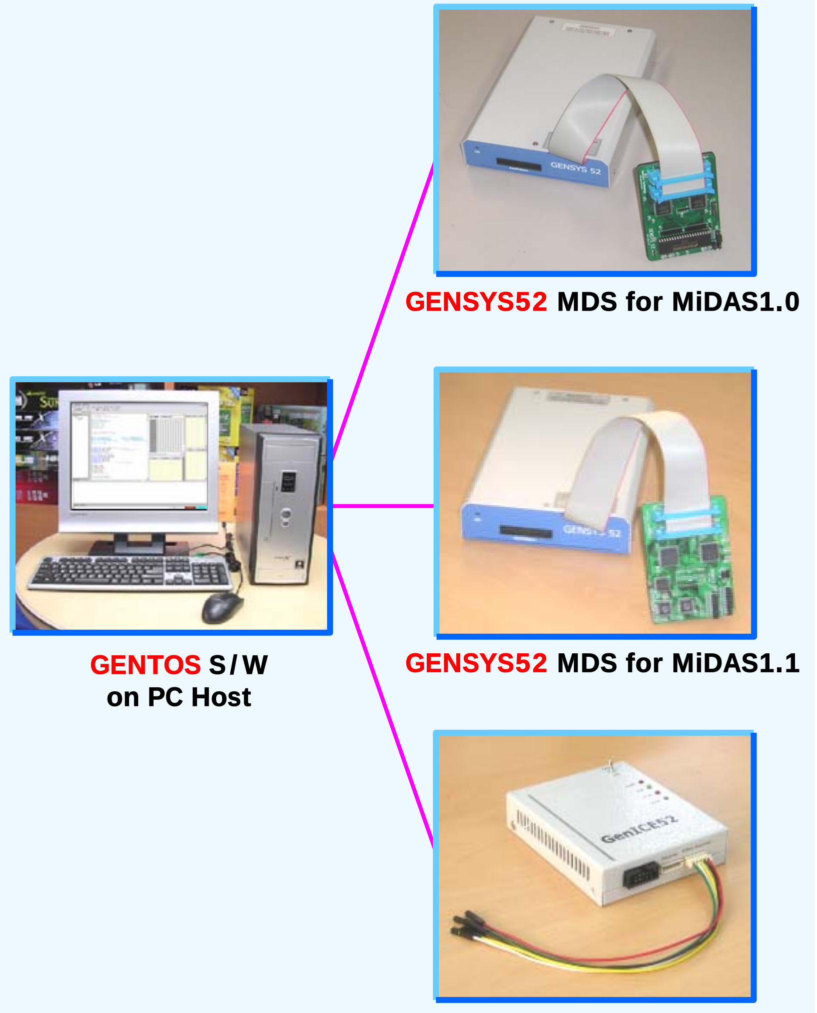
[PART VI : SUPPORTS]

[APPENDIX]

PART I : Introduction

1. Overview : GENTOS S/W, GENSYS52, and GenICE52
2. How to Develop Your Target System

1. Overview : GENTOS S/W, GENSYS52, and GenICE52



GENSYS52 MDS for MiDAS1.0

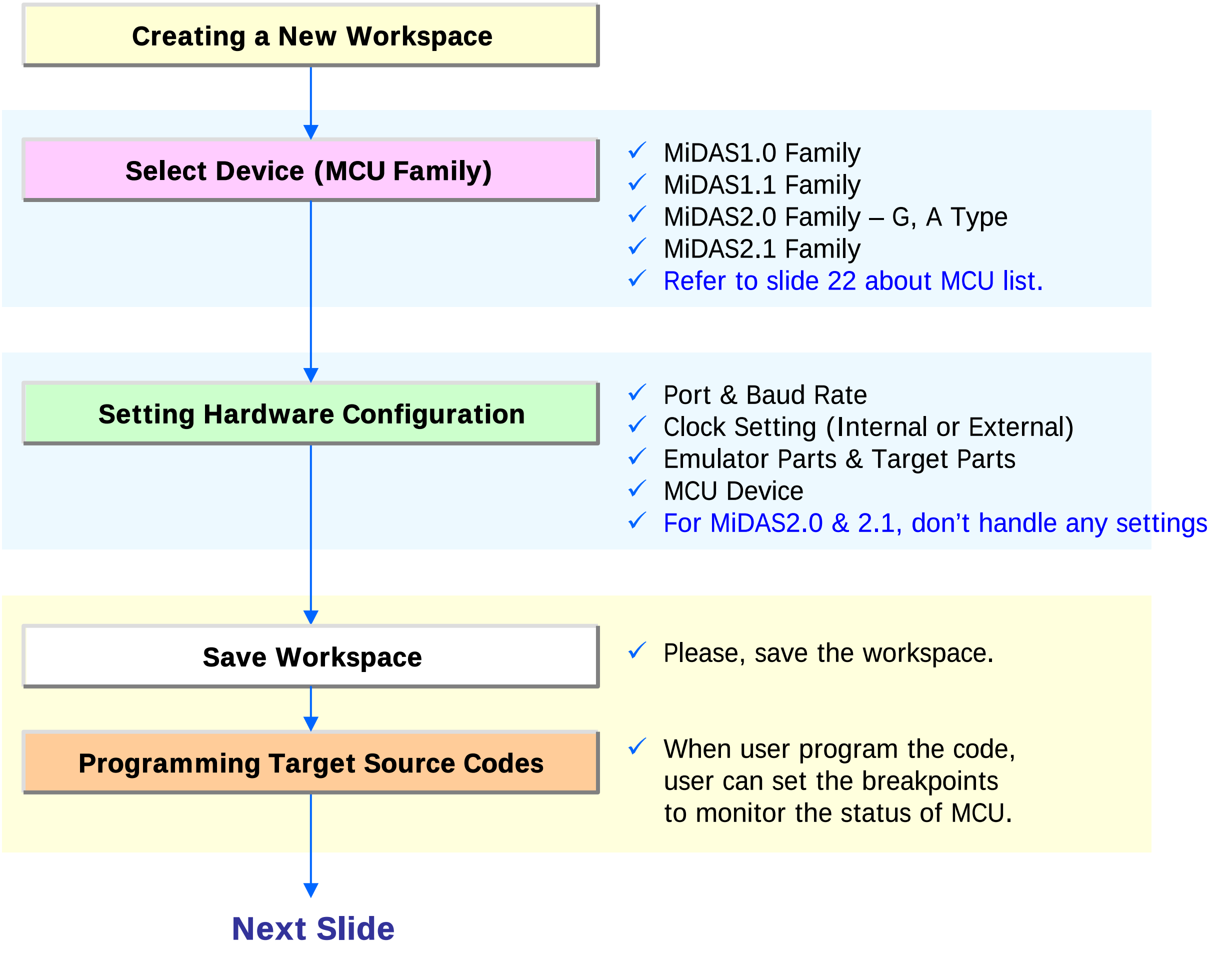
GENTOS S/W
on PC Host

GENSYS52 MDS for MiDAS1.1

GenICE52 MDS for MiDAS2.0 & 2.1

- GENTOS is an Coreriver's development environment, which contains all the necessary tools in one shell.
  - ✓ During complete development cycle, you can stay within GENTOS, write your code, run the compiler, assembler and linker and debug your software and hardware within one tool.
- Visual User Interface
  - ✓ Easy to Use GUI (GENTOS)
  - ✓ Integrated Develop Environment
  - ✓ Project Manager
  - ✓ Multi-File Source Editor
  - ✓ High-Level Source Debugger
  - ✓ Unified User Interface For All Tools
  - ✓ Extensive On-Line Help
  - ✓ Windows 98 / 2000 / XP Compatible
- The GENTOS increases your productivity and speeds up the development cycle.

2. How to Develop Your Target System



Creating a New Workspace

Select Device (MCU Family)

- ✓ MiDAS1.0 Family
- ✓ MiDAS1.1 Family
- ✓ MiDAS2.0 Family – G, A Type
- ✓ MiDAS2.1 Family
- ✓ Refer to slide 22 about MCU list.

Setting Hardware Configuration

- ✓ Port & Baud Rate
- ✓ Clock Setting (Internal or External)
- ✓ Emulator Parts & Target Parts
- ✓ MCU Device
- ✓ For MiDAS2.0 & 2.1, don't handle any settings

Save Workspace

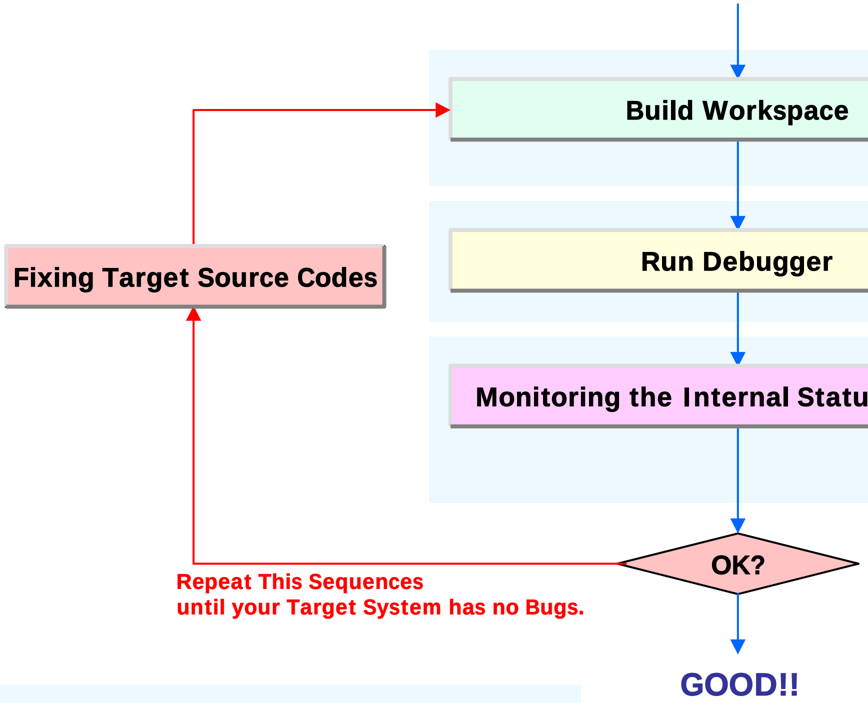
- ✓ Please, save the workspace.

Programming Target Source Codes

- ✓ When user program the code, user can set the breakpoints to monitor the status of MCU.

Next Slide

2. How to Develop Your Target System


Build Workspace
Run Debugger
Monitoring the Internal Status for MCU
Fixing Target Source Codes
OK?
Repeat This Sequences
until your Target System has no Bugs.
GOOD!!

- ✓ `gencc` : C Compiler
- ✓ `genasm` : Assembler
- ✓ `genlnk` : Linker

- ✓ Downloading Target Code
- ✓ Breakpoint

- ✓ SFRs
- ✓ Internal / External / Code Memory
- ✓ Watch : Variable (Only in C Programs)
- ✓ Disassembly

◆ MiDAS1.0 / 1.1 Families

- ✓ **Compile (Ctrl+F6)**
  - ◆ compile with `gencc` or `genasm`
- ✓ **Build (Ctrl+F7)**
  - ◆ compile with `gencc` or `genasm`
  - ◆ link with `genlnk`
- ✓ **Build & Run (F7)**
  - ◆ Build & Run
- ✓ **Go (F5)**
  - ◆ Run, or Continue when stop

◆ MiDAS2.0 & 2.1 Family

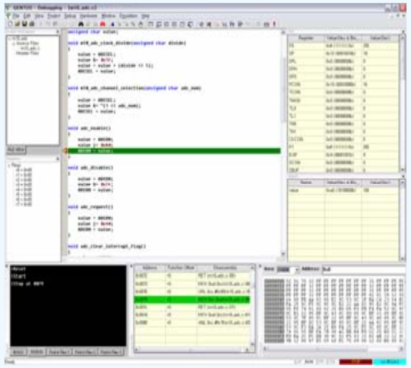
- ✓ **Compile (Ctrl+F6)**
  - ◆ compile with `gencc` or `genasm`
- ✓ **Build (Ctrl+F7)**
  - ◆ compile with `gencc` or `genasm`
  - ◆ link with `gencc or aslinmk`
- ✓ **Build & Run (F7)**
  - ◆ Build, write & Run
- ✓ **Go (F5) at Menu**
  - ◆ Run, or Continue when stop
- ✓ **Go at Tool button**
  - ◆ Write & Run, or Continue when stop
- ✓ "Write" means "Download the User Program into MCU Falsh".

PART II : GENTOS S/W

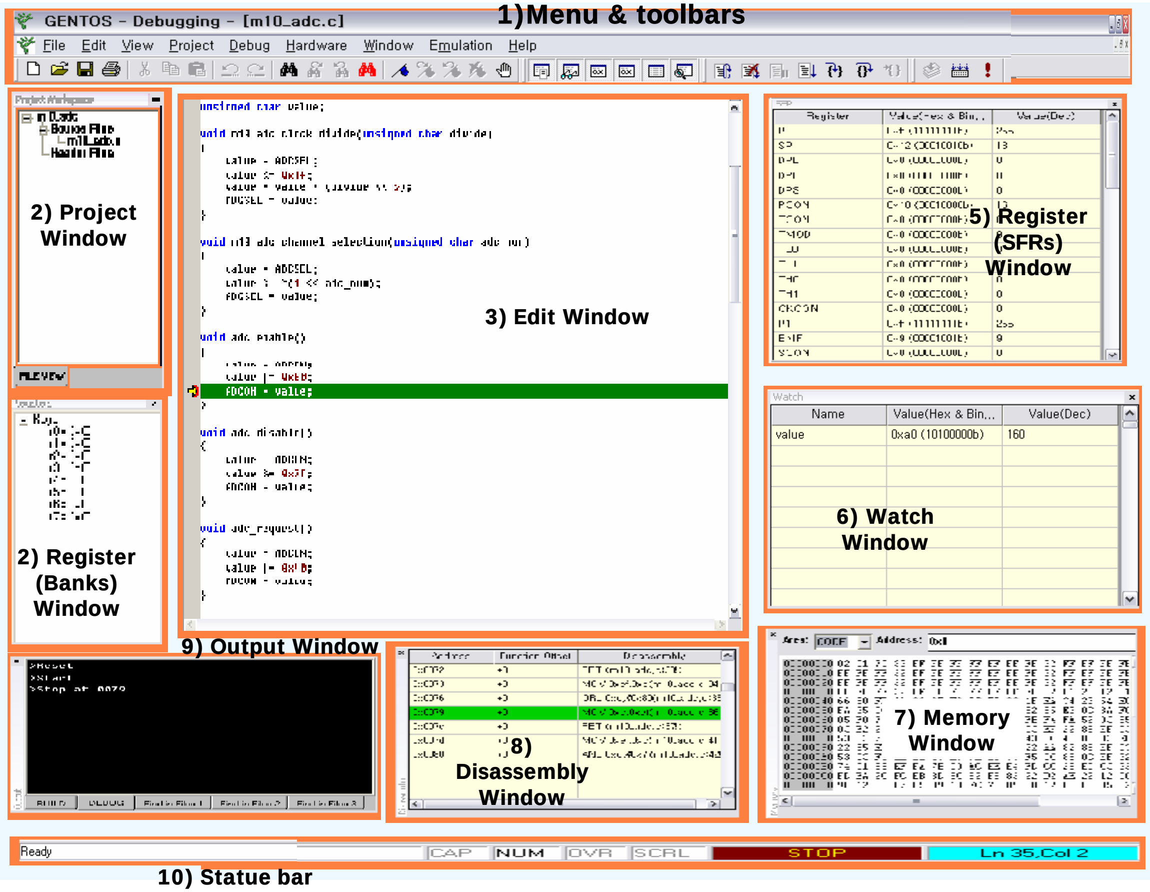
1. Overview
2. View Windows
3. Menus
4. Toolbars

1. Overview

- Multiple Window Interface (MWI)
  - ✓ Several windows can be viewed and edited at the same time.
  - ✓ You can open and process one or more of your source files.
  - ✓ There is plenty of other windows to be displayed, like the project workspace window, the output window and other special debug windows.



GENTOS Program :
Refer "8. How to Download & Install".



