

SKP16C26

Tutorial 1

Software Development Process using HEW

Overview

The following tutorial is a brief introduction on how to develop and debug programs using HEW (High-performance Embedded Workshop), KD30, and other software and hardware tools included with the SKP16C26.

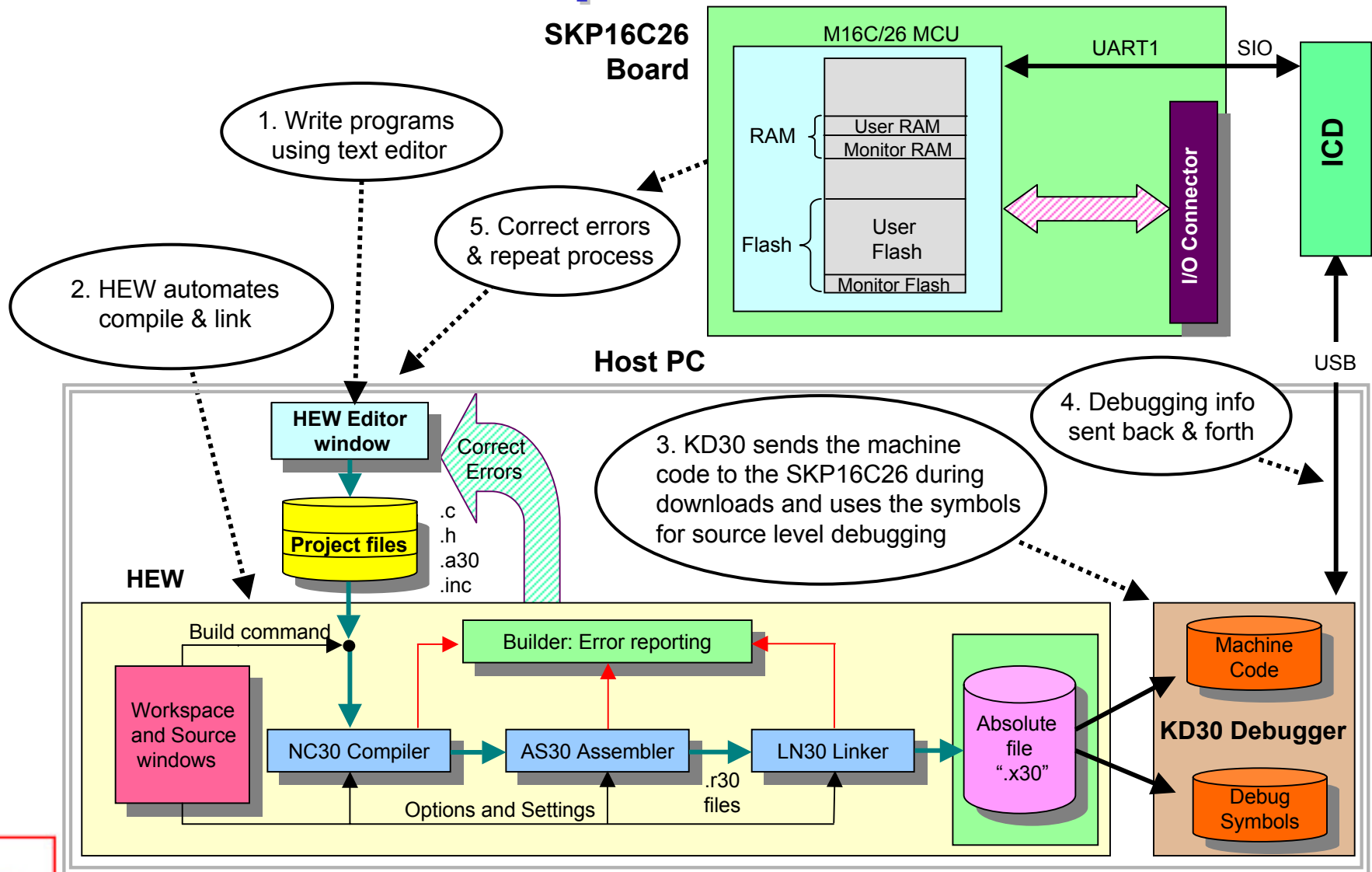
To get the most out of the Starter Kit, check out the references at the end of this tutorial.

Note: *This tutorial assumes the user has done the following:*

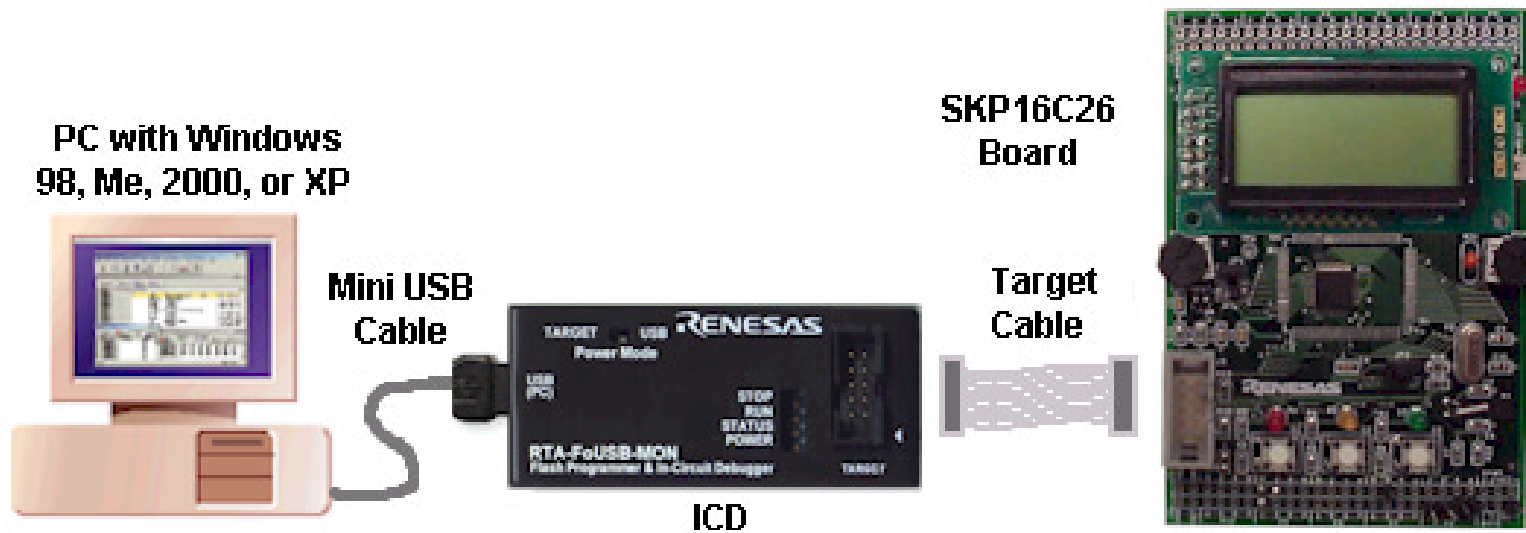
- 1.** *Followed the 'Quickstart Guide'*
- 2.** *Installed the SKP files, examples, and software tools in the default directories.*