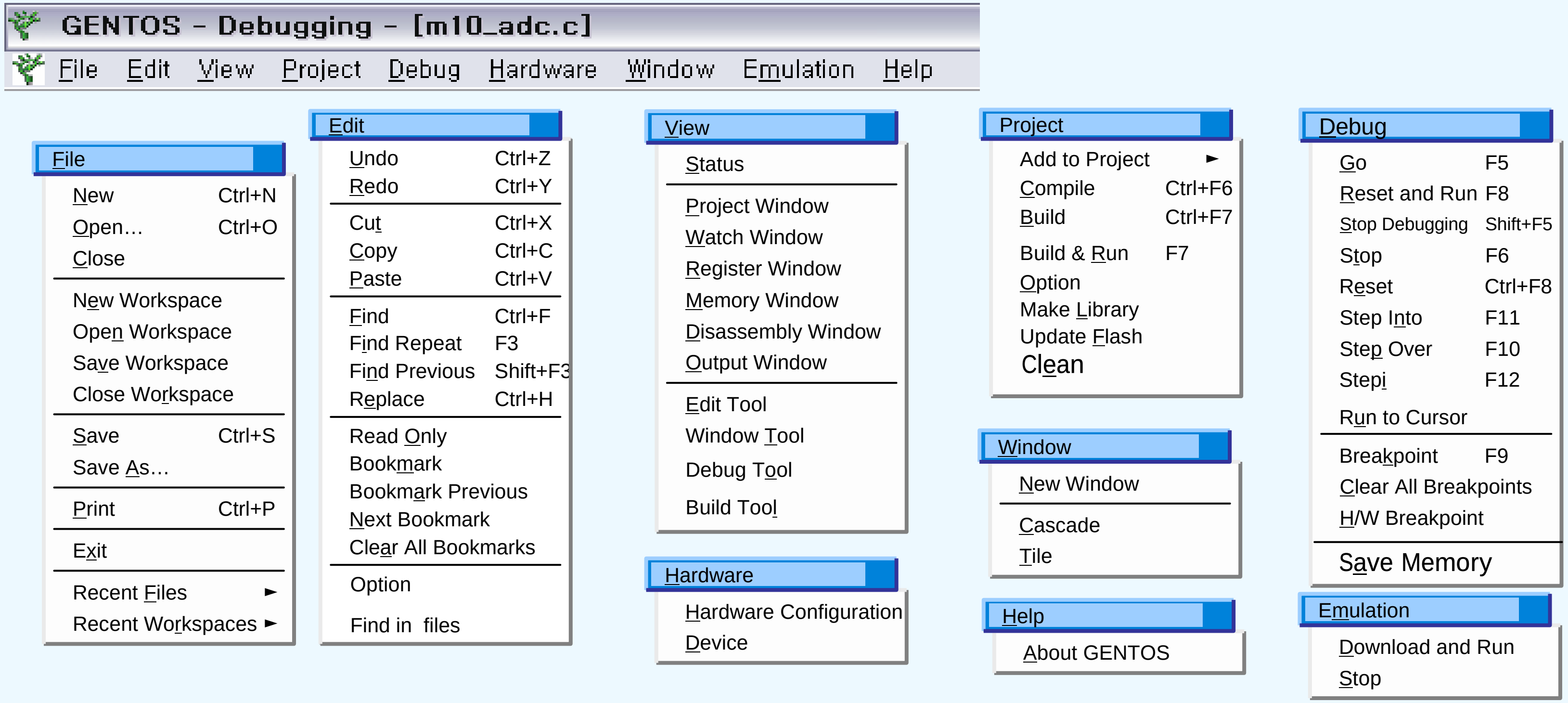
```
unsigned char value;

void m10_adc_clock_divide(unsigned char divide)
{
    value = ADCSEL;
    value &= 0xF+;
    value = value | (divide << 5);
    ADCSEL = value;
}

void m10_adc_channel_selection(unsigned char adc_num)
{
    value = ADCSEL;
    value &= ~(1 << adc_num);
    ADCSEL = value;
}

void adc_enable()
{
    value = ADCON;
    value |= 0x80;
    ADCON = value;
}

void adc_disable()
{
    value = ADCON;
    value &= 0x7F;
    ADCON = value;
}

void adc_request()
{
    value = ADCON;
    value |= 0x10;
    ADCON = value;
}
```

```
>Reset
>Start
>Stop at 0079
```

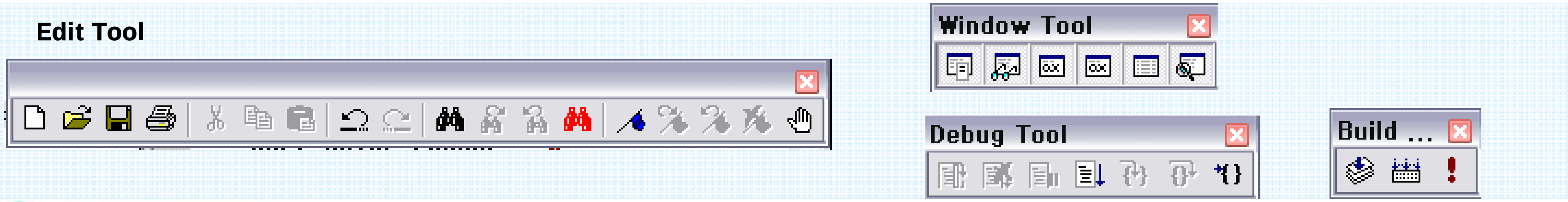
Name	Value(Hex & Bin...	Value(Dec)
value	0xa0 (10100000b)	160

2. View Windows : 1) Menu & Toolbars

- Menu : Refer to “3. Menus”

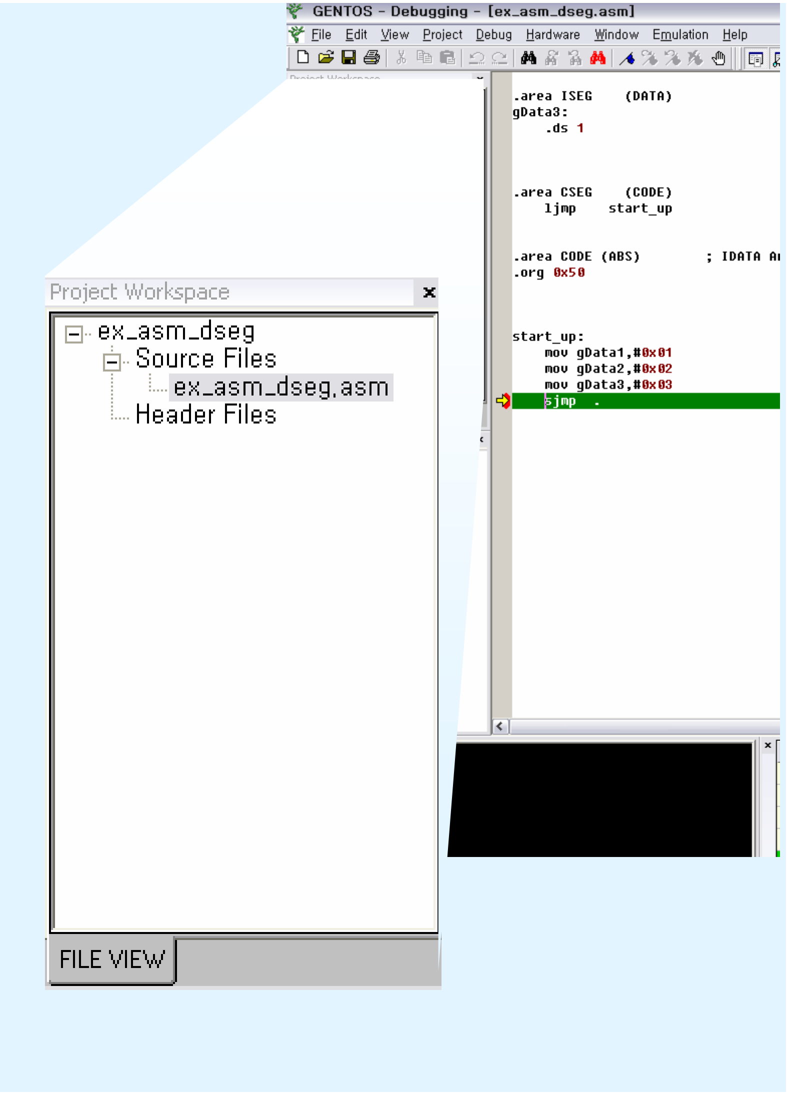


- Toolbar : Refer to “4. Toolbars”



2. View Windows : 2) Project Window

- The Project Manager
  - ✓ Organize Complete Project
  - ✓ Make it more friendly to other developers.
- Tree Structure
  - ✓ Create a group of all project files like
    - source,
    - header,
    - link,
    - library,
    - text, and
    - other user’s files
  - ✓ The user can navigate through all project files easily.



GENTOS - Debugging - [ex_asm_dseg.asm]

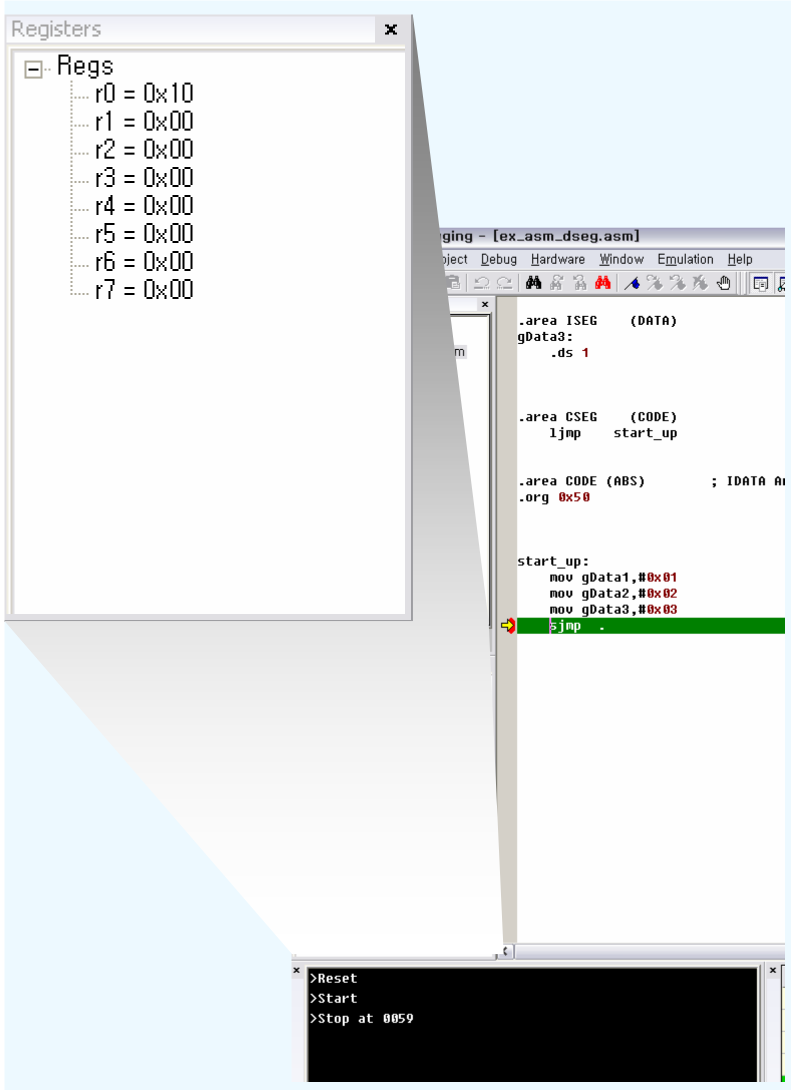
```
.area ISEG    (DATA)
gData3:
    .ds 1



.area CSEG    (CODE)
    ljmp    start_up


.area CODE (ABS)        ; IDATA A
.org 0x50



start_up:
    mov gData1,#0x01
    mov gData2,#0x02
    mov gData3,#0x03
    sjmp  .
```

2. View Windows : 3) Register (Banks) Window

- The Register (Banks) Window
  - ✓ Viewing MCU's Banks(00h~20h Registers)
  - ✓ 1 Bank make of 8bit Register(r0 ~ r7)

- Not Modifying
  - ✓ don't modify on register Window

    but modify in program code.



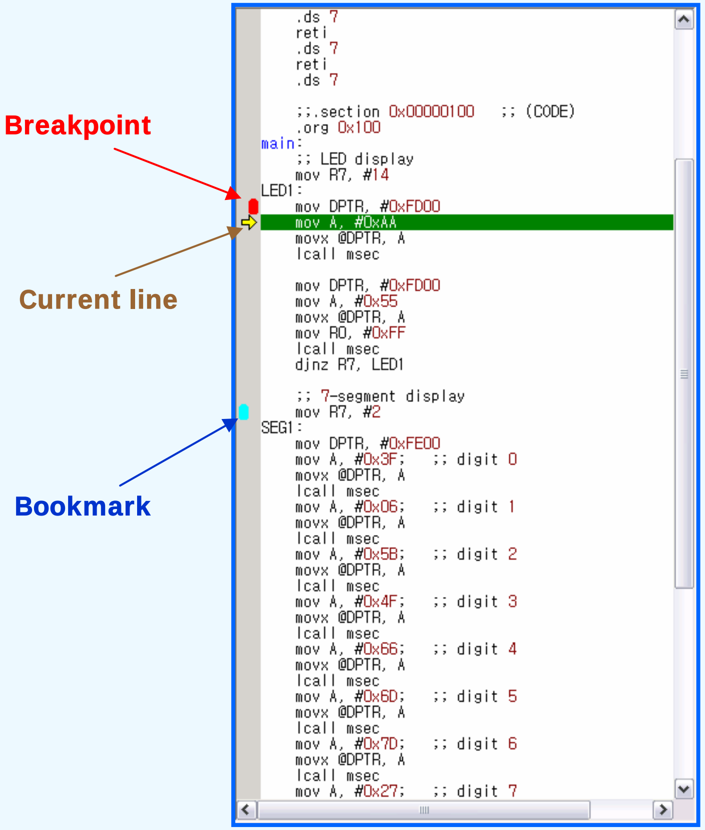
```
r0 = 0x10
r1 = 0x00
r2 = 0x00
r3 = 0x00
r4 = 0x00
r5 = 0x00
r6 = 0x00
r7 = 0x00
```

```
.area ISEG    (DATA)
gData3:
    .ds 1


.area CSEG    (CODE)
    ljmp    start_up


.area CODE (ABS)          ; IDATA Ar
.org 0x50


start_up:
    mov gData1,#0x01
    mov gData2,#0x02
    mov gData3,#0x03
    sjmp  .
```

```
>Reset
>Start
>Stop at 0059
```

2. View Windows : 4) Edit Window

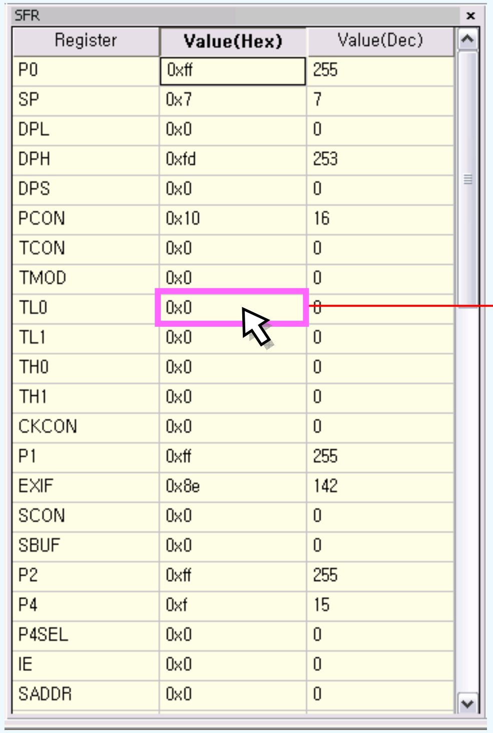
- Enables you to open and modify as many source files as you want to.
- As soon as you have finished your changes to your source, use make and build capabilities supported by GENTOS, which will recompile all changed files (incremental compiling).
- Basic Functions
  - ✓ find and replace
  - ✓ copy, cut, and paste
  - ✓ drag and drop text
- Advanced Functions
  - ✓ Syntax coloring
  - ✓ Customizing foreground / background colors and font
  - ✓ Breakpoint
  - ✓ Bookmark



```
    .ds 7
    reti
    .ds 7
    reti
    .ds 7

    ;;.section 0x00000100   ;; (CODE)
    .org 0x100
main:
    ;; LED display
    mov R7, #14
LED1:
    mov DPTR, #0xFD00
    mov A, #0xAA
    movx @DPTR, A
    lcall msec

    mov DPTR, #0xFD00
    mov A, #0x55
    movx @DPTR, A
    mov R0, #0xFF
    lcall msec
    djnz R7, LED1

    ;; 7-segment display
    mov R7, #2
SEG1:
    mov DPTR, #0xFE00
    mov A, #0x3F;   ;; digit 0
    movx @DPTR, A
    lcall msec
    mov A, #0x06;   ;; digit 1
    movx @DPTR, A
    lcall msec
    mov A, #0x5B;   ;; digit 2
    movx @DPTR, A
    lcall msec
    mov A, #0x4F;   ;; digit 3
    movx @DPTR, A
    lcall msec
    mov A, #0x66;   ;; digit 4
    movx @DPTR, A
    lcall msec
    mov A, #0x6D;   ;; digit 5
    movx @DPTR, A
    lcall msec
    mov A, #0x7D;   ;; digit 6
    movx @DPTR, A
    lcall msec
    mov A, #0x27;   ;; digit 7
```

2. View Windows : 5) Register (SFRs) Window

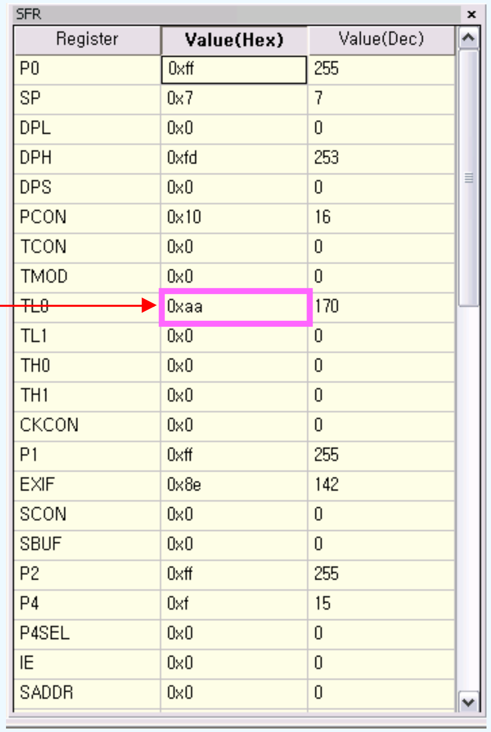
- Viewing MCU’s SFRs (Special Function Registers)
- Modifying Their Contents Manually



Register	Value(Hex)	Value(Dec)
P0	0xff	255
SP	0x7	7
DPL	0x0	0
DPH	0xfd	253
DPS	0x0	0
PCON	0x10	16
TCON	0x0	0
TMOD	0x0	0
TL0	0x0	0
TL1	0x0	0
TH0	0x0	0
TH1	0x0	0
CKCON	0x0	0
P1	0xff	255
EXIF	0x8e	142
SCON	0x0	0
SBUF	0x0	0
P2	0xff	255
P4	0xf	15
P4SEL	0x0	0
IE	0x0	0
SADDR	0x0	0

Modifying the Value of TL0 Register

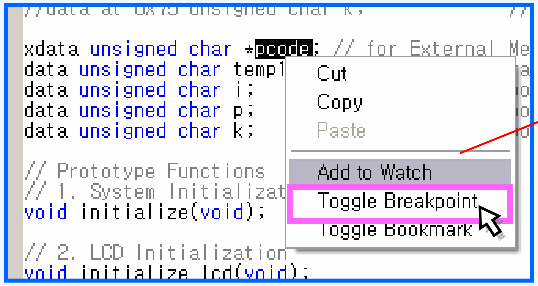
1. Double click the left mouse button
2. Modify the value for selected SFR



Register	Value(Hex)	Value(Dec)
P0	0xff	255
SP	0x7	7
DPL	0x0	0
DPH	0xfd	253
DPS	0x0	0
PCON	0x10	16
TCON	0x0	0
TMOD	0x0	0
TL0	0xaa	170
TL1	0x0	0
TH0	0x0	0
TH1	0x0	0
CKCON	0x0	0
P1	0xff	255
EXIF	0x8e	142
SCON	0x0	0
SBUF	0x0	0
P2	0xff	255
P4	0xf	15
P4SEL	0x0	0
IE	0x0	0
SADDR	0x0	0

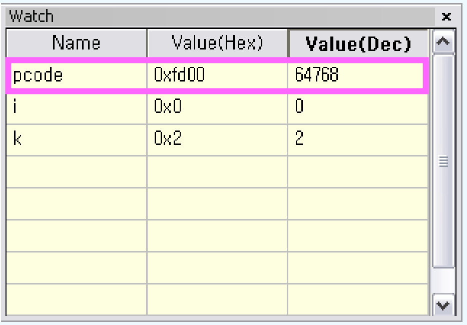
2. View Windows : 6) Watch Window

- Viewing the Global Variables
- Modifying Their Contents Manually



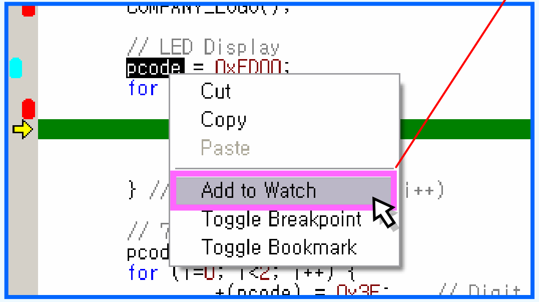
```
xdata unsigned char *pcode; // for External Me
data unsigned char templ
data unsigned char i;
data unsigned char p;
data unsigned char k;

// Prototype Functions
// 1. System Initializat
void initialize(void);

// 2. LCD Initialization
void initialize_lcd(void);
```

a. at the declaration part for global variables



Name	Value(Hex)	Value(Dec)
pcode	0xfd00	64768
i	0x0	0
k	0x2	2

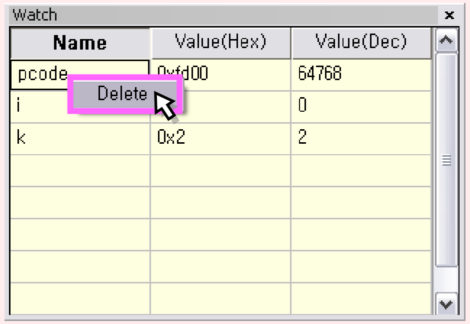
1. Select the name of global variable (Double click the left mouse button)
2. Click the right mouse button (Pop-up menu)
3. Select “Add to Watch”
4.Or drag@drop is possible.



```
COMPANY_LOGO();

// LED Display
pcode = 0xFD00;
for

} //                    i++)

// 7
pcod
for (i=0; i<2; i++) {
    *(pcode) = 0x3F;      // Digit
```

b. at the body code part

[Add the variable from Watch Window]

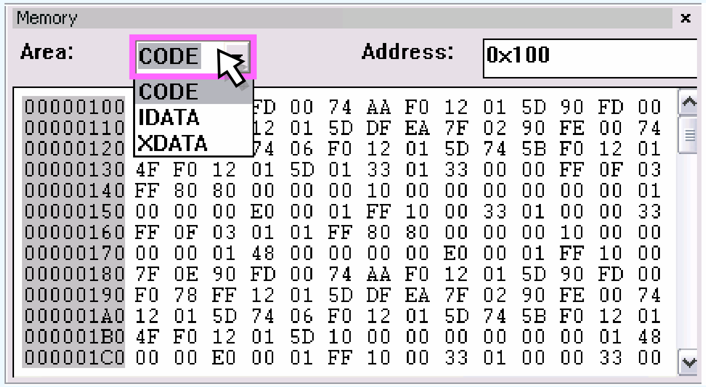


Name	Value(Hex)	Value(Dec)
pcode	0xfd00	64768
i		0
k	0x2	2

[Remove the variable from Watch Window]

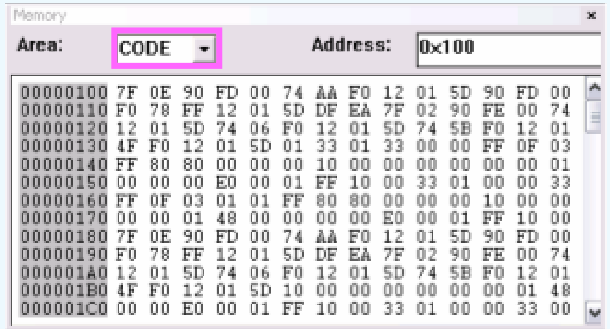
2. View Windows : 7) Memory Window

- Viewing the MCU’s Memories :
  - ✓ Code Memory (CODE)
  - ✓ Internal Memory (IDATA)
  - ✓ External Memory (XDATA)
- Modifying Their Contents Manually



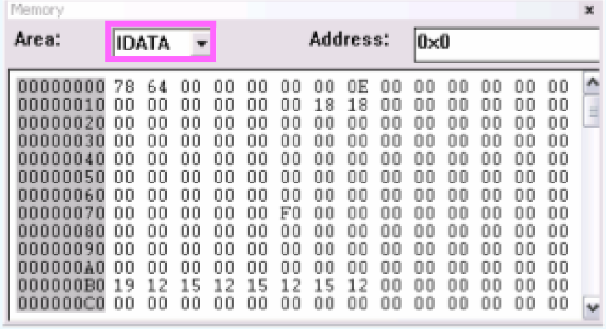
Area: CODE Address: 0x100

```
00000100          FD 00 74 AA F0 12 01 5D 90 FD 00
00000110          12 01 5D DF EA 7F 02 90 FE 00 74
00000120          74 06 F0 12 01 5D 74 5B F0 12 01
00000130 4F F0 12 01 5D 01 33 01 33 00 00 FF 0F 03
00000140 FF 80 80 00 00 00 10 00 00 00 00 00 00 01
00000150 00 00 00 E0 00 01 FF 10 00 33 01 00 00 33
00000160 FF 0F 03 01 01 FF 80 80 00 00 00 10 00 00
00000170 00 00 01 48 00 00 00 00 E0 00 01 FF 10 00
00000180 7F 0E 90 FD 00 74 AA F0 12 01 5D 90 FD 00
00000190 F0 78 FF 12 01 5D DF EA 7F 02 90 FE 00 74
000001A0 12 01 5D 74 06 F0 12 01 5D 74 5B F0 12 01
000001B0 4F F0 12 01 5D 10 00 00 00 00 00 00 01 48
000001C0 00 00 E0 00 01 FF 10 00 33 01 00 00 33 00
```



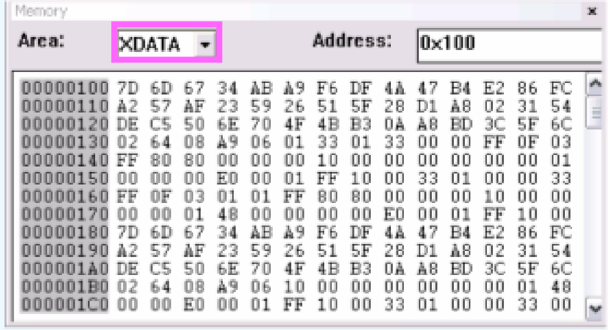
Area: CODE Address: 0x100

```
00000100 7F 0E 90 FD 00 74 AA F0 12 01 5D 90 FD 00
00000110 F0 78 FF 12 01 5D DF EA 7F 02 90 FE 00 74
00000120 12 01 5D 74 06 F0 12 01 5D 74 5B F0 12 01
00000130 4F F0 12 01 5D 01 33 01 33 00 00 FF 0F 03
00000140 FF 80 80 00 00 00 10 00 00 00 00 00 00 01
00000150 00 00 00 E0 00 01 FF 10 00 33 01 00 00 33
00000160 FF 0F 03 01 01 FF 80 80 00 00 00 10 00 00
00000170 00 00 01 48 00 00 00 00 E0 00 01 FF 10 00
00000180 7F 0E 90 FD 00 74 AA F0 12 01 5D 90 FD 00
00000190 F0 78 FF 12 01 5D DF EA 7F 02 90 FE 00 74
000001A0 12 01 5D 74 06 F0 12 01 5D 74 5B F0 12 01
000001B0 4F F0 12 01 5D 10 00 00 00 00 00 00 01 48
000001C0 00 00 E0 00 01 FF 10 00 33 01 00 00 33 00
```



Area: IDATA Address: 0x0

```
00000000 78 64 00 00 00 00 00 0E 00 00 00 00 00 00
00000010 00 00 00 00 00 00 18 18 00 00 00 00 00 00
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070 00 00 00 00 00 F0 00 00 00 00 00 00 00 00
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000B0 19 12 15 12 15 12 15 12 00 00 00 00 00 00
000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



Area: XDATA Address: 0x100

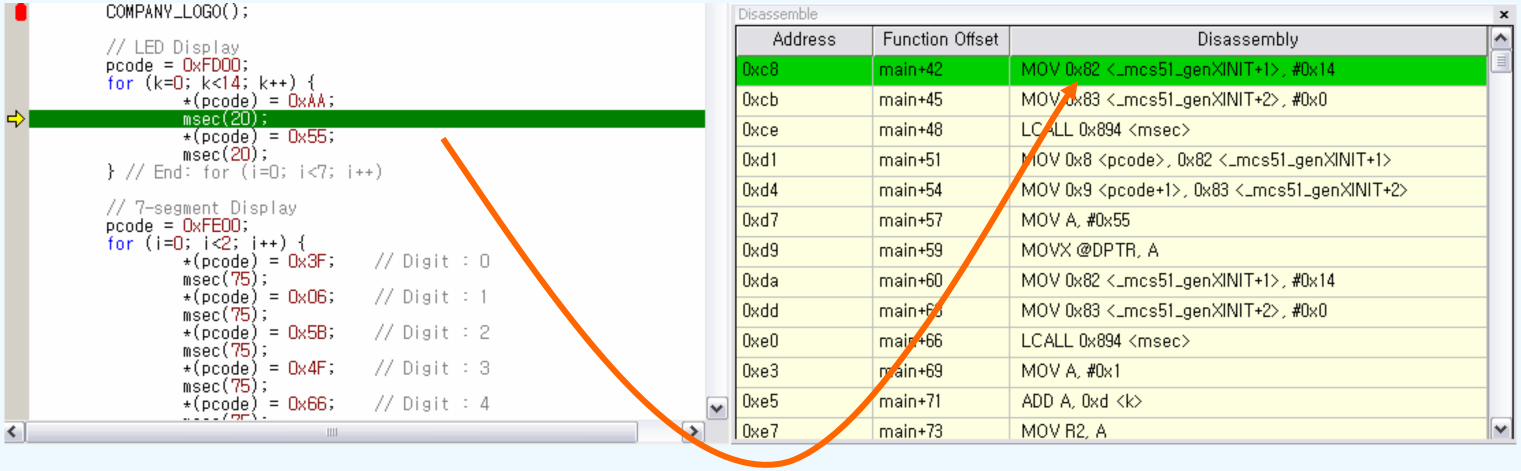
```
00000100 7D 6D 67 34 AB A9 F6 DF 4A 47 B4 E2 86 FC
00000110 A2 57 AF 23 59 26 51 5F 28 D1 A8 02 31 54
00000120 DE C5 50 6E 70 4F 4B B3 0A A8 BD 3C 5F 6C
00000130 02 64 08 A9 06 01 33 01 33 00 00 FF 0F 03
00000140 FF 80 80 00 00 00 10 00 00 00 00 00 00 01
00000150 00 00 00 E0 00 01 FF 10 00 33 01 00 00 33
00000160 FF 0F 03 01 01 FF 80 80 00 00 00 10 00 00
00000170 00 00 01 48 00 00 00 00 E0 00 01 FF 10 00
00000180 7D 6D 67 34 AB A9 F6 DF 4A 47 B4 E2 86 FC
00000190 A2 57 AF 23 59 26 51 5F 28 D1 A8 02 31 54
000001A0 DE C5 50 6E 70 4F 4B B3 0A A8 BD 3C 5F 6C
000001B0 02 64 08 A9 06 10 00 00 00 00 00 00 01 48
000001C0 00 00 E0 00 01 FF 10 00 33 01 00 00 33 00
```

2. View Windows : 8) Disassembly Window

- Displays the target program in disassembled form and additionally source lines, labels, symbols, and symbol values.

Edit Window

Disassembly Window



```
        COMPANY_LOGO();

        // LED Display
        pcode = 0xFD00;
        for (k=0; k<14; k++) {
                *(pcode) = 0xAA;
                msec(20);
                *(pcode) = 0x55;
                msec(20);
        } // End: for (i=0; i<7; i++)

        // 7-segment Display
        pcode = 0xFE00;
        for (i=0; i<2; i++) {
                *(pcode) = 0x3F;    // Digit : 0
                msec(75);
                *(pcode) = 0x06;    // Digit : 1
                msec(75);
                *(pcode) = 0x5B;    // Digit : 2
                msec(75);
                *(pcode) = 0x4F;    // Digit : 3
                msec(75);
                *(pcode) = 0x66;    // Digit : 4
```

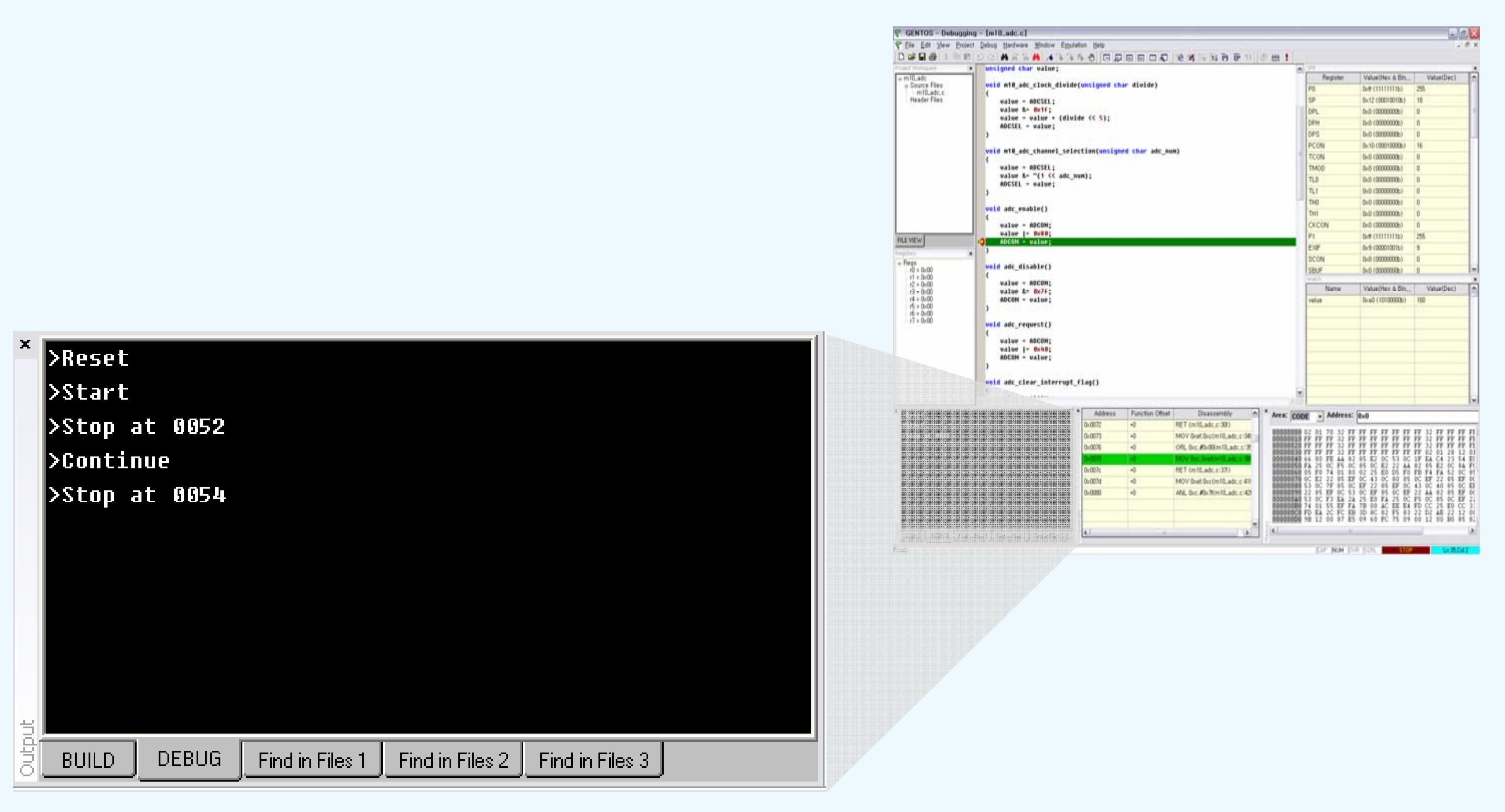
Disassemble

Address	Function Offset	Disassembly
0xc8	main+42	MOV 0x82 <_mcs51_genXINIT+1>, #0x14
0xcb	main+45	MOV 0x83 <_mcs51_genXINIT+2>, #0x0
0xce	main+48	LCALL 0x894 <msec>
0xd1	main+51	MOV 0x8 <pcode>, 0x82 <_mcs51_genXINIT+1>
0xd4	main+54	MOV 0x9 <pcode+1>, 0x83 <_mcs51_genXINIT+2>
0xd7	main+57	MOV A, #0x55
0xd9	main+59	MOVX @DPTR, A
0xda	main+60	MOV 0x82 <_mcs51_genXINIT+1>, #0x14
0xdd	main+63	MOV 0x83 <_mcs51_genXINIT+2>, #0x0
0xe0	main+66	LCALL 0x894 <msec>
0xe3	main+69	MOV A, #0x1
0xe5	main+71	ADD A, 0xd <k>
0xe7	main+73	MOV R2, A

C-Code Convert Assemble Code

2. View Windows : 9) Output Window

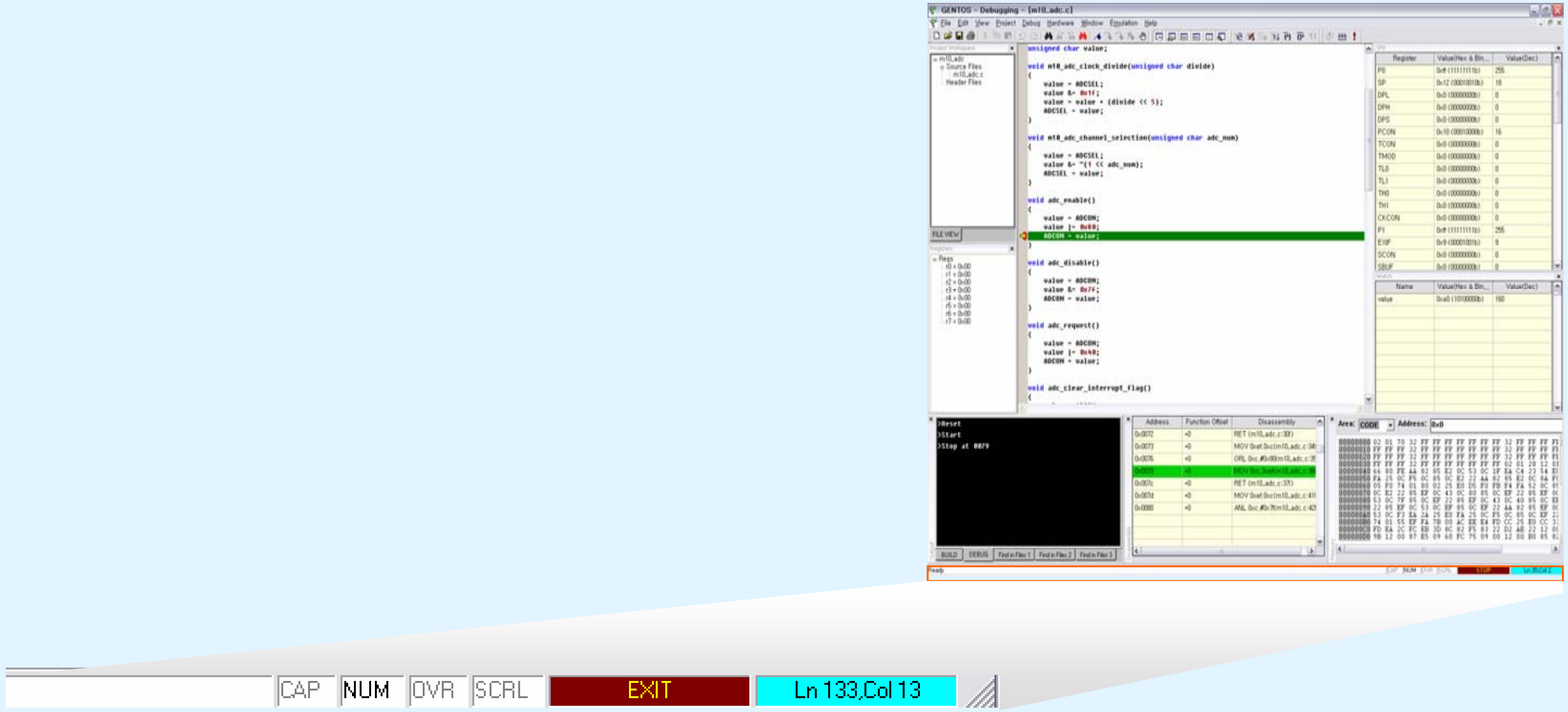
- Displays state of code.
  - ✓ Each small window DEBUG, BUILD, Find in Files1~3 display State of program run