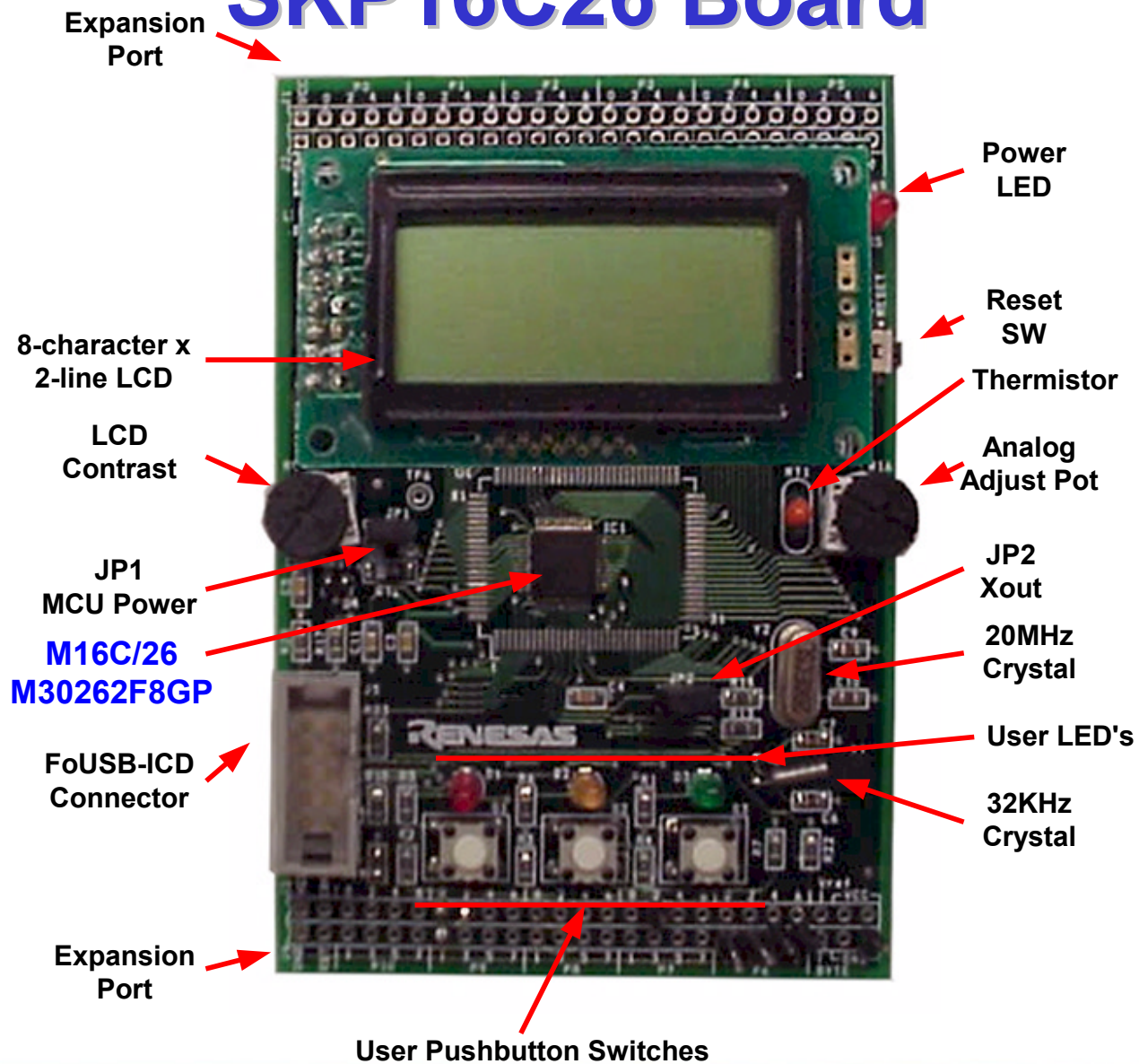
The Development Process



SKP16C26 Connectivity



SKP16C26 Board



SKP16C26 Board Features

M16C/26 (M30262F8GP) MCU

- 20MHz Operating Frequency at 3.0V – 5V, 10MHz Operating Frequency at 2.7V – 5V
- 64kB Flash ROM, 2kB x 2 Virtual EEPROM, and 2kB RAM
- 38 GPIO and 4 Key-on Wakeup Inputs
- 8 Timers plus a Watchdog Timer
- 8-channel 10-bit ADC
- 2 DMAC
- 3 SIO's (supports I²C and SPI)
- Voltage and Oscillation Failure Detection
- Clock sources: Main (Xin), Sub (Xcin), Internal R/C (ring)

Onboard Features

- LED's (3 User, 1 Power)
- Removable 2-line x 8-character LCD
- Pushbutton Switches (3 User, 1 Reset)
- Thermistor and potentiometer on two A/D inputs
- I/O available on Expansion Ports

ICD (RTA-FoUSB-MON)

The ICD (In-Circuit Debugger) provides power and a USB interface to the Host PC and communicates commands and data to and from the SKP16C26 board via a synchronous serial interface.

As a debugging tool (during program debug), the ICD + KD30 downloads a small kernel (or ROM Monitor) program with the user program to the SKP16C26 Board . This kernel provides a communication interface between the M16C/26 MCU and the ICD + KD30 Debugger application on MCU status. While the kernel uses some resources of the M16C/26, the operation of the ICD is transparent to the user's program.

As a programming tool, the ICD + Flash-over-USB™(FoUSB) Programmer can be used to download user programs to the M16C/26 MCU on the SKP16C26 Board and many other Renesas' flash MCU's (the ICD will support other Renesas flash MCU's by downloading an MCU Monitor Image (MMI) file for a particular MCU thru KD30 or FoUSB Programmer).

NOTE: The kernel is only downloaded with the user program when using KD30 Debugger but NOT the FoUSB Programmer.



Development Tools

HEW

An Integrated Development Environment (IDE) that invokes all necessary software for building your project

KD30

PC software that communicates with the ROM Monitor Program (in flash on the MCU) for program debug

NC30 Entry Version

C-compiler (limited version of NC30). Conforms to ANSI C standards (see release notes on limitations)

AS30

Relocatable Assembler

Supports structured language and a wide variety of macro instructions

Flash-over-USB™ Programmer

Flash programmer for Renesas Flash MCU's.

HEW Overview

HEW is an acronym for **H**igh-performance **E**Embedded **W**orkshop.

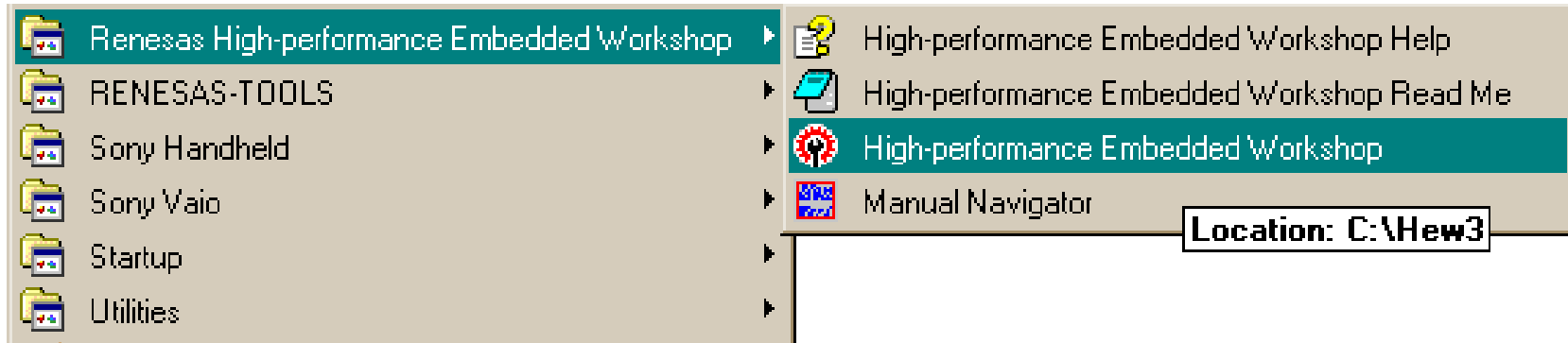
When writing a microcontroller (or any computer) program, the program is usually split into multiple files to make it easier to read and understand.

While exactly how the files are organized is up to the programmer, typically, the code is split up in a logical manner into various files (e.g. math functions in one file, serial port drivers in another, etc).

After all the files in a **project** are compiled and assembled, a **linker** combines all the files into a single file. These steps can be tedious and repetitive. To make the process simple, we use an **Integrated Development Environment (IDE)** called **HEW**.

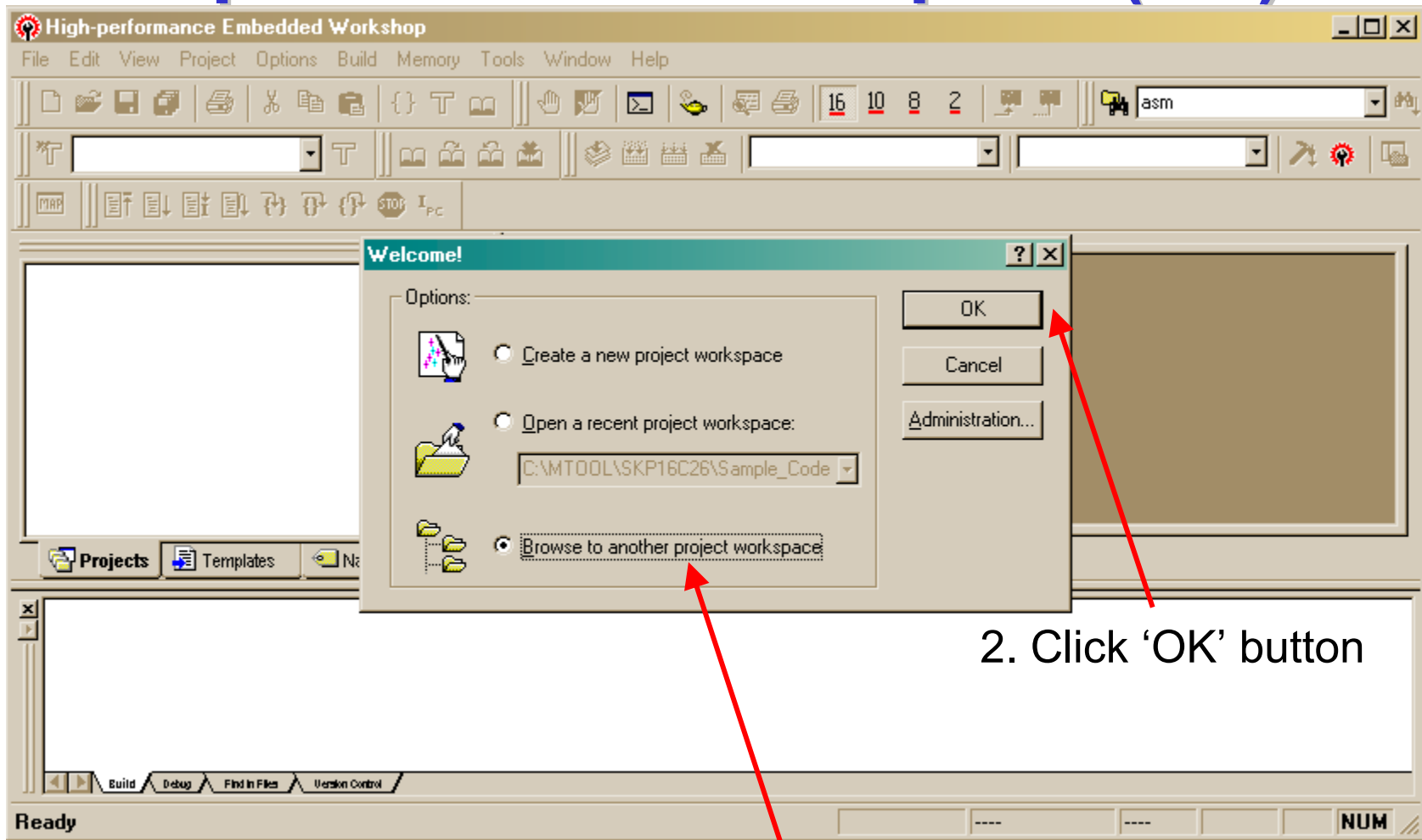


Start HEW



From the Windows Start menu, click on
**Programs > Renesas High-performance Embedded Workshop>
High-performance Embedded Workshop**