```
>Reset
>Start
>Stop at 0052
>Continue
>Stop at 0054
```

2. View Windows : 10) Output Window

- Displays state of MiDAS.



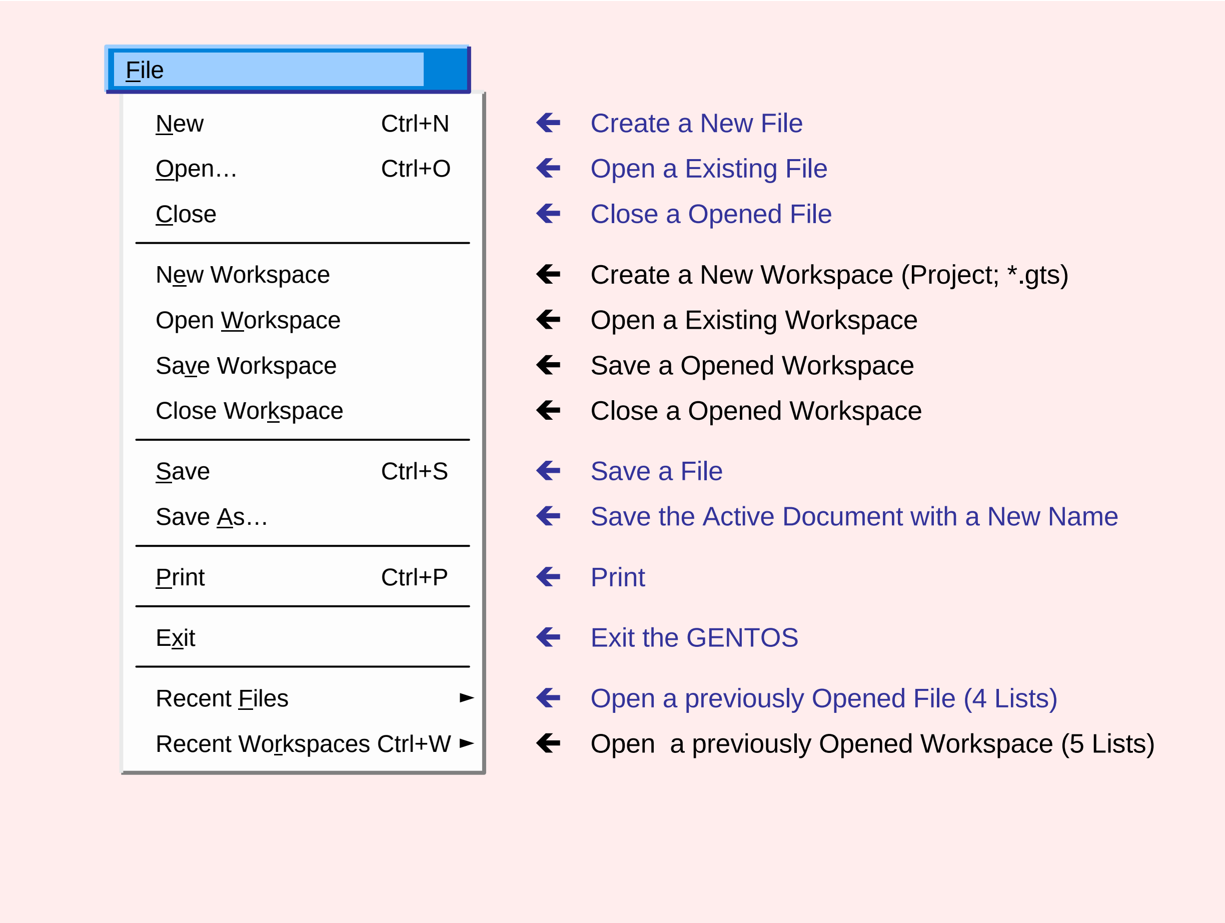
```
unsigned char value;

void m10_adc_clock_divide(unsigned char divide)
{
    value = ADCSEL;
    value &= 0x1f;
    value = value + (divide << 5);
    ADCSEL = value;
}

void m10_adc_channel_selection(unsigned char adc_num)
{
    value = ADCSEL;
    value &= ~(1 << adc_num);
    ADCSEL = value;
}

void adc_enable()
{
    value = ADCON;
    value |= 0x80;
    ADCON = value;
}

void adc_disable()
{
    value = ADCON;
    value &= 0x7f;
    ADCON = value;
}

void adc_request()
{
    value = ADCON;
    value |= 0x40;
    ADCON = value;
}

void adc_clear_interrupt_flag()
{
```

```
>Reset
>Start
>Stop at 0079
```

Display Keyboard Status & Code Line

3. Menus : 1) File

- The menu “File” supports the management of files & Workspaces.
  - ✓ Command : Open, Save, Save as, Close, Print, and Exit
  - ✓ Additional Service : the Recently Opened File List and Workspace List

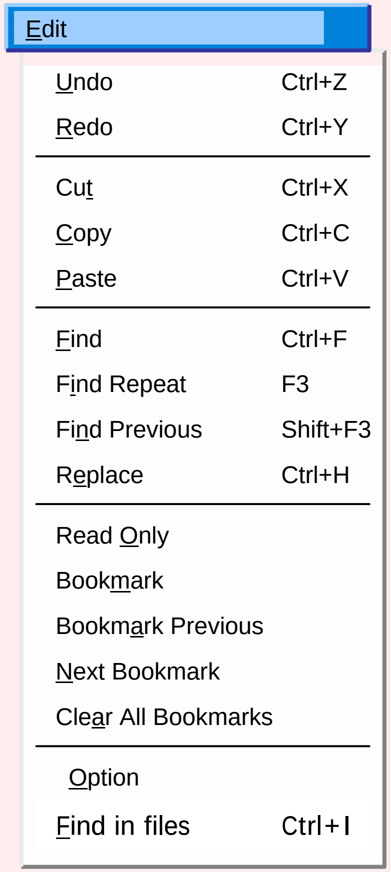


File

New	Ctrl+N	Create a New File
Open…	Ctrl+O	Open a Existing File
Close		Close a Opened File
New Workspace		Create a New Workspace (Project; *.gts)
Open Workspace		Open a Existing Workspace
Save Workspace		Save a Opened Workspace
Close Workspace		Close a Opened Workspace
Save	Ctrl+S	Save a File
Save As…		Save the Active Document with a New Name
Print	Ctrl+P	Print
Exit		Exit the GENTOS
Recent Files ►		Open a previously Opened File (4 Lists)
Recent Workspaces Ctrl+W ►		Open a previously Opened Workspace (5 Lists)

3. Menus : 2) Edit

- The menu “Edit” supports the Documentation for source code.
  - ✓ Command : Undo, Redo, Cut, Copy, Paste, Find, Replace, and Bookmark
  - ✓ Additional Service : Option for GENTOS customizing



← Undo

← Redo

← Cut the Selection and Put It on Clipboard

← Copy the Selection and Put It on Clipboard

← Paste the Currently Copied Code

← Find the Specified Code

← Find Repeat

← Find Previous

← Replace the Code

← Apply the File to Read Only Mode ON/OFF

← Toggle the Bookmark

← Go the Previous Bookmark

← Go the Next Bookmark

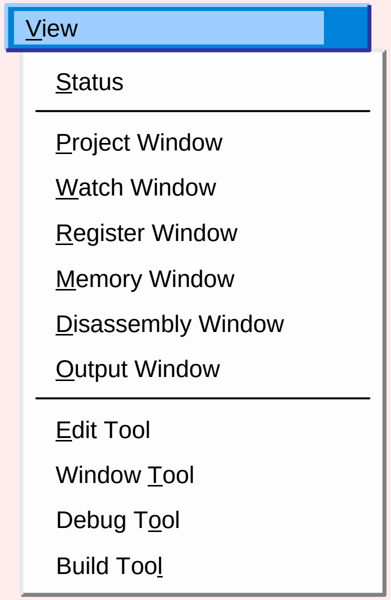
← Clear All Bookmarks

← Customize the Environment of Editor

← Fine in files

3. Menus : 3) View

- The menu “View” supports the active/inactive for view windows and toolbars.
  - ✓ Supported Window : Project, Watch, Register (SFRs), Memory, Disassembly, and Output
  - ✓ Additional Service : Status Toolbar, Edit Toolbar, and Debug Toolbar



← Show or Hide the “Status Bar”

← Show or Hide the “Project” Window

← Show or Hide the “Watch” Window

← Show or Hide the “Register” (SFRs) Window

← Show or Hide the “Memory” Window

← Show or Hide the “Disassembly” Window

← Show or Hide the “Output” Window

← Show or Hide the “Edit Tool”

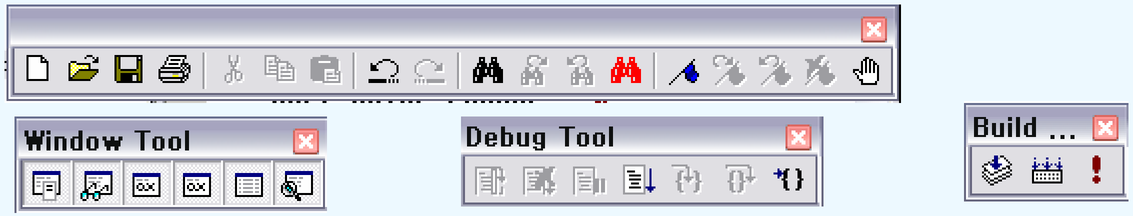
← Show or Hide the “Window Tool”

← Show or Hide the “Debug Tool”

← Show or Hide the “Build Tool”

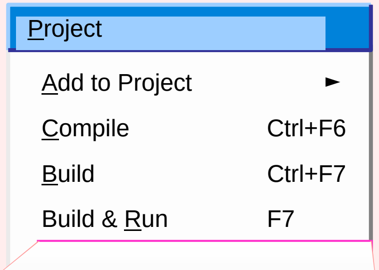
Refer to “4. Toolbars”

Edit Tool



3. Menus : 4) Project

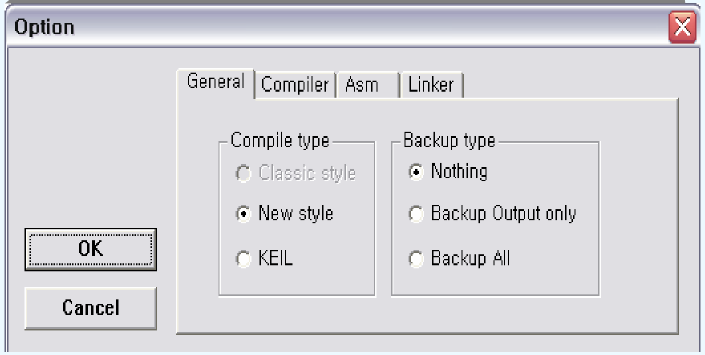
- The menu “Project” supports the management of project.
  - ✓ Command : Add to Project, Build, and Option



Project

Add to Project ►	
Compile	Ctrl+F6
Build	Ctrl+F7
Build & Run	F7

- ← Add the New or Existed File to Project
- ← Compile the Project
- ← Build the Project
- ← Build & Run the Project
- ← Customize the Environment of Tools
- ← Make Library
- ← Download the code to the MCU Chip using GenICE52
- ← Clean Project



Option

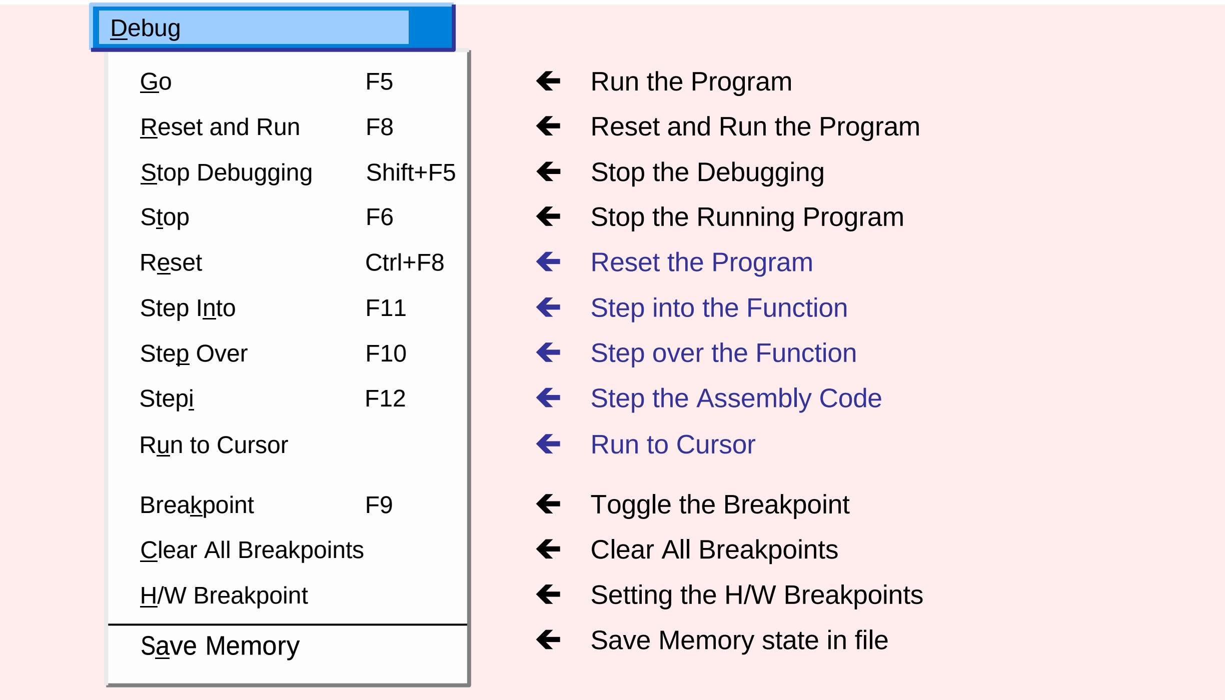
Compile type: ○ Classic style ● New style ○ KEIL

Backup type: ● Nothing ○ Backup Output only ○ Backup All

- Options : For more detail, refer to Appendix D, F, and H.
  - ✓ Compiler
    - Path : `C:\GENTOS\BIN\GENCC.EXE`
    - Options : `--debug` (default)
  - ✓ Assembler
    - Path : `C:\GENTOS\BIN\GENASM.EXE`
    - Options : `-gstabs` (default)
  - ✓ Linker
    - Path : `C:\GENTOS\BIN\GENLNK.EXE`
    - Options : (default : none)
  - ✓ Debugger
    - Path : `C:\GENTOS\BIN\GENDB.EXE`
    - Option : `-f` (default)

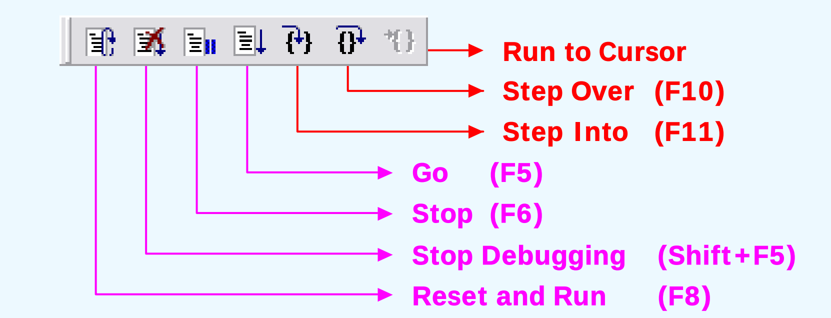
3. Menus : 5) Debug

- The menu “Debug” supports the debugging functions for project.
  - ✓ Command : Go, Reset and Run, Stop, Reset, Step, and Breakpoint



Debug

Command	Shortcut	Description
Go	F5	Run the Program
Reset and Run	F8	Reset and Run the Program
Stop Debugging	Shift+F5	Stop the Debugging
Stop	F6	Stop the Running Program
Reset	Ctrl+F8	Reset the Program
Step Into	F11	Step into the Function
Step Over	F10	Step over the Function
Stepi	F12	Step the Assembly Code
Run to Cursor		Run to Cursor
Breakpoint	F9	Toggle the Breakpoint
Clear All Breakpoints		Clear All Breakpoints
H/W Breakpoint		Setting the H/W Breakpoints
Save Memory		Save Memory state in file



Run to Cursor

Step Over (F10)

Step Into (F11)

Go (F5)

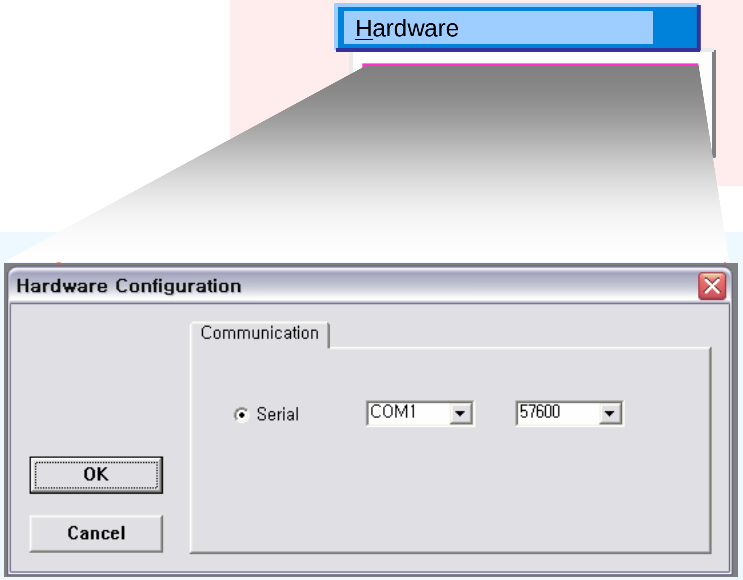
Stop (F6)

Stop Debugging (Shift+F5)

Reset and Run (F8)

3. Menus : 6) Hardware

- The menu “Hardware/Hardware Configuration” supports the setting for GENSYS 52 Hardware.
  - ✓ Options : Communication, ROM, RAM, Clock, and RESET



Hardware Configuration

Communication

● Serial COM1 57600

OK

Cancel

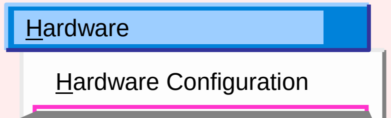
← Setting the Configuration for GENSYS 52

← Setting the MCU Device List

- Hardware Configuration
  - ✓ Communication
    - Serial : COM1(Default), 57600 (Default)
  - ✓ ROM
    - Option : Emulator (Default) or Target ROM
  - ✓ RAM
    - Option : Emulator or Target RAM (Default)
  - ✓ Clock
    - Frequency : 1.843MHz ~ 24.576MHz
    - Option : Emulator (Default) or Target Clock
  - ✓ RESET
    - Option : Emulator (Default) or Target Reset

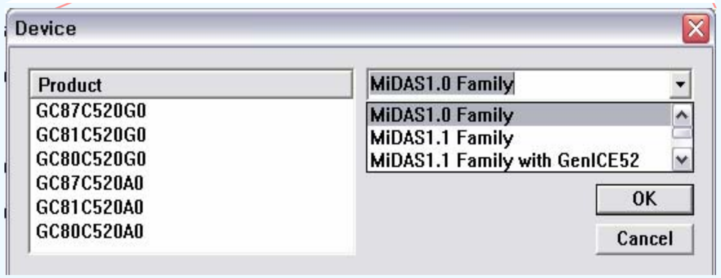
3. Menus : 6) Hardware (Cont'd)

- The menu "Hardware/Device" supports the setting for MCU Device.
  - ✓ Options : MiDAS1.0 Family, MiDAS1.1 Family, and MiDAS2.0 Family
  - ✓ This information is saved to Project (Workspace) file.



← Setting the Configuration for GENSYS 52

← Setting the MCU Device List


Device

Product

GC87C520G0
GC81C520G0
GC80C520G0
GC87C520A0
GC81C520A0
GC80C520A0

MiDAS1.0 Family

MiDAS1.0 Family
MiDAS1.1 Family
MiDAS1.1 Family with GenICE52

MCU Device List (Family/Series)

✓ **MiDAS1.0 Family**

◆ GC87C520G0	: General MCU, EPROM 8K
◆ GC81C520G0	: General MCU, MASK ROM 8K
◆ GC80C520G0	: General MCU, ROMless
◆ GC87C520A0	: ADC Application MCU, EPROM 8K
◆ GC81C520A0	: ADC Application MCU, MASK ROM 8K
◆ GC80C520A0	: ADC Application MCU, ROMless

✓ **MiDAS1.1 Family**

◆ GC87C510A0	: Low-cost ADC Application MCU, EPROM 4K(OTP)
◆ GC81C510A0	: Low-cost ADC Application MCU, MASK ROM 4K

✓ **MiDAS2.0 Family – G Type, AType**

- ◆ GC89C591G0-PL44/TQ44
- ◆ GC81C591G0-PL44/TQ44
- ◆ GC89C591A0-TQ64/80/100
- ◆ GC81C591A0-TQ64/80/100

GC89C591A0-PL44/TQ44

- ◆ GC89C591A0-PL44/TQ44

✓ **MiDAS2.1 Family - GC89C520A0**

✓ **AFCore Family - AFCore 1.0**

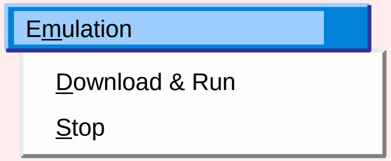
✓ **TPMS**

✓ **RoboCore**

3. Menus : 7) Emulation

- The menu “Emulation” supports the ROM Emulator with HEX File.
  - ✓ Objective :
    When you have only a HEX file (*.hex or *.ihex), you can download it to GENSYS 52 and run it without source-level debugging.
  - ✓ Caution :
    You only run a HEX file.

- Procedure
  - ✓ You have to set up the hardware menu. (Refer to the “3.Menus : 6) Hardware” slide)
    - Hardware/Hardware Configuration
    - Hardware/Device
  - ✓ Select the menu “Emulation → Download and Run”.
  - ✓ When you want to stop the running GENSYS 52, select the menu “Emulation → Stop”.



← Download the your HEX file and Run GENSYS 52

← Stop the Running GENSYS 52

3. Menus : 8) Accelerator Keys

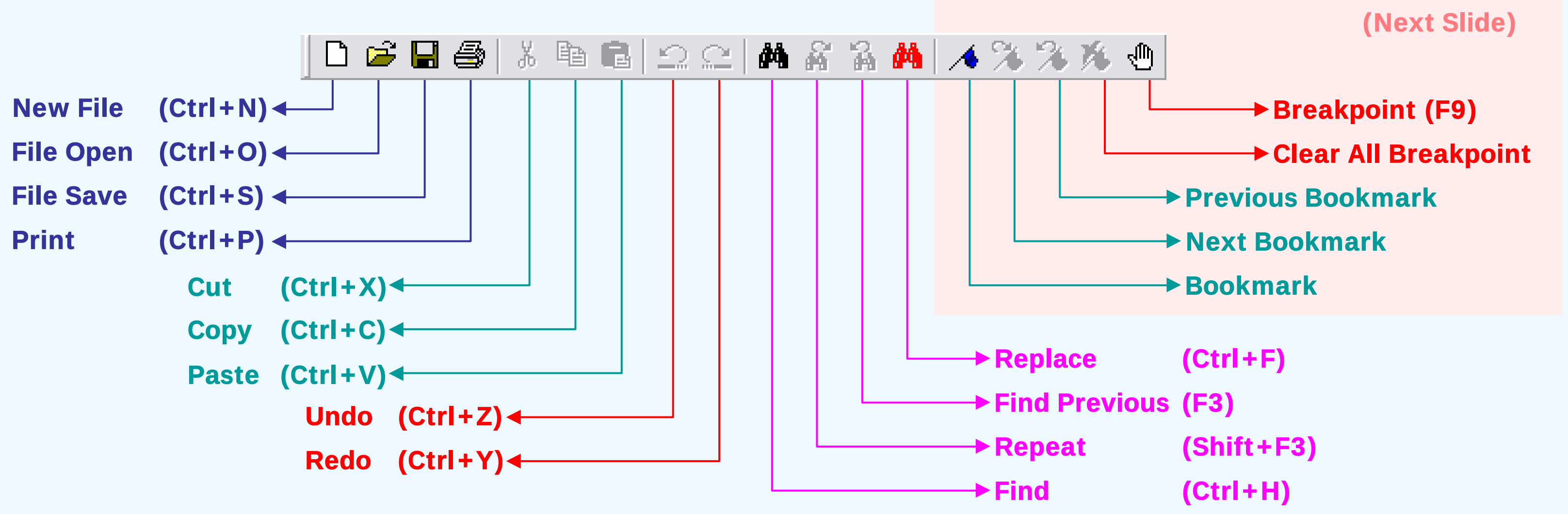
Top Menu	Sub Menu	Accelerator Key	Descriptions
File	New	Control + N	Create the New File
	Open...	Control + O	Open the Existing File
	Save	Control + S	Close the Opened File
	Print	Control + P	Print
	Recent Workspace	Control + W	Open the Recently Opened Workspace
Edit	Undo	Control + Z	Undo
	Redo	Control + Y	Redo
	Cut	Control + X	Cut the Selection and Put It on Clipboard
	Copy	Control + C	Copy the Selection and Put It on Clipboard
	Paste	Control + V	Paste the Currently Copied Code
	Find	Control + F	Find the Specified Code
	Find Repeat	F3	Find Repeat
	Find in files	Control + I	Find things in file
	Find Previous	Shift + F3	Find Previous
	Replace	Control + H	Replace the Code
Project	Build	Control + F7	Build the Project
	Build & Run	F7	Build and program Run
	Complie	Control + F6	Complie to source Code
	Rebuild All	Shift + F7	Remove object file and re Build All
Debug	Go	F5	Run the Program
	Reset and Run	F8	Reset and Run the Program
	Stop Debugging	Shift + F5	Stop the Debugging
	Stop	F6	Stop the Running Program
	Step Into	F11	Step into the Function
	Step Over	F10	Step over the Function
	Stepi	F12	Step the Assembly Code
	Breakpoint	F9	Toggle the Breakpoint

4. Toolbars : 1) Edit Tool

- Toolbars is consist of three parts :
  - ✓ Edit Tool
  - ✓ Debug Tool
  - ✓ Window Tool

- Edit Tool
  - ✓ File Management :
    - New, Open, Save, and Print.
  - ✓ Code Manipulation
    - Cut, Copy, and paste
  - ✓ Command Undo & Redo
  - ✓ Find Command
  - ✓ Bookmark & Breakpoint Setting

[Edit Tool]



(Next Slide)

New File (Ctrl+N)
File Open (Ctrl+O)
File Save (Ctrl+S)
Print (Ctrl+P)
Cut (Ctrl+X)
Copy (Ctrl+C)
Paste (Ctrl+V)
Undo (Ctrl+Z)
Redo (Ctrl+Y)
Breakpoint (F9)
Clear All Breakpoint
Previous Bookmark
Next Bookmark
Bookmark
Replace (Ctrl+F)
Find Previous (F3)
Repeat (Shift+F3)
Find (Ctrl+H)

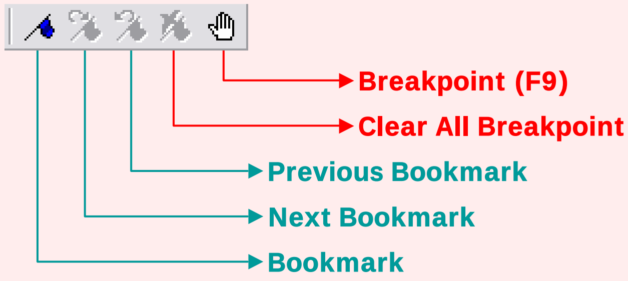
4. Toolbars : 1) Edit Tool (Cont'd)

- Unlimited Number of Breakpoints
  - ✓ PC breakpoint by program address
  - ✓ H/W breakpoint by external memory access

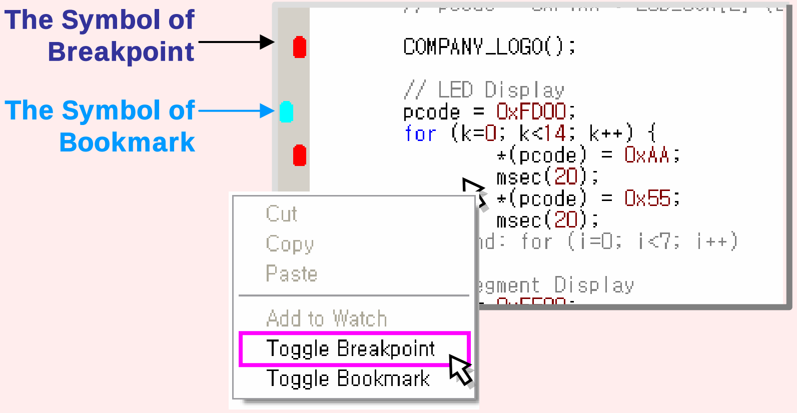
- How to Enable/Disable Breakpoints
  - ✓ Menu "Debug → Breakpoint",
  - ✓ "F9" key,
  - ✓ "Tool button" from the toolbar , or
  - ✓ "Toggle Breakpoint" from pop-up menu.

- Hot to Clear All Breakpoints
  - ✓ Menu "Debug → Clear All Breakpoints" or
  - ✓ Click the tool button" from "Edit Tool".

[Edit Tool : Breakpoint & Bookmark Buttons]



[The Symbols & Pop-up Menu]



```
COMPANY_LOGO();