Open a HEW Workspace (1/3)

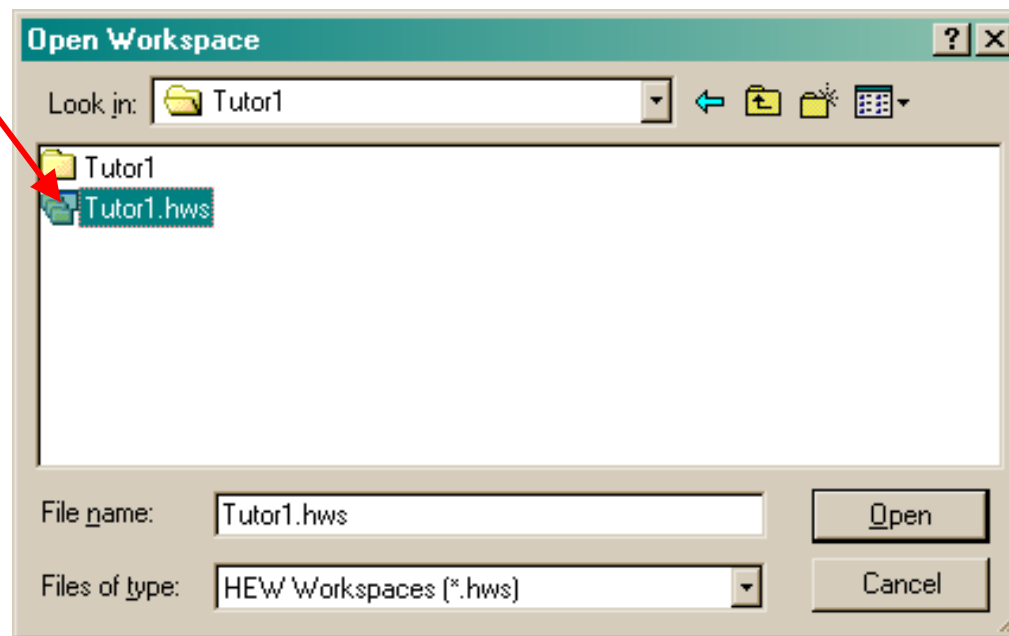


2. Click 'OK' button

1. After HEW opens, from the Welcome dialog box, select 'Browse to another project workspace' option, then click OK.

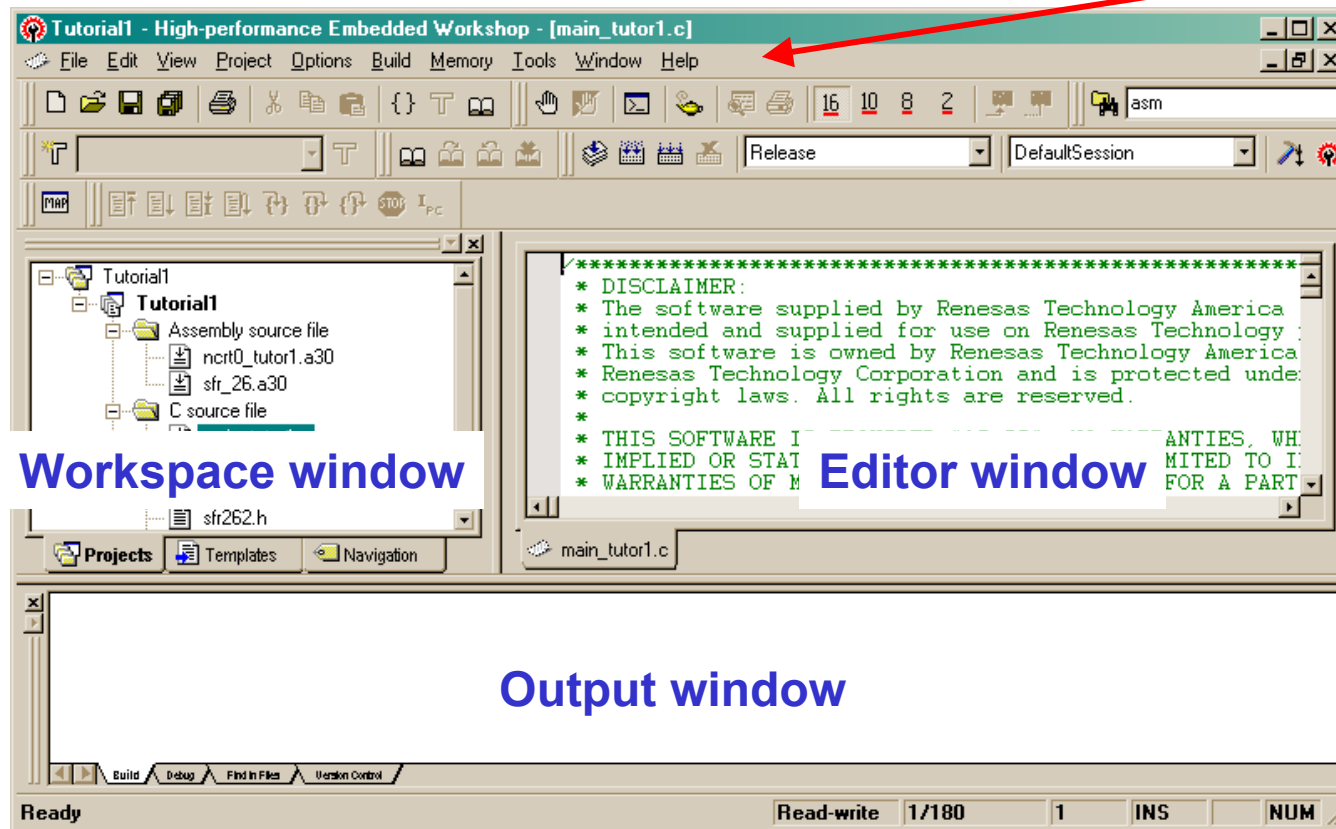
Open a HEW Workspace (2/3)

Using the Open Workspace dialog box, browse until you get to 'C:\MTOOL\SKP16C26\Sample_Code\Tutor1' folder. Click on Tutor1.hws HEW workspace file and then click on 'Open' button.



Open a HEW Workspace (3/3)

HEW should look like the figure below.



Menu bar

Toolbars

Workspace window

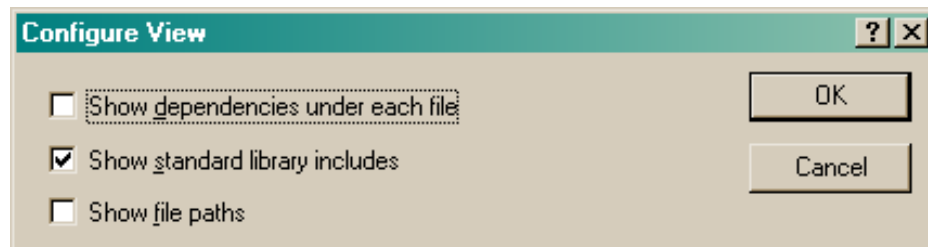
Editor window

Output window

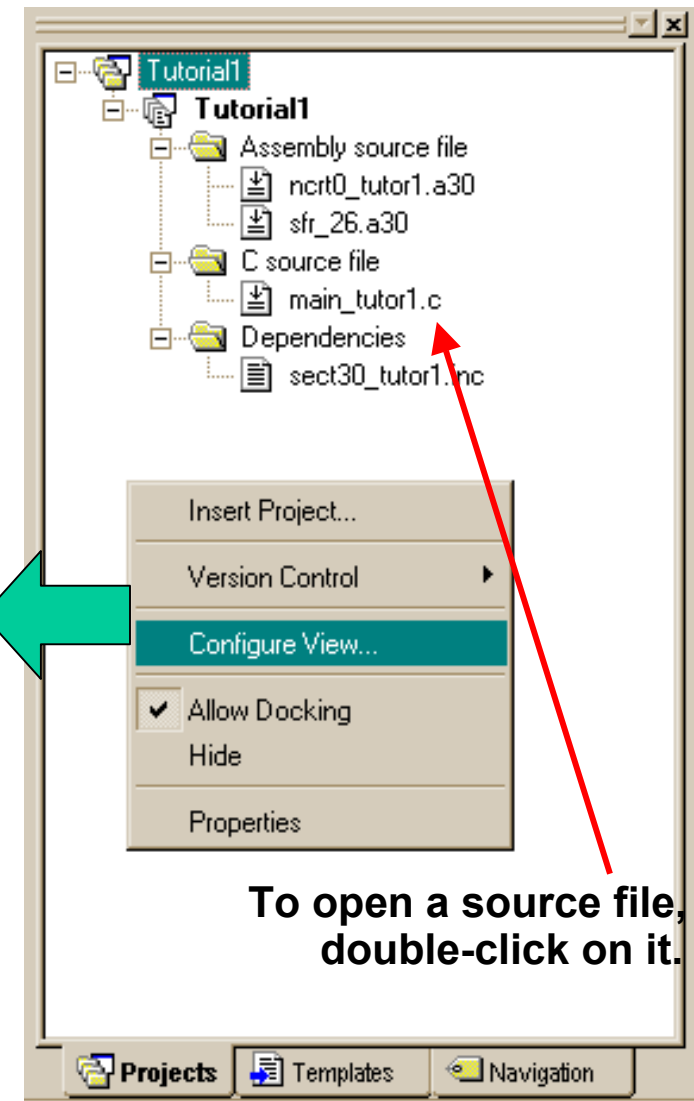
Workspace Window

In the Projects tab, source files and header files are displayed.

To change how dependencies are displayed, e.g. show dependencies for each source file, right-click within the window, and select Configure View.



Try the following, click on 'Show dependencies under each file' and see what happens to files displayed on the window.