// LED Display
pcode = 0xFD00;
for (k=0; k<14; k++) {
        *(pcode) = 0xAA;
        msec(20);
        *(pcode) = 0x55;
        msec(20);
 nd: for (i=0; i<7; i++)

 egment Display
```

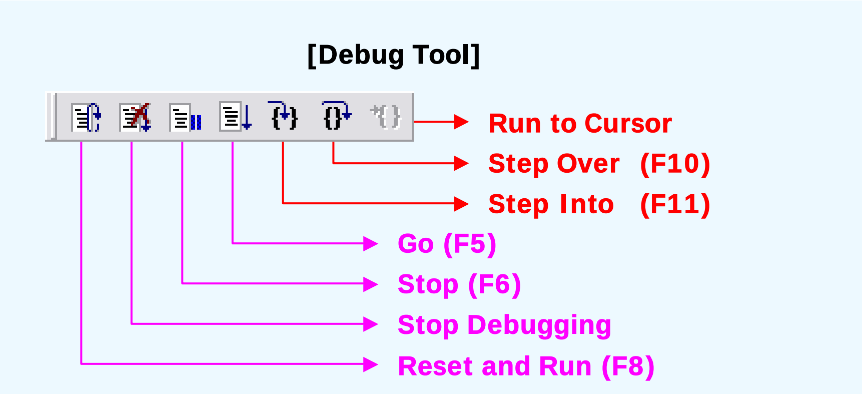
4. Toolbars : 2) Debug Tool & Window Tool

- Debug Tool
  - ✓ Reset and Run
  - ✓ Stop
  - ✓ Stop Debugging
  - ✓ Go
  - ✓ Step Into
  - ✓ Step Over

- Window Tool : show or hide the
  - ✓ Project Window
  - ✓ Watch Window
  - ✓ Register (SFRs) Window
  - ✓ Memory Window
  - ✓ Disassembly Window

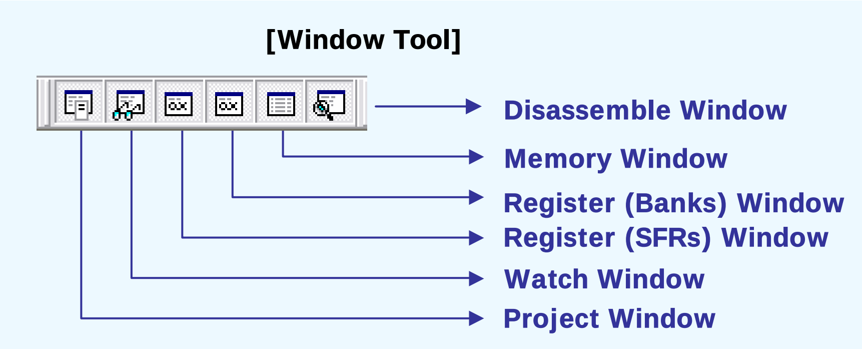
- Build Tool
  - ✓ Compile
  - ✓ Build
  - ✓ Build & Run

Jerry : Build & Run (← Execute)



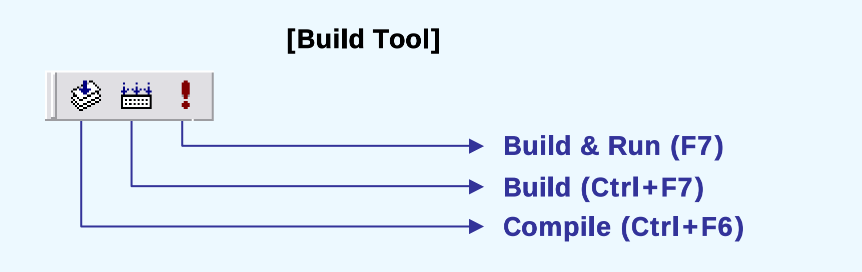
[Debug Tool]

Run to Cursor
Step Over (F10)
Step Into (F11)
Go (F5)
Stop (F6)
Stop Debugging
Reset and Run (F8)



[Window Tool]

Disassemble Window
Memory Window
Register (Banks) Window
Register (SFRs) Window
Watch Window
Project Window



[Build Tool]

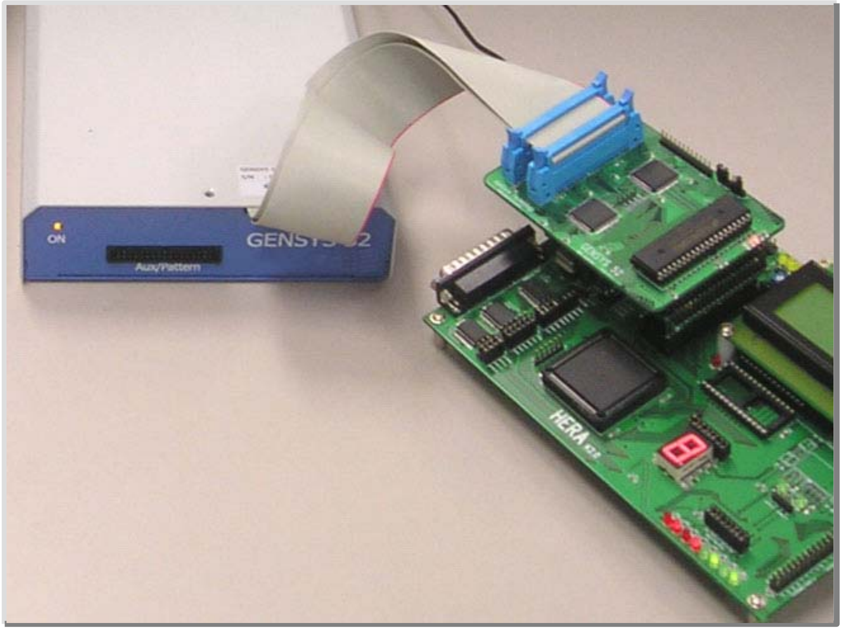
Build & Run (F7)
Build (Ctrl+F7)
Compile (Ctrl+F6)

PART III : GENSYS52 MDS for MiDAS1.0

1. Overview : GENSYS52 MDS for MiDAS1.0
2. Configuration
3. Jumper Setting of POD
4. POD Adaptors for MiDAS1.0

1. Overview : GENSYS52 MDS for MiDAS1.0

- Clock: 1.843MHz ~ 24.576MHz
  - ✓ Using Programmable Emulator Clock
- Host Interface
  - ✓ Serial Port : COM1 ~ COM4
- PC Breakpoint : Full Range
- H/W Breakpoint : Full Range
- Multiple Voltage : 3.3V, 5V
- Various Emulation Configurations
  - ✓ ROM : Emulator or Target
  - ✓ RAM : Emulator or Target
  - ✓ Clock : Emulator or Target
  - ✓ Reset : Emulator or Target
- POD Socket
  - ✓ 40-pin DIP
- Adaptors
  - ✓ 44-pin PLCC
  - ✓ 28-pin DIP

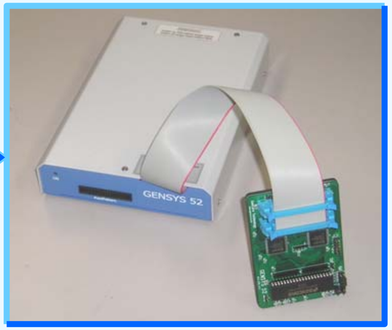


2. Configuration

Configuration for MDS Environment

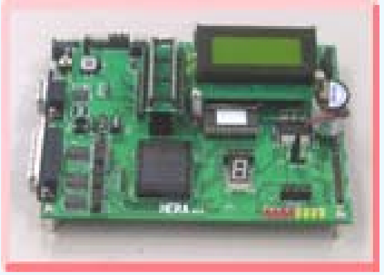
[GENTOS Program on PC Host]

Cable Assembly



[GENSYS52 MDS for MiDAS1.0]

Connect the POD to Target System



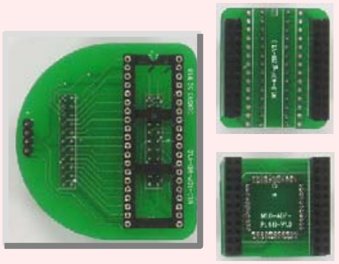
[User's Target System]

Accessories

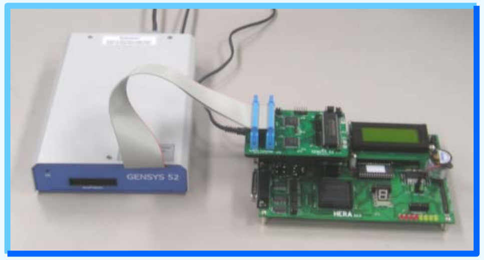
Cable
(1.5 meter)

Power Adaptor
(SMPS, 9V, 3Ah)

Adaptor for MDAS1.1(Option)
to 44-pin PLCC or 28-pin DIP



*Refer to "4. Adaptors for POD"

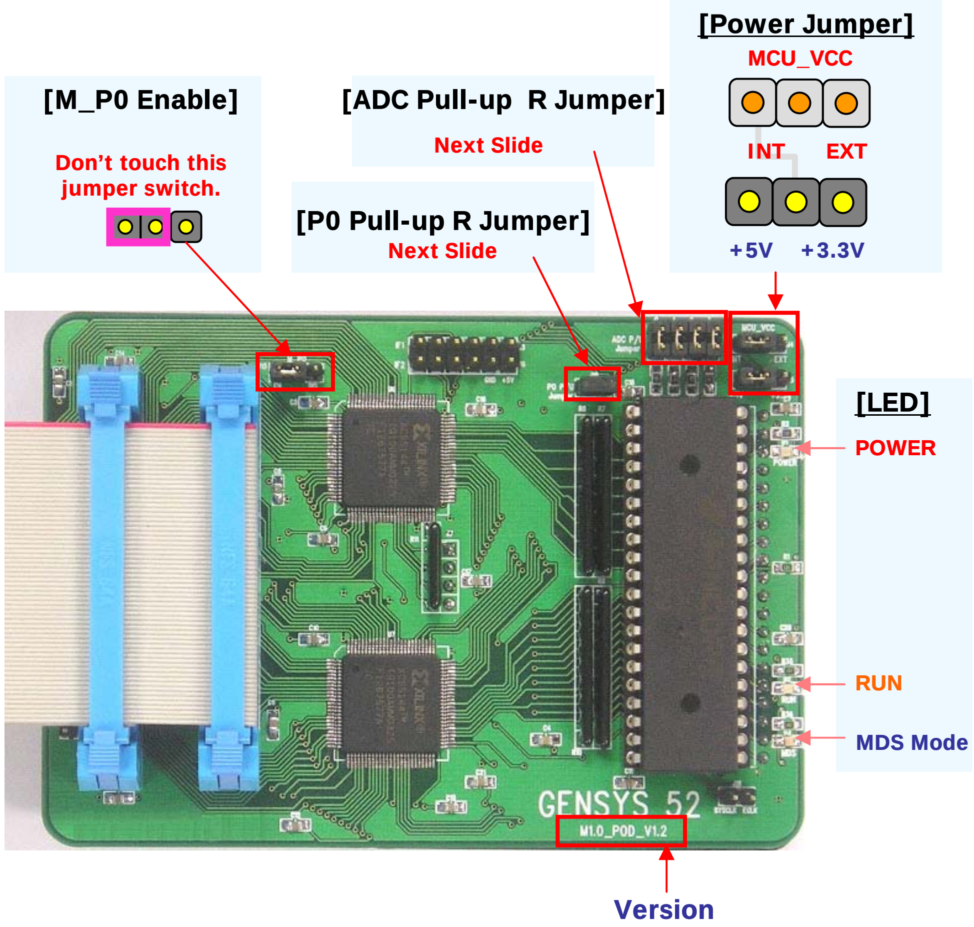


[Under the test]

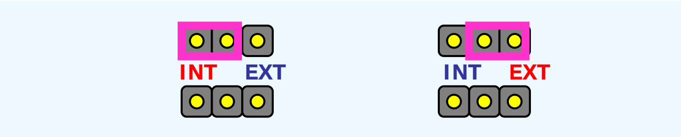
* Please, supply the power to the target system.

3. Jumper Setting of POD

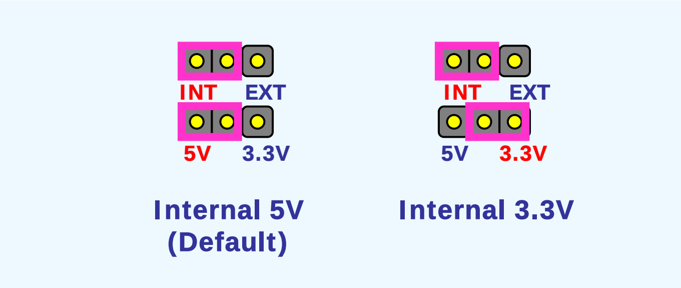
- The voltage options of MCU is supported with jumper switches of POD board.



- The Selection of MCU’s Voltage.
  - ✓ Select the MCU’s VCC power to “Internal” or “External” using Jumper CAP.



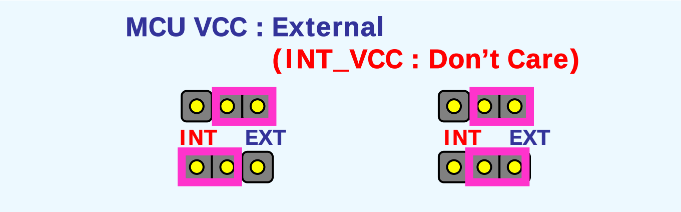
  - ✓ When you selected “Internal” power, you can select 3.3 [V] or 5 [V].



Internal 5V (Default)

Internal 3.3V

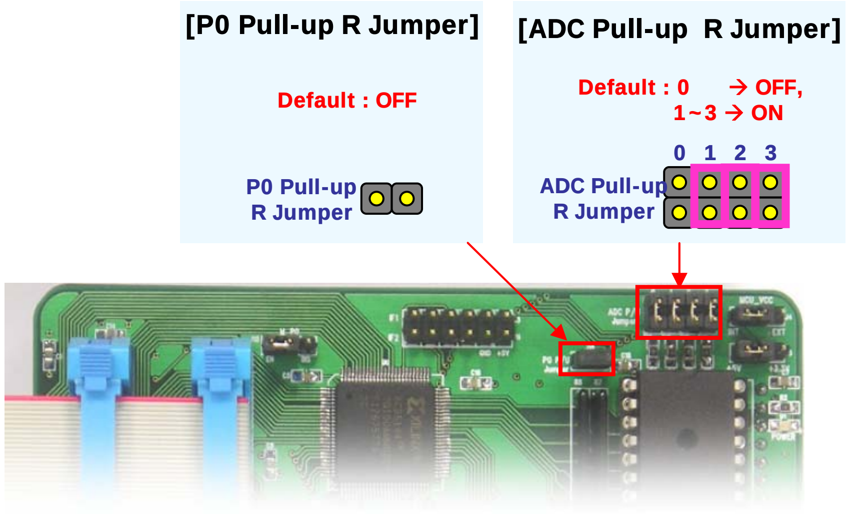
  - ✓ When you selected “External” power, INT_VCC is “Don’t Care”.



MCU VCC : External
(INT_VCC : Don’t Care)

3. Jumper Setting of POD (Cont'd)

- ◆ The Pull-up R setting for P1(except for P1.1, P1.2, and P1.3), P2, P3, and P4 ports is fixed to VCC.
  (MDS cannot support the these Pull-up R 'off' setting.)
- ◆ The Pull-up R setting for P0, P1.1, P1.2, and P1.3 is supported using the additional jumper switches.



[P0 Pull-up R Jumper]

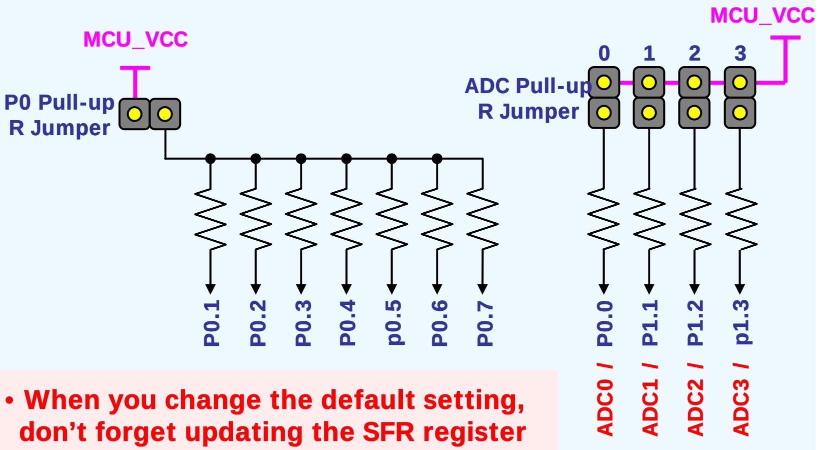
Default : OFF

[ADC Pull-up R Jumper]

Default : 0 → OFF, 1~3 → ON

- ◆ P0 Pull-up R Jumper
  - ✓ When P0 port is used as address, P0 Pull-up R jumper is opened.
  - ✓ When P0 port is used as I/O port, P0 Pull-up R jumper is connected to MCU_VCC with jumper.
  - ✓ Default : OFF (open)
- ◆ ADC Pull-up R Jumper
  - ✓ When the 4 IO inputs (P0.1, P1.1, P1.2, and (or) P1.3) are used as ADC input channel, ADC Pull-up R jumper is opened.
  - ✓ Default : **0** → OFF, **1~3** → ON

[Schematic for Pull-up R jumpers]

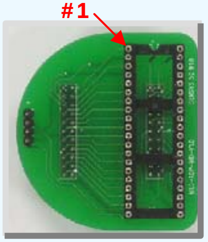


- When you change the default setting, don't forget updating the SFR register (P0SEL and P1SEL) together.

4. POD Adaptor for MiDAS1.0

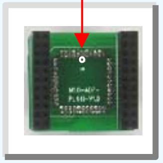
- The connection of POD board is supported to 40-pin DIP socket. (Default part)
- The connection for 44-pin PLCC socket or 28-pin DIP socket is supported using adaptors.

Adaptor
(Middle Part)



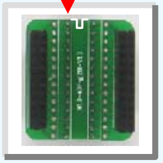
+

Adaptor
(Bottom Part)

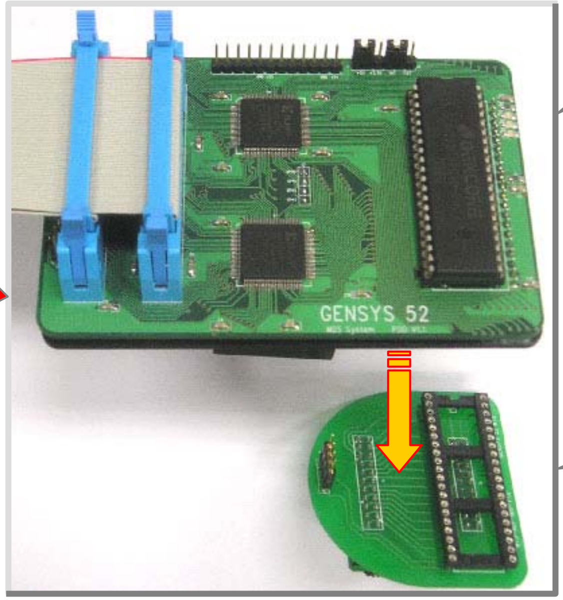


Adaptor for
44-pin PLCC

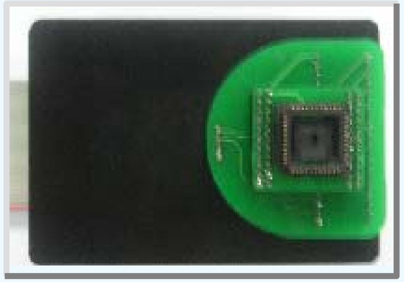
or



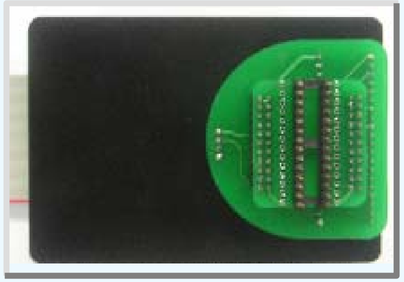
Adaptor for
28-pin DIP



The POD assembled with Adaptor
for 44-pin PLCC Type



The POD assembled with Adaptor
for 28-pin DIP Type



PART IV : GENSYS52 MDS for MiDAS1.1

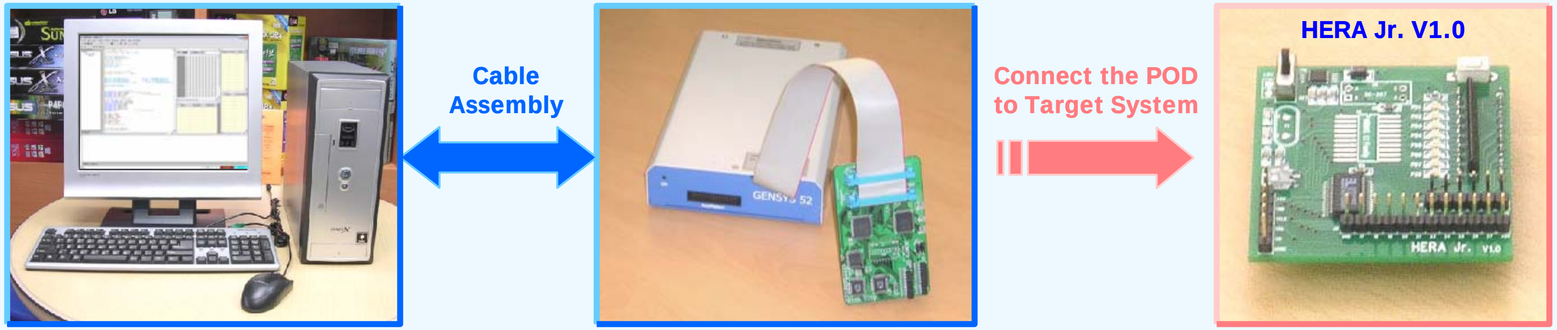
1. Overview : GENSYS52 MDS for MiDAS1.1
2. Configuration
3. Jumper Setting of POD

1. Overview : GENSYS52 MDS for MiDAS1.1

- Clock: 1.832MHz ~ 24.576MHz
  - ✓ Using Programmable Emulator Clock
- Host Interface
  - ✓ Serial Port : COM1 ~ COM4
- PC Breakpoint : Full Range
- H/W Breakpoint : Full Range