Editor (Source) Window

```
*****  
* DISCLAIMER: *  
* The software supplied by Renesas Technology America Inc. is *  
* intended and supplied for use on Renesas Technology products. *  
* This software is owned by Renesas Technology America, Inc. or *  
* Renesas Technology Corporation and is protected under applicable *  
* copyright laws. All rights are reserved. *  
* *  
* THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, *  
* IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO IMPLIED *  
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE *  
* APPLY TO THIS SOFTWARE. RENESAS TECHNOLOGY AMERICA, INC. AND *  
* AND RENESAS TECHNOLOGY CORPORATION RESERVE THE RIGHT, WITHOUT *  
* NOTICE, TO MAKE CHANGES TO THIS SOFTWARE. NEITHER RENESAS *  
* TECHNOLOGY AMERICA, INC. NOR RENESAS TECHNOLOGY CORPORATION SHALL, *  
* IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR *  
* CONSEQUENTIAL DAMAGES FOR ANY REASON WHATSOEVER ARISING OUT OF THE *  
* USE OR APPLICATION OF THIS SOFTWARE. *  
*****/  
  
*****  
* *  
* DESCRIPTION: main routine for tutor1 *  
* *  
* AUTHOR: System LSI BU, Applications Engineering *  
* *  
* PURPOSE: This program blinks the three LEDs (D1, D2, & D3) *  
* sequentially. The blink rate is controlled by the *  
* Analog Adjust potentiometer connected to AN0 of the *  
* M16C/26 ADC. *  
* *  
*****
```

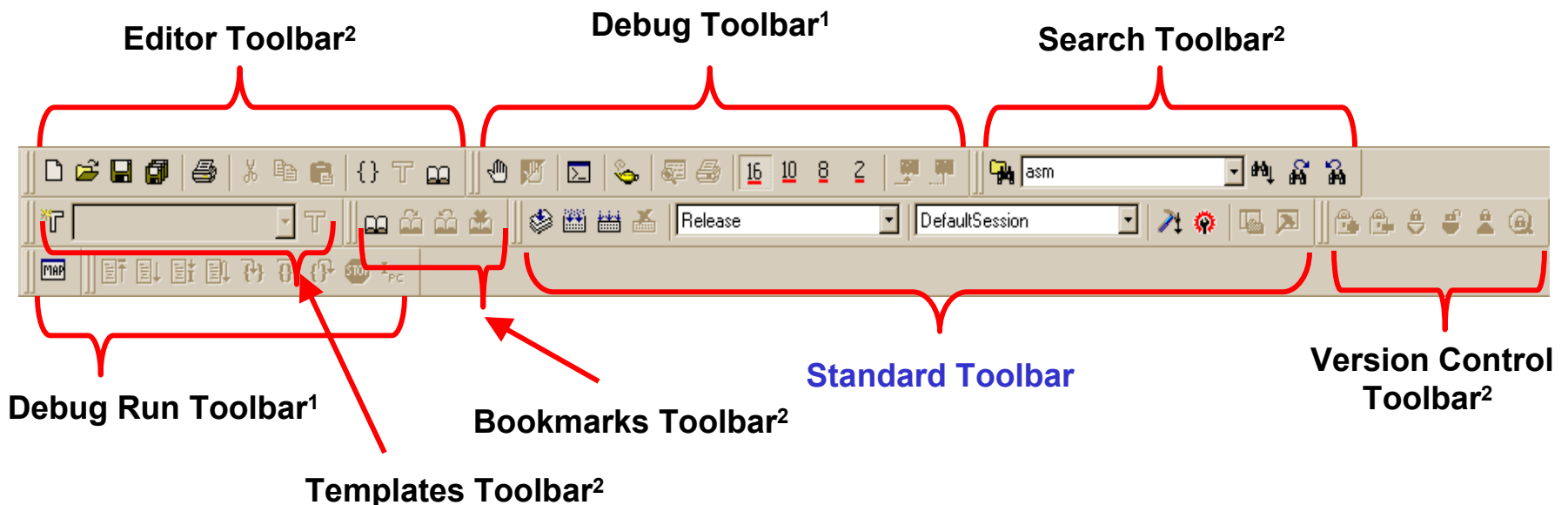
Any opened source file within the workspace are shown on the Editor window.

Line, total no. of lines, and column numbers are displayed here



HEW Toolbars

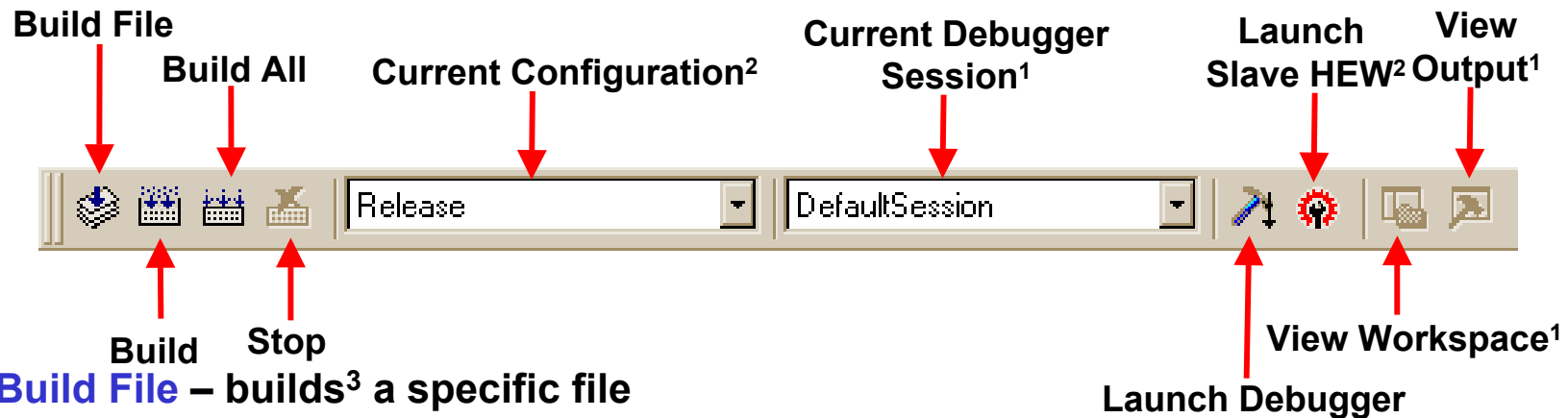
HEW is a powerful development environment with a lot of features and functionality. For this tutorial, the focus will be on features (i.e. Standard Toolbar) that will help you understand the M16C development process using HEW.



Notes:

1. On HEW 3.0 R1, M16C is not supported by the Debug and Debug Run toolbars.
2. See HEW user's manual about these toolbars.

Standard Toolbar



Build File – builds³ a specific file

Build – builds files that were modified since last build

Build All – builds the whole project regardless of whether there were modifications or not

Stop – stops a running build process

Current Configuration – build configuration (e.g. for debug, optimized, etc)

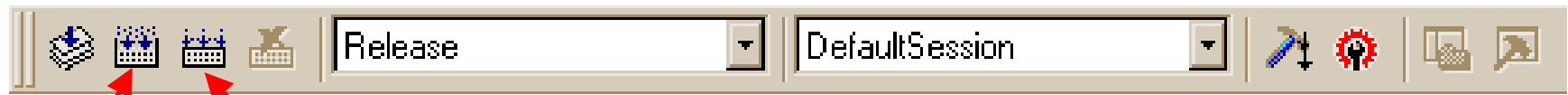
Current Debugger Session – debug session configuration

Launch Debugger – calls defined debugger

Notes:

1. *Current Debugger Session, View Workspace, & View Output are not currently supported for M16C development.*
2. *See HEW User's manual for details.*
3. *A 'build' means running certain files (e.g. source files) under some tools (e.g. compiler, linker) to produce an output file (e.g. X30 or MOT executable files for M16C)*

Build(re-build) Tutor1



Build

Build All
(re-build)

Let's rebuild the Tutor1 project into an executable module, click on the **'Build All'** icon. This will re-compile and link all the source files.

If any of the source files are modified, click on the **'Build'** icon as this will only compile these modified files, which makes generating an executable module faster.

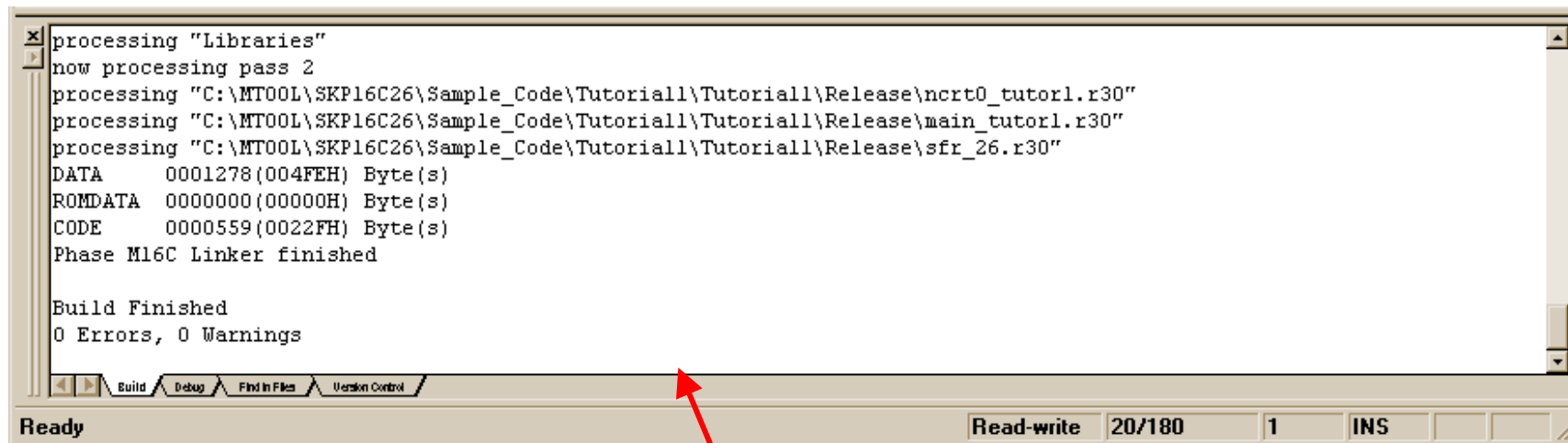
Always perform a **'Build All'** when the configuration changed.

Status, errors, messages, etc during a build process is displayed on the Output window...



Output Window

The major use of the Output window is to determine if any errors or warnings occurred, and where, during the build process.



The screenshot shows a window titled 'Output Window' with a scrollable text area. The text inside reads: 'processing "Libraries"', 'now processing pass 2', 'processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\ncrt0_tutor1.r30"', 'processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\main_tutor1.r30"', 'processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\sfr_26.r30"', 'DATA 0001278(004FEH) Byte(s)', 'ROMDATA 0000000(000000H) Byte(s)', 'CODE 0000559(0022FH) Byte(s)', 'Phase M16C Linker finished', 'Build Finished', and '0 Errors, 0 Warnings'. Below the text area is a toolbar with buttons for 'Build', 'Debug', 'Find in Files', and 'Version Control'. At the bottom of the window, the status bar shows 'Ready', 'Read-write', '20/180', '1', and 'INS'. A red arrow points to the 'Build' button.

```
processing "Libraries"
now processing pass 2
processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\ncrt0_tutor1.r30"
processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\main_tutor1.r30"
processing "C:\MT00L\SKP16C26\Sample_Code\Tutorial1\Tutorial1\Release\sfr_26.r30"
DATA 0001278(004FEH) Byte(s)
ROMDATA 0000000(000000H) Byte(s)
CODE 0000559(0022FH) Byte(s)
Phase M16C Linker finished

Build Finished
0 Errors, 0 Warnings
```

The no. of errors and warnings will show up in this window. You can then scroll up to find where the error(s) occurred. If no errors or warnings were found, 'Build Finished' will be displayed.

Now that an executable file has been created, the next step is to download and run the program on the SKP16C26 Board using the KD30 Debugger + ICD...

Do not close HEW yet. We will be returning to it later.

KD30 Debugger Overview

The KD30 Debugger can be used to verify that the program we developed works exactly as we intended and when it does not, we can also use KD30 to find out why.

Breakpoints can be set in KD30 to stop the program at certain points (of our program) so we can verify that up to that point, the program still works correctly using registers or variables in memory. The number of breakpoints will vary from MCU to MCU. **For M16C/26, the maximum no. of breakpoints with KD30 is 6.**

KD30 allows “step” execution in our program, which means program execution on a per line basis (whether in source level or machine code level).

Various windows in KD30 allow us to see register values and memory locations.



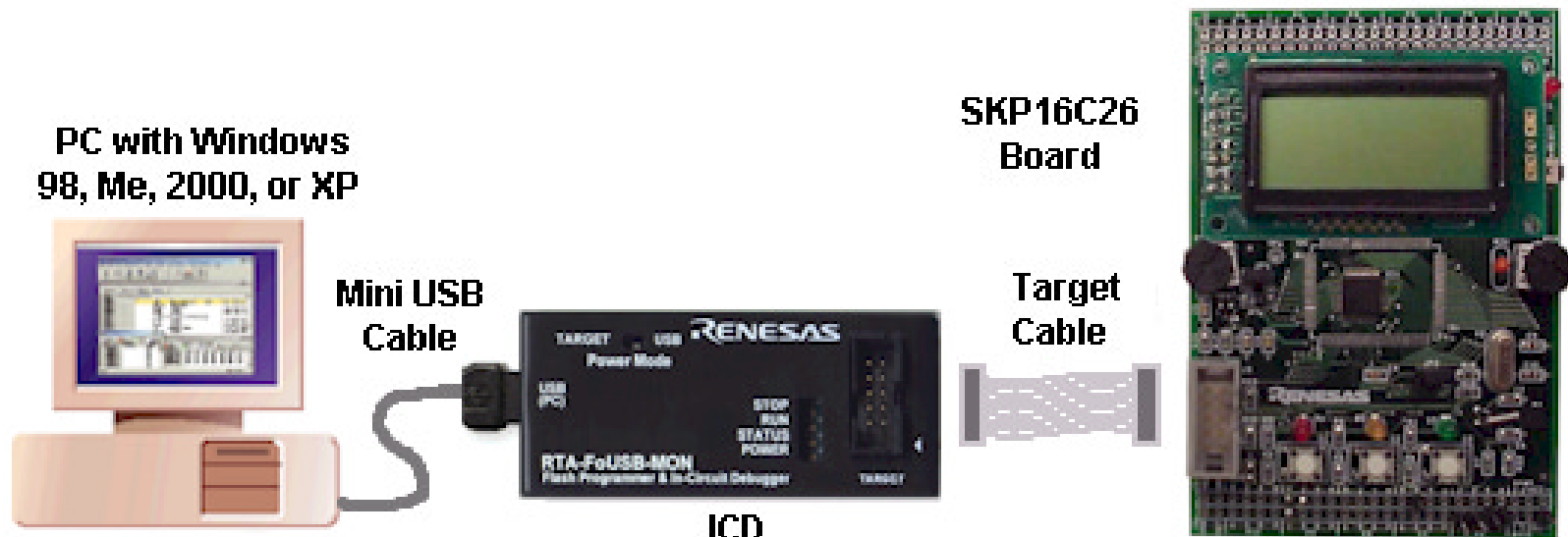
KD30 Debugger Exercise

- Download and run a program on the SKP16C26 board
- General use of the KD30 Debugger including stepping and setting breakpoints
- Return to HEW, modify the program, rebuild, and run the updated program on the SKP16C26 board



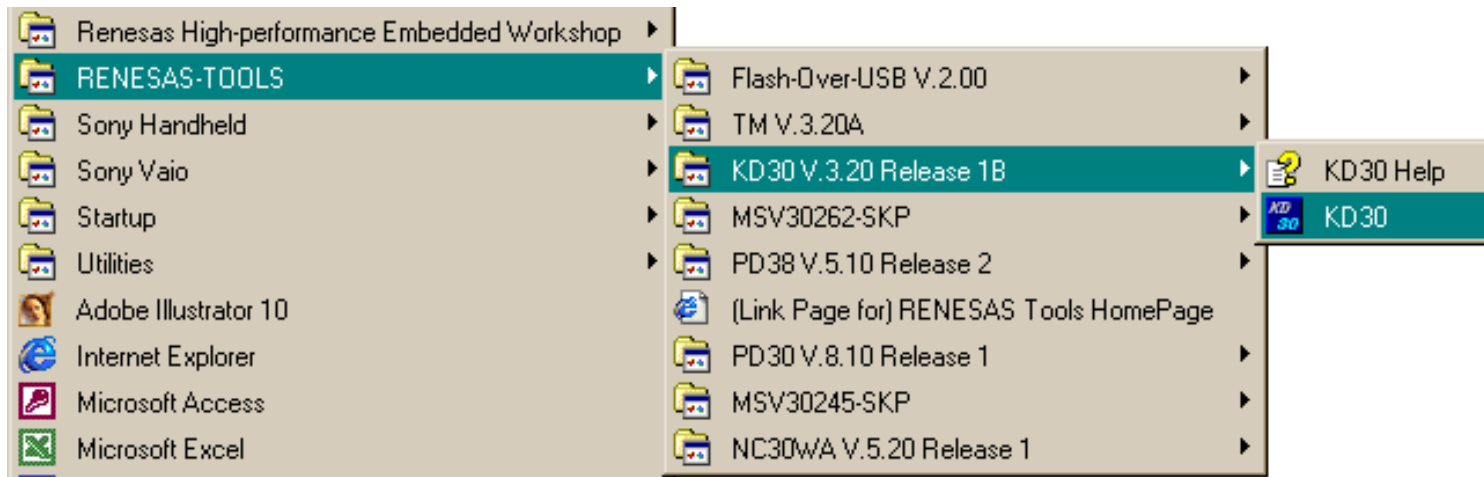
Connect Hardware

Before starting KD30, connect the ICD to the SKP16C26 Board as shown. Connect the USB cable to the PC. On the ICD, the Power LED is on and the Status (Yellow) LED is blinking once a second (this means that the ICD USB driver was loaded correctly by Windows™). If not (i.e. blinking three times a second), the Windows™ driver has not been loaded. Try disconnecting the mini USB cable, wait a few seconds, and then plug it back in. If this does not work, please check Appendix. A Troubleshooting of the SKP16C26 user's manual.



Start KD30

Launch KD30 from the Windows Start Menu,



or from HEW's Standard Toolbar¹.

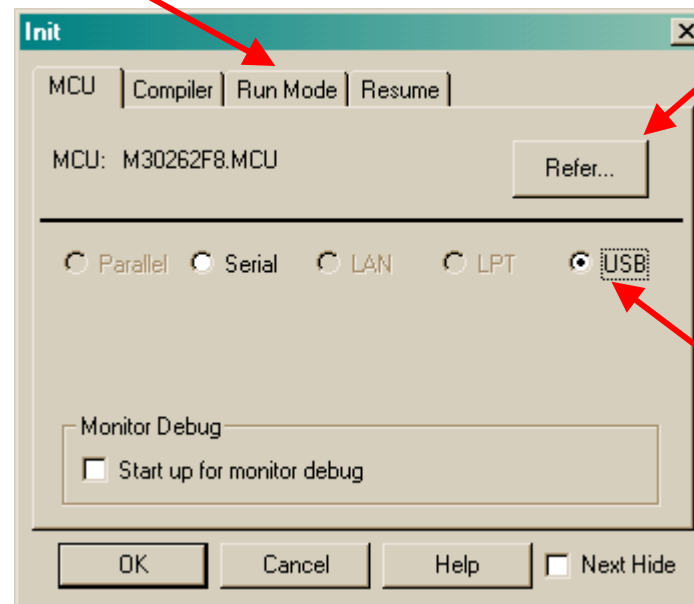


Note: 1. To call KD30 from HEW requires some configuration that is discussed in tutorial 2, Creating a New Project.