- Multiple Voltage : 3.3V, 5V
- Various Emulation Configurations
  - ✓ ROM : Emulator
  - ✓ RAM : Emulator
  - ✓ Clock : Emulator or Target
  - ✓ Reset : Emulator or Target
- POD Socket
  - ✓ 20-pin SPDIP

2. Configuration

- Configuration for MDS Environment



[GENTOS Program on PC Host]

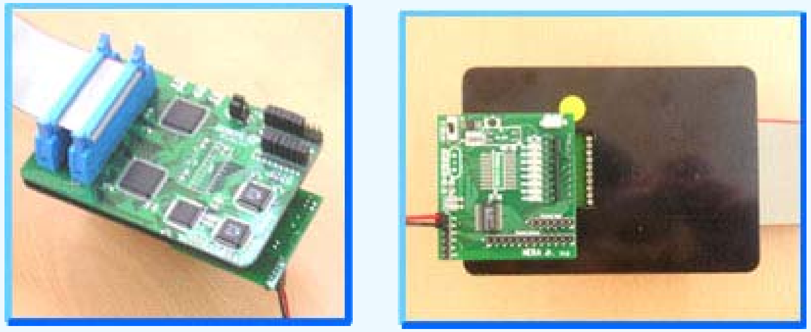
[GENSYS52 MDS for MiDAS1.1]

[User's Target System]

- Accessories

Cable
(1.5 meter)

Power Adaptor
(SMPS, 9V, 3Ah)

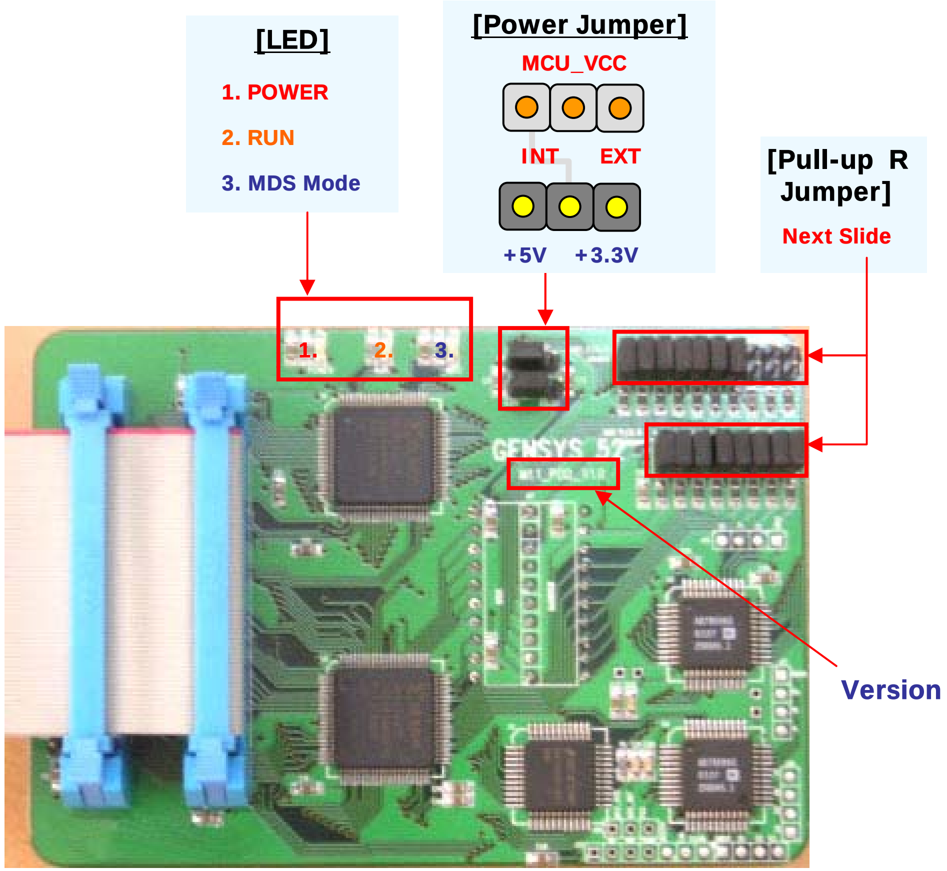


[Under the test]

* Please, supply the power to the target system.

3. Jumper Setting of POD

- The voltage options of MCU is supported with jumper switches of POD board.



[LED]

1. POWER
2. RUN
3. MDS Mode

[Power Jumper]

MCU_VCC

INT EXT

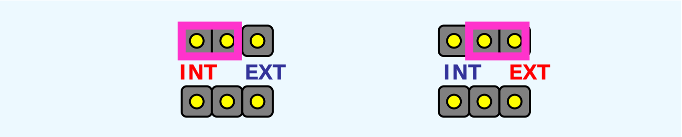
+5V +3.3V

[Pull-up R Jumper]

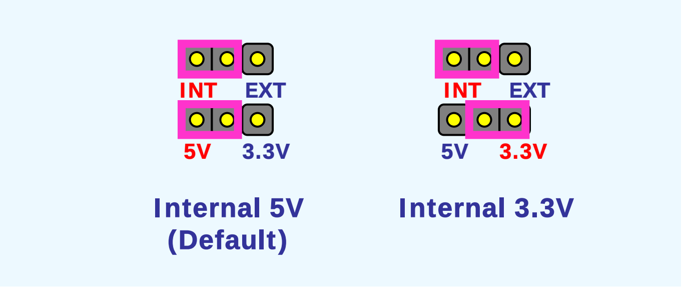
Next Slide

Version

- The Selection of MCU's Voltage.
  - ✓ Select the MCU's VCC power to "Internal" or "External" using Jumper CAP.



  - ✓ When you selected "Internal" power, you can select 3.3 [V] or 5 [V].



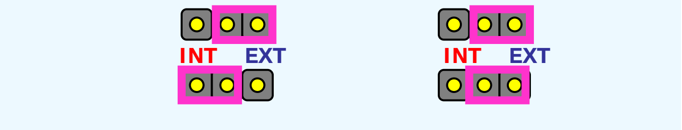
Internal 5V (Default)

Internal 3.3V

  - ✓ When you selected "External" power, INT_VCC is "Don't Care".

MCU VCC : External

(INT_VCC : Don't Care)

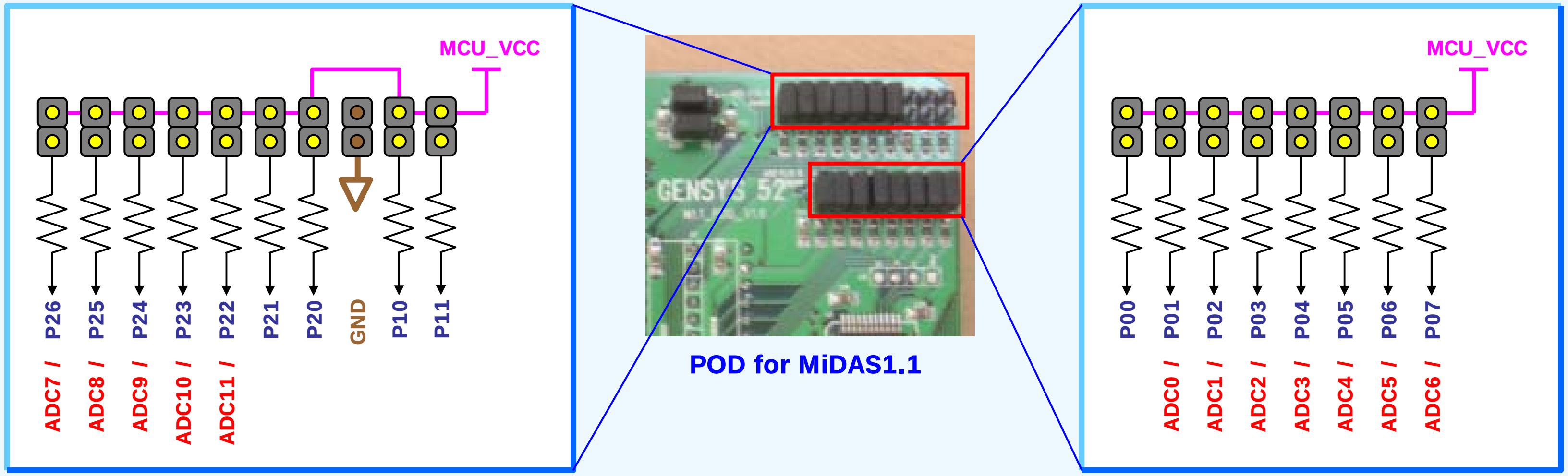


3. Jumper Setting of POD (Cont'd)

- The Pull-up R setting for P0, P1(except for P1.2), and P2 ports is supported using the additional jumper switches.

- ADC Pull-up R Jumper
  - ✓ When the IO inputs (P0.1 ~ P0.7, and P2.2 ~ P2.6) is used as ADC input channel, ADC Pull-up R jumper is opened.

[Schematic for Pull-up R jumpers]



Default :

P0 and P2	: ON
P1.0 and P1.1	: OFF

- **When you change the default setting, don't forget updating the SFR register (P0SEL, P1SEL and P2SEL) together.**

PART V : GenICE52 System for MiDAS2.0 &2.1

1. Overview : GenICE52 System for MiDAS2.0 & 2.1
2. Configuration

1. Overview : GenICE52 for MiDAS2.0 & 2.1

- Host Interface
  - ✓ Serial Port : COM1 ~ COM4
- PC Breakpoint : 8EA
- H/W Breakpoint : 1EA Iram,1EA Xram
- Voltage : 3.3V
- Various Emulation Configurations
  - ✓ ROM : Target (After Download)
  - ✓ RAM : Target
  - ✓ Clock :
    - User mode : Target
    - MDS mode : Internal RING Osc.
  - ✓ Reset : Target
- Download
  - ✓ EJTAG Cable

2. Configuration

◆ Configuration for MDS Environment

[**GENTOS** Program on PC Host]

Cable Assembly

[**GenICE52** MDS for MiDAS2.0 & 2.1]

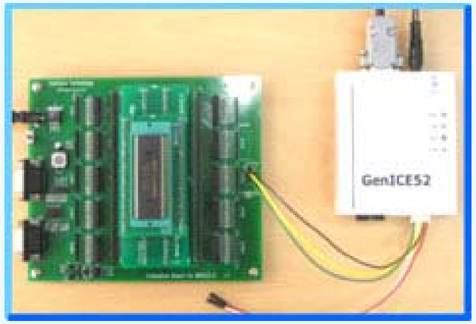
Connect the Download Cable to the EJTAG Pins of Target System

[User's Target System]

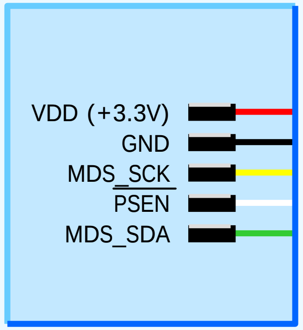
◆ Accessories

Cable
(1.5 meter)

Power Adaptor
(SMPS, 9V, 3Ah)



[Under the test]



[EJTAG Pins of GenICE52]

* Please, supply the power to the target system.

PART VI : SUPPORTS

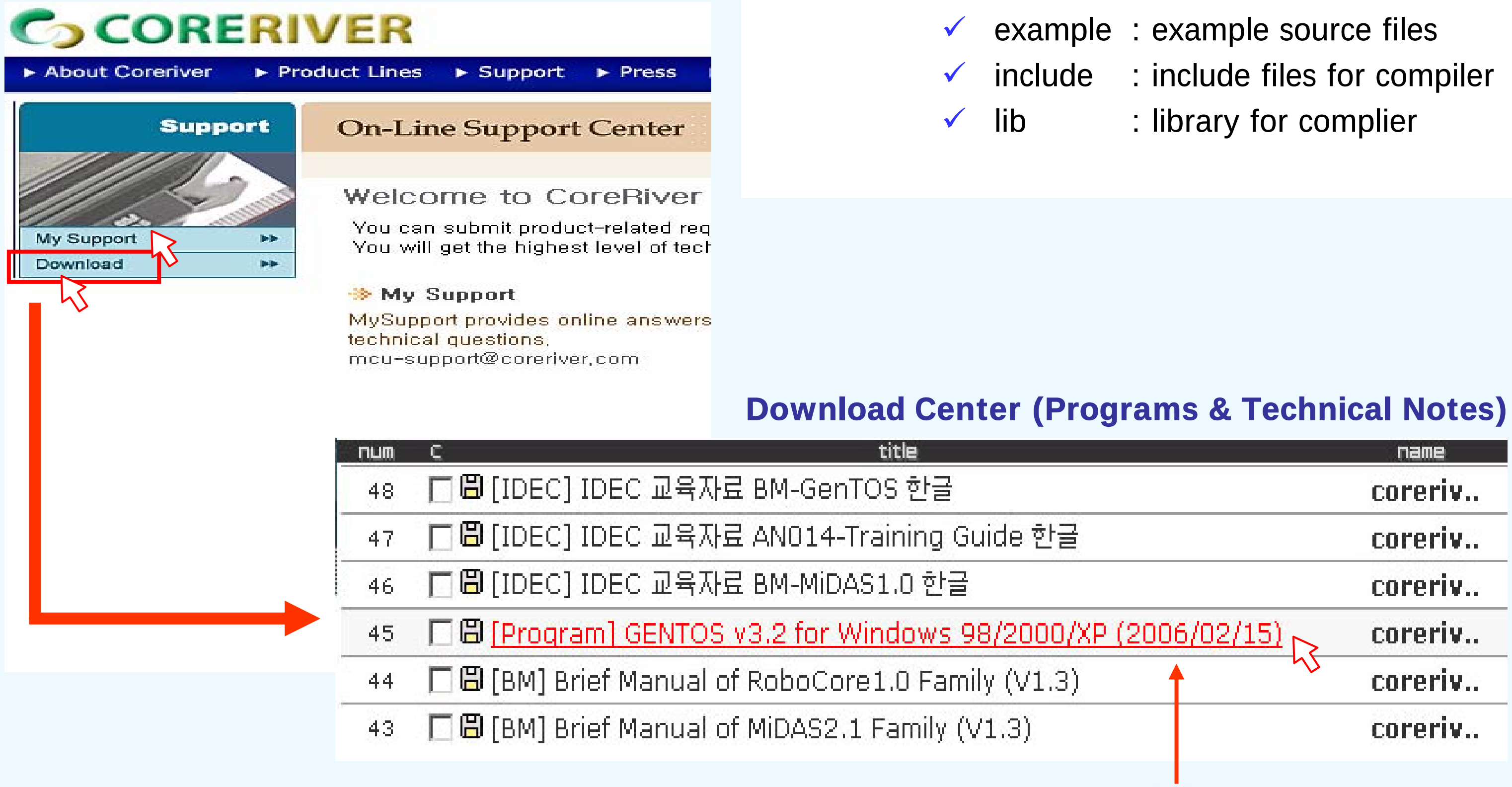
1. How to Download the Documents and Program
2. Technical Support

1. How to Download the Documents & Program

- You can check the new version of documents and programs for GENTOS.

- Download the GENTOS program from “Download ”.
- GENTOS must be installed at “C:₩GENTOS”. (Mandatory)
  - ✓ bin : executable files
  - ✓ example : example source files
  - ✓ include : include files for compiler
  - ✓ lib : library for complier

Homepage : www.coreriver.com



Welcome to CoreRiver

You can submit product-related req
You will get the highest level of tech

My Support

MySupport provides online answers
technical questions,
mcu-support@coreriver.com

Download Center (Programs & Technical Notes)

num	C	title	name
48	□	[IDEC] IDEC 교육자료 BM-GenTOS 한글	coreriv..
47	□	[IDEC] IDEC 교육자료 AN014-Training Guide 한글	coreriv..
46	□	[IDEC] IDEC 교육자료 BM-MiDAS1.0 한글	coreriv..
45	□	[Program] GENTOS v3.2 for Windows 98/2000/XP (2006/02/15)	coreriv..
44	□	[BM] Brief Manual of RoboCore1.0 Family (V1.3)	coreriv..
43	□	[BM] Brief Manual of MiDAS2.1 Family (V1.3)	coreriv..

Download the GUI Program : GENTOS.

2. Technical Support

- URL
  - ✓ www.coreriver.com
- E-mail
  - ✓ mcu-support@coreriver.com
- Reporting Bugs
  - ✓ Send an email to report bugs.
  - ✓ Bugs will be fixed ASAP.
  - ✓ When reporting a bug, it is very useful to include a small test program which reproduces the problem.
  - ✓ If you can isolate the problem by looking at the general assembly code, this can be very helpful.
  - ✓ Compiling your program with the –dumpall option can sometimes be useful in locating optimization problems.

Appendix

A. Software Developing Kit in GENTOS

B. Embedded C Compiler :Gencc (contain itself linker)

C. Processor &Preprocessor Options

D. Linker Options

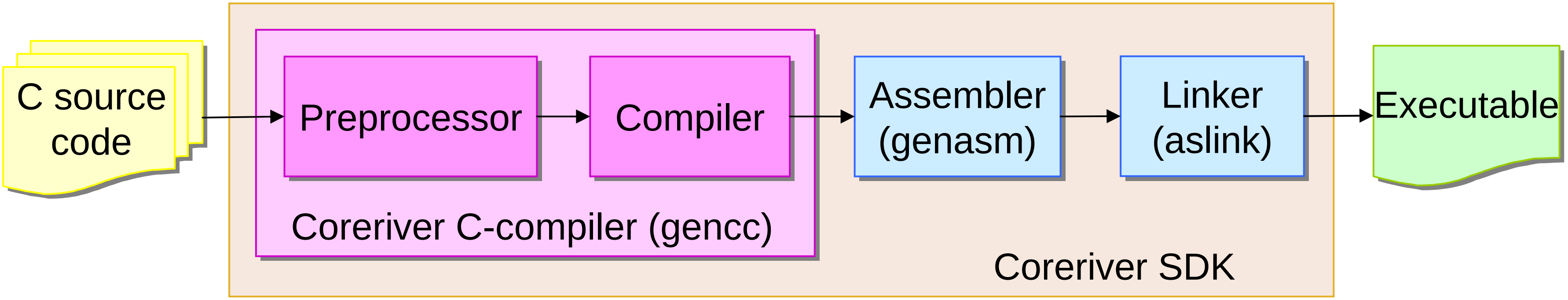
E. Intel 8051 Options

F. Optimization Options

G. Other Options

H. ASX8051 Options (not Gencc ,assembler Option)

A. S/W Developing Kit in GENTOS



main.c **sub.c** → preprocessing → **main.i** **sub.i** → compiling → **main.asm** **sub.asm** → assembling → **main.lst** **sub.lst** → linking → **main.hex**