KD30 Init Window (1/2)

Step 3. Now click the 'Run Mode' tab

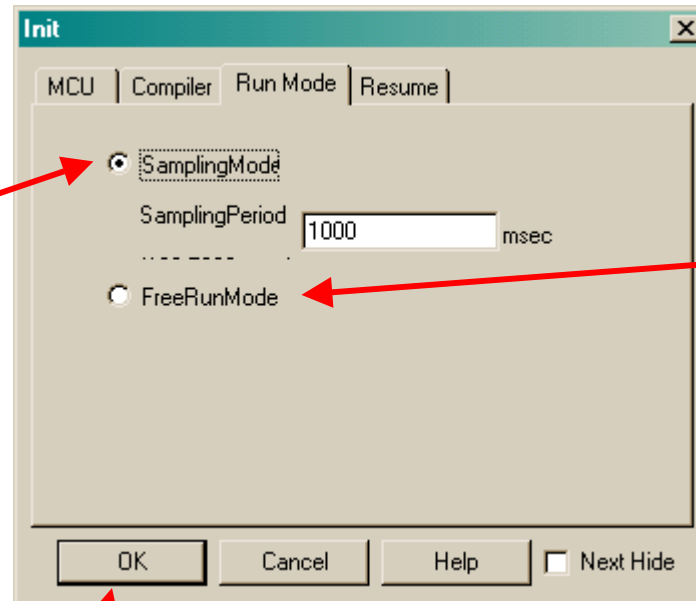
Step 1. Click on 'Refer..' and select 'M30262F8.MCU'.



Step 2. Select USB

KD30 Init Window (2/2)

For full debugging features, be sure 'Sampling Mode' is selected.



'Free Run Mode' is for real time execution of your program, but debugging is limited. Do **NOT** select for this tutorial.

Now click 'OK' to open KD30's Program window (be sure hardware is connected). If you get an error, check all connections. See SKP user's manual on 'Troubleshooting' for details.

Note 1. See KD30 User's Manual or Help for the differences between Sampling Mode and Free Run Mode. Also, see the ICD (RTA-FoUSB-MON) User's Manual for details on how ICD works under these two modes.

KD30 Program Window

The screenshot shows the KD30 software interface. The main window is titled "Program Window" and displays a disassembly table. The table has the following columns: Address, BRK, PASS, Objcode, Label, and Mnemonic. The row at address 0D0000 is highlighted in yellow. The status bar at the bottom indicates "Ready" and "MCU : STOP".

Address	BRK	PASS	Objcode	Label	Mnemonic
0D0000	-	-	EB400205		LDC #0502H, ISP
0D0004	-	-	EB300000		LDC #0000H, FLG
0D0008	-	-	EB600004		LDC #0400H, SB
0D000C	-	-	EB200F00		LDC #000FH, INTBH
0D0010	-	-	EB1000D0		LDC #D000H, INTBL
0D0014	-	-	B4		MOU .B #0, R0L
0D0015	-	-	AA0004		MOU .W #0400H, A1
0D0018	-	-	75C30000		MOU .W #0000H, R3
0D001C	-	-	7CEA		SSTR .B
0D001E	-	-	B4		MOU .B #0, R0L
0D001F	-	-	AA0004		MOU .W #0400H, A1
0D0022	-	-	75C30000		MOU .W #0000H, R3
0D0026	-	-	7CEA		SSTR .B
0D0028	-	-	B4		MOU .B #0, R0L
0D0029	-	-	AA0004		MOU .W #0400H, A1
0D002C	-	-	75C30200		MOU .W #0002H, R3
0D0030	-	-	7CEA		SSTR .B
0D0032	-	-	B4		MOU .B #0, R0L
0D0033	-	-	AA0204		MOU .W #0402H, A1
0D0036	-	-	75C30000		MOU .W #0000H, R3

KD30 will disassemble the flash contents or display 'UND' if the flash is blank.

KD30 Toolbar

Go Button

Executes target program

Step Button

One step execution of target program

Break Button

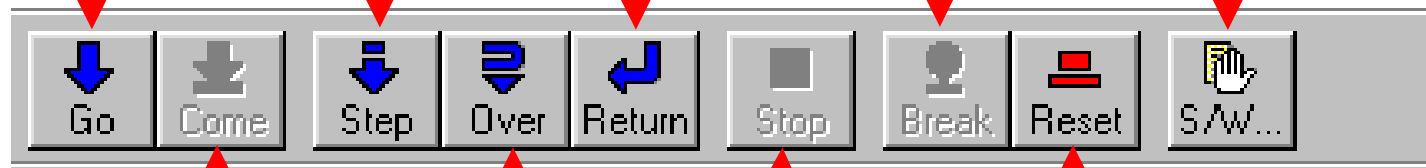
Sets a software breakpoint at the current cursor position

Return Button

Runs the program up to the higher routine

S/W Button

Sets a software breakpoint



Come Button

Executes the target program from the value in the program counter to the position of the cursor in the window

Stop Button

Stops execution of the target program

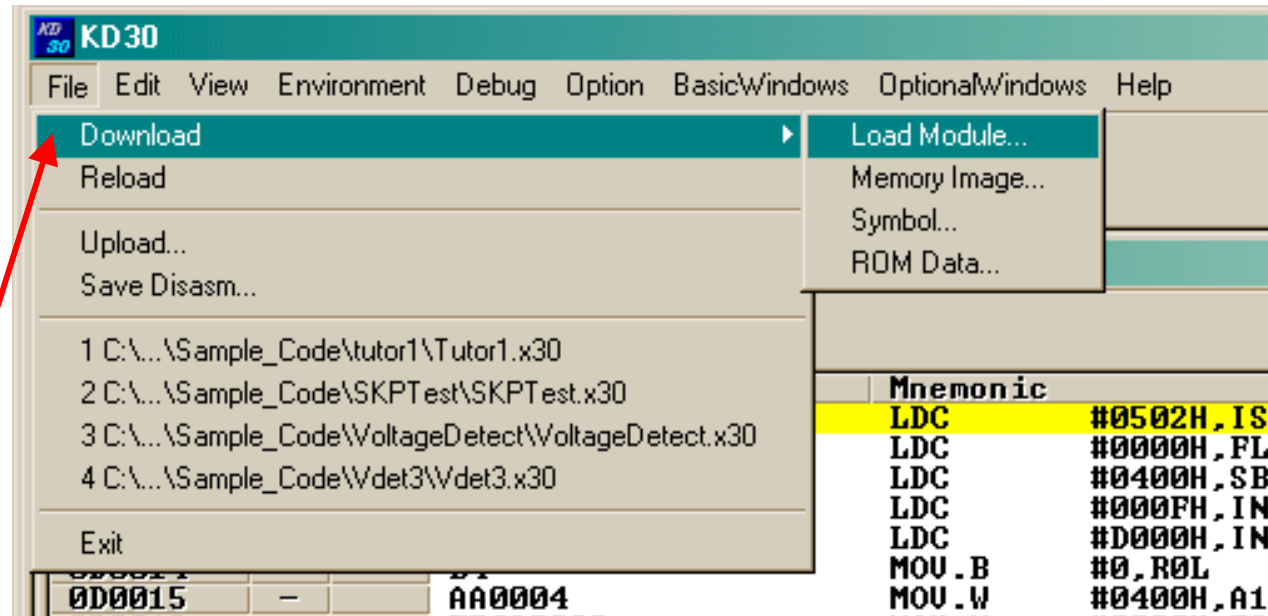
Over Button

Step over function/subroutine call

Reset Button

Resets the target program

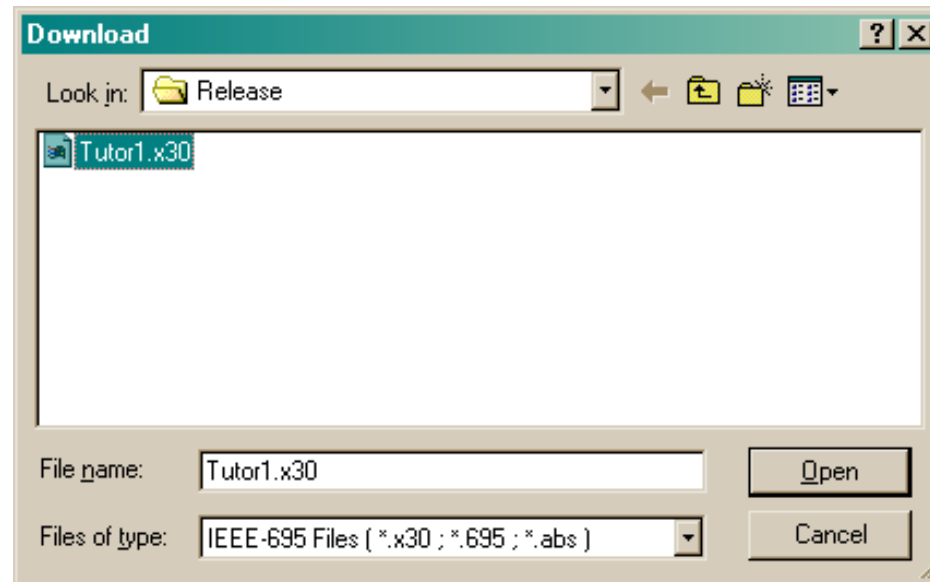
Download a Program to the SKP16C26 Board (M16C/26 MCU)



Click on 'File', then select 'Download', 'Load Module'...

Note: When you download code or program in KD30, the program counter is automatically reset to the address the reset vector points to.

Download a Program to the SKP16C26 Board (M16C/26 MCU)



From the `c:\MTOOL\SKP16C26\sample_code\tutor1\tutor1\release` folder, select 'tutor1.x30'.

Download a Program to the SKP16C26 Board (M16C/26 MCU)

After downloading the program, KD30 opens the source file where the reset vector is.

KD30 [C:\MT00L\SKP16C26\Sample_Code\Tutor1\Tutor1\Release\Tutor1.x30]

File Edit View Environment Debug Option BasicWindows OptionalWindows Help

Go Come Step Over Return Stop Break Reset S/W...

Program Window [ncrt0_tutor1.a30]

View Source Mix Dis...

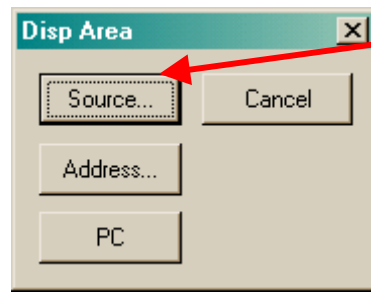
Line	BRK	PASS	Source
00035			start:
00036	-		ldc #istack_top, isp ; set istack_top
00037	-		ldc #0000h,flg ;Clear U Flag's
00038			
00039	-		ldc #data_SE_top,sb ;set sb register
00040	-		ldintb #VECTOR_ADR
00041			
00042			;=====
00043			; ***** NOTE: CHANGING BCLK SPEED HERE
00044			; SETTING CLOCK SPEED TO F1 <DIU BY 1> ON XIN: BCLK =
00045			; BECAUSE, AFTER RESET, BCLK DEFAULTS TO F8 <DIU BY 8>
00046			;=====
00047	-		mov.b #01h,0AH ; unprotect CM
00048	-		mov.b #08h,06H ; enable CM17

Ready MCU : STOP

Current location of MCU program counter is highlighted.

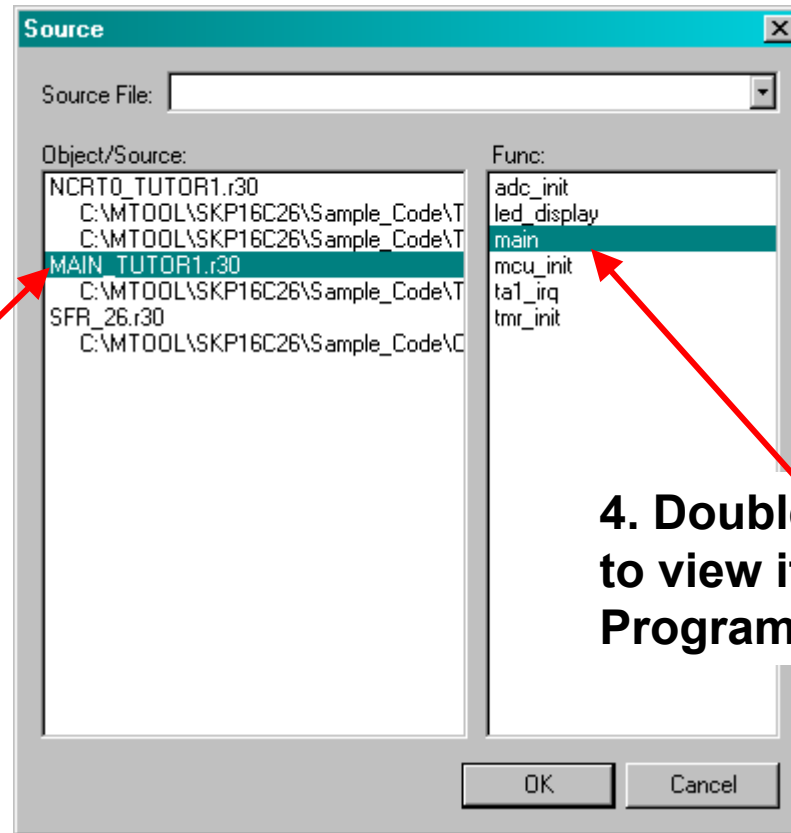
Now click on "View" to see the program source code...

Viewing Source Files in the Project



1. Click 'source'

2. Source window is displayed.



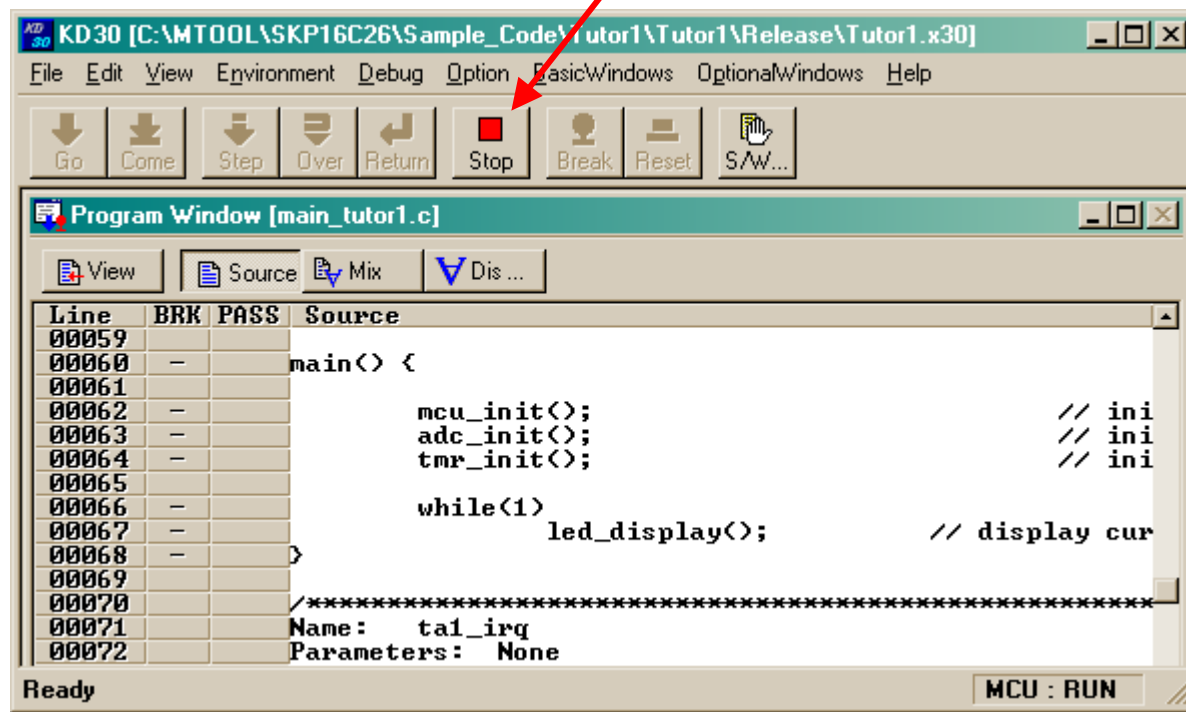
3. Click 'main_tutor1.r30'