```
#include <stdio.h>
char i;
#define MAX 10
/* comment line */
void main() {
  sfr p0 = 0x80;
  for (i=0; i<MAX;
i++)
      p0 = i;
  sub();
}
```

```
void sub() {
  char j;
  j = p0;
}
```

```
"stdio.h" is copied
here
char i;
void main() {
  sfr p0 = 0x80;
  for (i=0; i<10 ;
i++)
      p0 = i;
  sub();
}
```

```
void sub() {
  char j;
  j = p0;
}
```

```
EQU P0 80h
ORG 0000h
  MOV A, #00h
Loop :
  MOV P0, A
  INC A
  CJNE A, #0Ah, Loop
  LCALL sub
  END
```

```
ORG 0000h
sub :
  MOV R0, P0
  RET
```

```
Addr  Inst
0000 7400
0002 7580
0004 04
0005 B40AFA
0008 120000
LCALL sub
```

```
Addr  Inst
0000 A880
0002 22
```

```
Addr  Inst
0000 7400
0002 7580
0004 04
0005 B40AFA
0008 120100
……
0100 A880
0102 22
```

B. Embedded C Compiler: Gencc

- MCU Specific Optimization
  - ✓ Sub-expression elimination
  - ✓ Dead code elimination
  - ✓ Copy propagation
  - ✓ Loop optimizations (loop invariant and strength reduction)
  - ✓ Constant folding and propagation
  - ✓ Global register allocation

- Supported data types
  - ✓ `short` (16 bits, 2 bytes),
  - ✓ `char` (8 bits, 1 byte),
  - ✓ `int` (16 bits, 2 bytes),
  - ✓ `long` (32 bits, 4 bytes), and
  - ✓ `float` (4 bytes IEEE)

- Support the inline assembler code

C. Processor & Preprocessor Options (1 of 3)

- Options

-Processor Option

✓ -mmcs51

Generate code for the Intel MCS51 family of processors. This is the default processor target.

-Preprocessor Option

✓ -I<path>

The additional location where the preprocessor will look for <..h> or "..h" files.

✓ -D<macro[=value]>

Command line definition of macros. Passed to the preprocessor.

✓ -M

Tell the preprocessor to output a rule suitable for make describing the dependencies of each object file. For each source file, the preprocessor outputs one make-rule whose target is the object file name for that source file and whose dependencies are all the files `#include'd in it

C. Processor & Preprocessor Options (2 of 3)

- Options (cont'd)

  ✓ -C

  Tell the preprocessor not to discard comments. Used with the `-E' option.

  ✓ -MM

  Like `-M' but the output mentions only the user header files included with `#include "file"'. System header files included with `#include <file>' are omitted.

  ✓ -Aquestion(answer)

  Assert the answer answer for question, in case it is tested with a preprocessor conditional such as `#if #question(answer)'. `-A-‘ disables the standard assertions that normally describe the target machine.

  ✓ -Umacro Undefine macro macro.

  `-U' options are evaluated after all `-D' options, but before any `-include' and `-imacros' options.

C. Processor & Preprocessor Options (3 of 3)

- Options (cont’d)
  - ✓ -dM

    Tell the preprocessor to output only a list of the macro definitions that are in effect at the end of preprocessing. Used with the `-E' option.
  - ✓ -dD

    Tell the preprocessor to pass all macro definitions into the output, in their proper sequence in the rest of the output.
  - ✓ -dN

    Like `-dD' except that the macro arguments and contents are omitted. Only `#define name' is included in the output.
  - ✓ -Wp

    preprocessor Option [preprocessor Option]... Pass the preprocessor Option to the preprocessor sdcpp.

D. Linker Options (1 of 3)

◆ Options

- ✓ -L --lib-path <absolute path to additional libraries>

  This option is passed to the linkage editor's additional libraries search path. The path name must be absolute.

- ✓ --xram-loc <Value>

  The start location of the external ram, default value is 0. The value entered can be in Hexadecimal or Decimal format, e.g.: --xram-loc 0x8000 or --xram-loc 32768.

  set start address

- ✓ --code-loc <Value>

  The start location of the code segment, default value 0. Note when this option is used the interrupt vector table is also relocated to the given address. The value entered can be in Hexadecimal or Decimal format, e.g.: --code-loc 0x8000 or --code-loc 32768.

- ✓ --idata-loc <Value>

  The start location of the indirectly addressable internal ram of the 8051, default value is 0x80. The value entered can be in Hexadecimal or Decimal format, eg. --idata-loc 0x88 or --idata-loc 136.

D. Linker Options (2 of 3)

- Options (cont'd)
  - ✓ --stack-loc <Value>

    By default the stack is placed after the data segment. Using this option the stack can be placed anywhere in the internal memory space of the 8051. The value entered can be in Hexadecimal or Decimal format, e.g. --stack-loc 0x20 or --stack-loc32.

  - ✓ --data-loc <Value>

    The start location of the internal ram data segment.

    The value entered can be in Hexadecimal or Decimal format, eg. --data-loc 0x20 or --data-loc 32. (By default, the start location of the internal ram data segment is set as low as possible in memory, taking into account the used register banks and the bit segment at address 0x20. For example if register banks 0 and 1 are used without bit variables, the data segment will be set, if --data-loc is not used, to location 0x10.)

  - ✓ --bit-loc <Value>

    The start location of the bit addressable internal ram of the 8051.

    This is not implemented yet. Instead an option can be passed directly to the linker: -Wl -bBSEG=<Value>.

D. Linker Options (3 of 3)

- Options (cont'd)
  - ✓ --out-fmt-ihx

    The linker output (final object code) is in Intel Hex format. This is the default option. The format itself is documented in the documentation of srecord.

  - ✓ --out-fmt-ihx

    The linker output (final object code) is in Intel Hex format. This is the default option. The format itself is documented in the documentation of srecord.

  - ✓ --out-fmt-s19

    The linker output (final object code) is in Motorola S19 format. The format itself is documented in the documentation of srecord.

  - ✓ -Wl linkOption[,linkOption]...

    Pass the linkOption to the linker.

E. Intel 8051 Options (1 of 2)

- Usage: gencc `[options]`
- Options
  - ✓ --model-

    small Generate code for Small Model programs, see section Memory Models for more details. This is the default model.
  - ✓ --model-

    large Generate code for Large model programs, see section Memory Models for more details. If this option is used all source files in the project have to be compiled with this option.
  - ✓ --xstack

    Uses a pseudo stack in the first 256 bytes in the external ram for allocating variables and passing parameters. See section [sub:External-Stack] External Stack for more details.
  - ✓ --iram-

    size <Value> Causes the linker to check if the internal ram usage is within limits of the given value.
  - ✓ --xram-

    size <Value> Causes the linker to check if the external ram usage is within limits of the given value.

E. Intel 8051 Options (2 of 2)

- Options (cont'd)
  - ✓ --code-size <Value>
    Causes the linker to check if the code memory usage is within limits of the given value.
  - ✓ --stack-size <Value>
    Causes the linker to check if there is at minimum <Value> bytes for stack.
  - ✓ --pack-iram
    Causes the linker to use unused register banks for data variables and pack data, idata and stack together. This is the default now.
  - ✓ --no-pack-iram
    Causes the linker to use old style for allocating memory areas.

F. Optimization Options (1 of 4)

- Options (cont'd)
  - ✓ --nogcse

    Will not do global subexpression elimination, this option may be used when the compiler creates undesirably large stack/data spaces to store compiler temporaries.

    A warning message will be generated when this happens and the compiler will indicate the number of extra bytes it allocated.

  - ✓ --noinvariant

    Will not do loop invariant optimizations, this may be turned off for reasons explained for the previous option.

  - ✓ --noinduction

    Will not do loop induction optimizations, see section strength reduction for more details.

    It is recommended that this option is NOT used, #pragma noinduction can be used to turn off induction optimizations for a given function only.

F. Optimization Options (2 of 4)

- Options (cont'd)
  - ✓ --nojtbound
    Will not generate boundary condition check when switch statements are implemented using jump-tables. See section [sub:'switch'-Statements] Switch Statements for more details.
  - ✓ --noloopreverse
    Will not do loop reversal optimization.
  - ✓ --nolabelopt
    Will not optimize labels (makes the dumpfiles more readable).
  - ✓ --no-xinit-opt
    Will not memcpy initialized data from code space into xdata space. This saves a few bytes in code space if you don't have initialized data.

F. Optimization Options (3 of 4)

- Options (cont’d)
  - ✓ -nooverlay
    The compiler will not overlay parameters and local variables of any function, see section Parameters and local variables for more details.
  - ✓ --no-peep
    Disable peep-hole optimization.
  - ✓ --peep-file <filename>
    This option can be used to use additional rules to be used by the peep hole optimizer. See section [sub:Peephole-Optimizer] Peep Hole optimizations for details on how to write these rules.
  - ✓ --peep-asm
    Pass the inline assembler code through the peep hole optimizer. This can cause unexpected changes to inline assembler code, please go through the peephole optimizer rules defined in the source file tree '<target>/peeph.def' before using this option.

F. Optimization Options (4 of 4)

- Options (cont'd)
  - ✓ --opt-code-speed

    The compiler will optimize code generation towards fast code, possibly at the expense of code size.

  - ✓ --opt-code-size

    The compiler will optimize code generation towards compact code, possibly at the expense of code speed.

G. Other Options (1 of 7)

- Options
  - ✓ -c --compile-

    only will compile and assemble the source,but will not call the linkage editor.

  - ✓ --c1

    mode reads the preprocessed source from standard input and compiles it. The file name for the assembler output must be specified using the -o option.

  - ✓ -E

    Run only the C preprocessor. Preprocess all the C source files specified and output the results to standard output.

  - ✓ -o <path/file>

    The output path resp. file where everything will be placed. If the parameter is a path, it must have a trailing slash (or backslash for the Windows binaries) to be recognized as a path.

  - ✓ --stack-auto

    All functions in the source file will be compiled as reentrant, i.e. the parameters and local variables will be allocated on the stack. See section [sec:Parameters-and-Local-Variables] Parameters and Local Variables for more details.

G. Other Options (2 of 7)

- Options (cont'd)
  - ✓ --callee-saves function1[,function2][,function3]....

    The compiler by default uses a caller saves convention for register saving across function calls, however this can cause unnecessary register pushing & popping when calling small functions from larger functions. This option can be used to switch the register saving convention for the function names specified.

  - ✓ --debug

    When this option is used the compiler will generate debug information. The debug information collected in a file with .cdb extension can be used with the SDCDB.

  - ✓ --int-long-reent

    Integer (16 bit) and long (32 bit) libraries have been compiled as reentrant. Note by default these libraries are compiled as non-reentrant.See section Installation for more details.

  - ✓ -S

    Stop after the stage of compilation proper; do not assemble. The output is an assembler code file for the input file specified.

G. Other Options (3 of 7)

◆ Options (cont'd)

- ✓ --cyclomatic

  This option will cause the compiler to generate an information message for each function in the source file. The message contains some important information about the function.

- ✓ --float-reent

  Floating point library is compiled as reentrant. See section Installation for more details.

- ✓ --main-return

  This option can be used if the code generated is called by a monitor program or if the main routine includes an endless loop. This option might result in slightly smaller code and save two bytes of stack space. The return from the 'main' function will return to the function calling main.

- ✓ --nostdinc

  This will prevent the compiler from passing on the default include path to the preprocessor.

- ✓ --nostdlib

  This will prevent the compiler from passing on the default library path to the linker.

G. Other Options (4 of 7)

◆ Options (cont’d)

- ✓ --verbose

  Shows the various actions the compiler is performing.

- ✓ -V

  Shows the actual commands the compiler is executing.

- ✓ --no-c-code-in-asm

  Hides your ugly and inefficient c-code from the asm file, so you can always blame the compiler.

- ✓ --no-peep-

  comments Will not include peep-hole comments in the generated files.

- ✓ --i-code-in-asm

  Include i-codes in the asm file. Sounds like noise but is most helpful for debugging the compiler itself.

G. Other Options (5 of 7)

- Options (cont'd)
  - ✓ --less-pedantic

    Disable some of the more pedantic warnings (jwk burps: please be more specific here, please!). If you want rather more than less warnings you should consider using a separate tool dedicated to syntax checking like splint [www.splint.org].
  - ✓ --disable-warning <nnnn>

    Disable specific warning with number <nnnn>.
  - ✓ --print-search-dirs

    Display the directories in the compiler's search path
  - ✓ --use-stdout

    Send errors and warnings to stdout instead of stderr.
  - ✓ -Wa asmOption[,asmOption]...

    Pass the asmOption to the assembler

G. Other Options (6 of 7)

- Options (cont'd)
  - ✓ Intermediate Dump Options<sub:Intermediate-Dump-Options>

    The following options are provided for the purpose of retargetting and debugging the compiler.

  - ✓ --dumpraw

    This option will cause the compiler to dump the intermediate code into a file of named <source filename>.dumpraw just after the intermediate code has been generated for a function, i.e. before any optimizations are done.

  - ✓ --dumpgcse

    Will create a dump of iCode's, after global subexpression elimination, into a file named <source filename>.dumpgcse.

  - ✓ --dumpdeadcode

    Will create a dump of iCode's, after deadcode elimination, into a file named <sourcefilename> . dumpdeadcode.

G. Other Options (7 of 7)

◆ Options (cont'd)

✓ --dumploop

Will create a dump of iCode's, after loop optimizations, into a file named <source filename>. dumploop.

✓ --dumprange

Will create a dump of iCode's, after live range analysis, into a file named <source filename>. dumprange.

✓ --dumlrange

Will dump the life ranges for all symbols.

✓ --dumpregassign

Will create a dump of iCode's, after register assignment, into a file named <source filename>.dumprassgn.

✓ --dumplrange

Will create a dump of the live ranges of iTemp's.

✓ --dumpall

Will cause all the above mentioned dumps to be created.

H. ASX8051 Options (not gencc, assembler Option)

- Options
  - ✓ d : decimal listing
  - ✓ q : octal listing
  - ✓ x : hex listing (default). The listing radix affects the .lst, .rel, and .sym files.
  - ✓ G : undefined symbols made global
  - ✓ a : all user symbols made global
  - ✓ l : create list output file1.lst
  - ✓ o : create object output file1.rel
  - ✓ s : create symbol output file1.sym
  - ✓ p : disable listing pagination
  - ✓ w : wide listing format for symbol table
  - ✓ z : enable case sensitivity for symbols. relocatable reference flagging
  - ✓ f : by ` in the listing file
  - ✓ ff : by mode in the listing file
  - ✓ j : generate a line number symbol