4. Double-click 'main' to view it on the Program Window



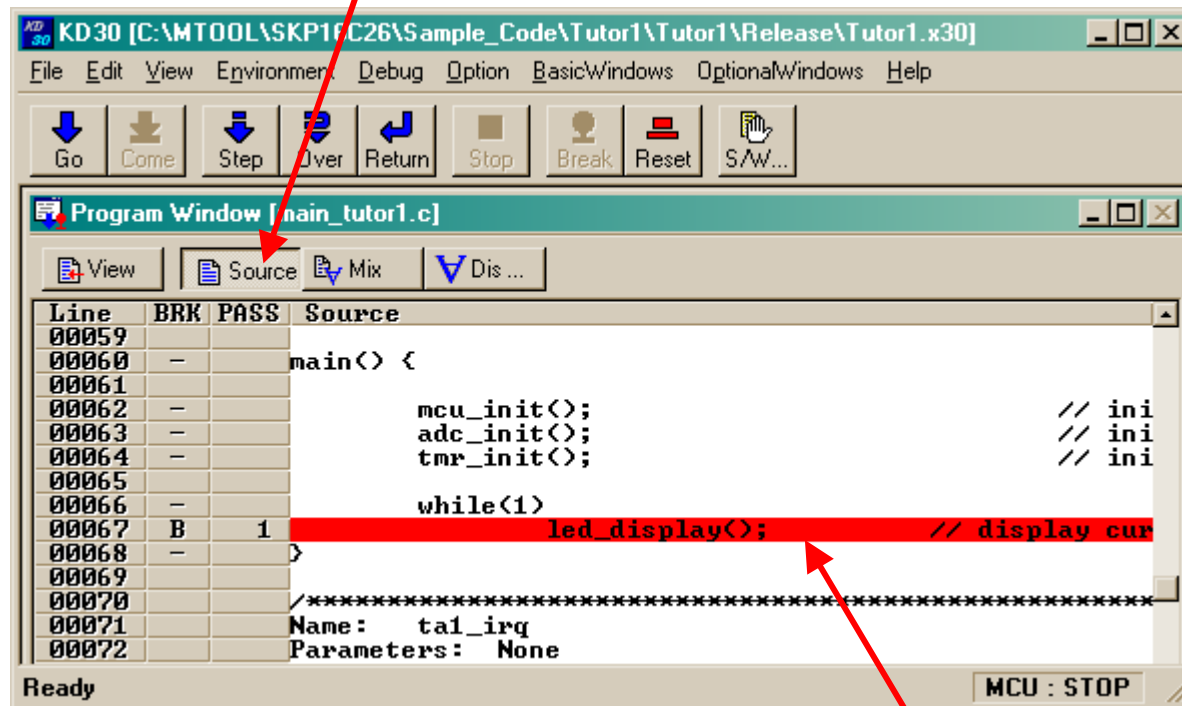
Stopping Program Execution

Click on the 'Stop' icon to stop the program



Setting Breakpoints

1. Click on the 'Source' to view source code only (not MIX display).



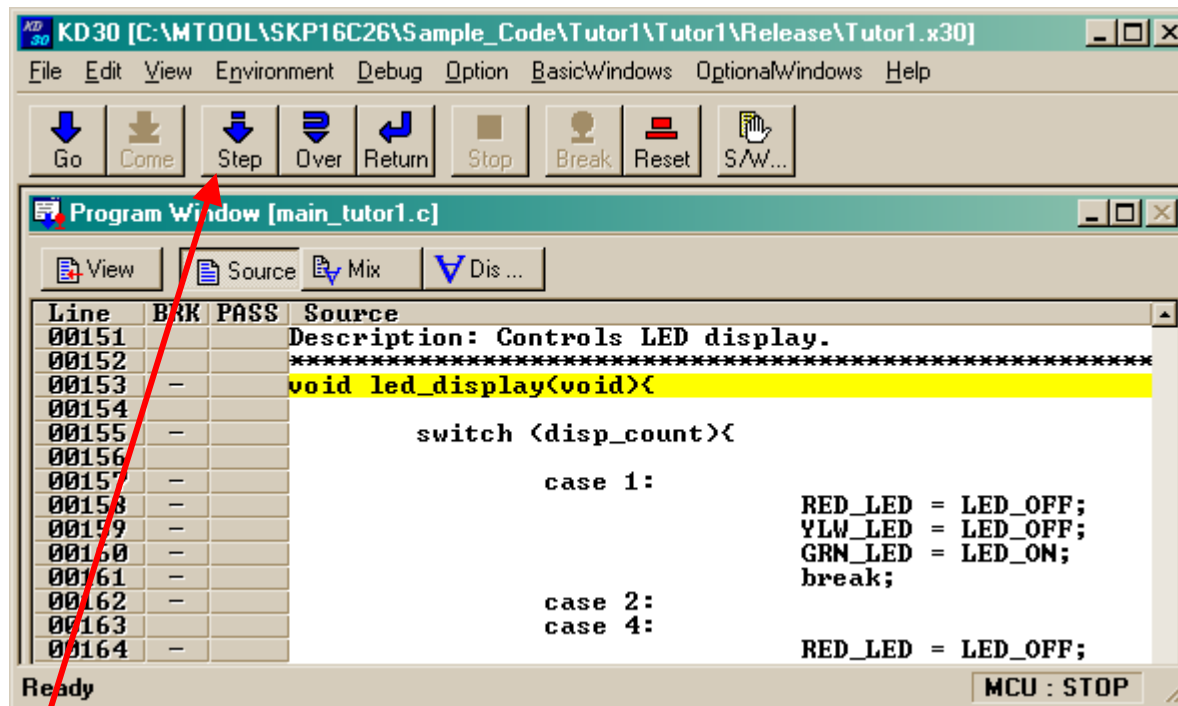
The screenshot shows the KD30 IDE interface. The 'Program Window' is open to the source code of 'main_tutor1.c'. The 'View' menu is open, and 'Source' is selected. The source code is displayed in a table format with columns for Line, BRK, PASS, and Source. A breakpoint is set on line 00067, indicated by a 'B' in the BRK column and a red highlight on the source code line. A red arrow points from the 'Source' button in the View menu to the 'Source' column header, and another red arrow points from the 'B' in the BRK column to the 'led_display();' line.

Line	BRK	PASS	Source
00059			
00060	-		main() {
00061			
00062	-		mcu_init(); // ini
00063	-		adc_init(); // ini
00064	-		tmr_init(); // ini
00065			
00066	-		while(1)
00067	B	1	led_display(); // display cur
00068	-		}
00069			
00070			/*-----*/
00071			Name: ta1_irq
00072			Parameters: None

2. Locate and then set a breakpoint on 'led_display();' by a double-click on '-' in the 'BRK' column that denotes an executable line. A 'B' will appear in its place after the breakpoint is set and the line is highlighted in red.

3. Click on 'Go' icon to run program...

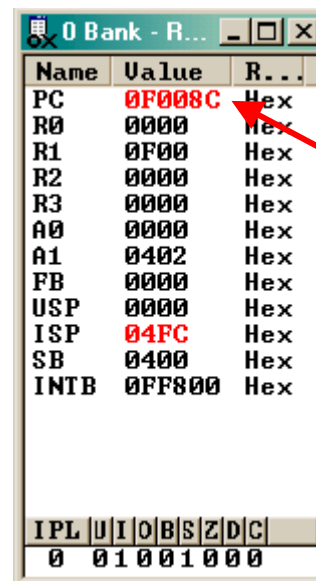
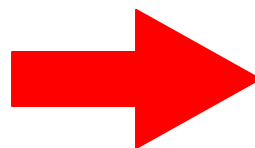
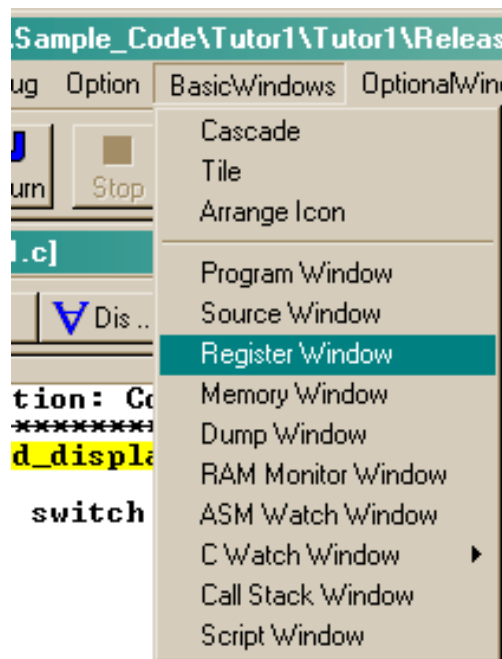
Program 'Stepping'



Try 'stepping' a few lines of code by clicking on 'Step' icon. Click on 'Go' afterwards to run program again.

Basic Windows: Register

Now open the 'Register' window

A screenshot of the Register window. It displays a table of CPU registers with their names, values, and data types. The values for PC, R0, R1, R2, R3, A0, A1, FB, USP, ISP, SB, and INTB are shown in red. The PC register value is 0F008C. The R0 register value is 0000. The R1 register value is 0F00. The R2 register value is 0000. The R3 register value is 0000. The A0 register value is 0000. The A1 register value is 0402. The FB register value is 0000. The USP register value is 0000. The ISP register value is 04FC. The SB register value is 0400. The INTB register value is 0FF800. The data type for all registers is Hex. At the bottom of the window, there is a status bar with the text 'IPL UI O B S Z D C' and the value '0 01001000'.

Values in red indicate changes since last "viewed". Try 'stepping' and note the changes.

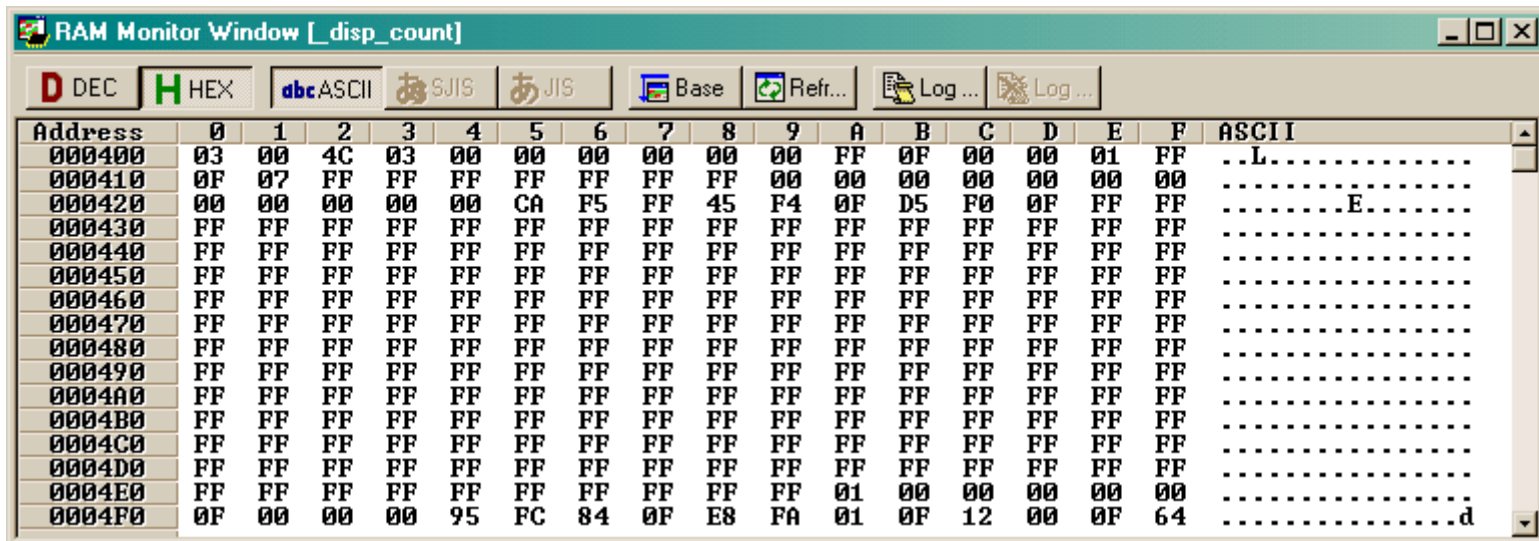
The Register window displays the values of the CPU registers after executing an instruction.

Note: Resize the Register window as needed.

Basic Windows: RAM Monitor

Open a RAM Monitor window (Basic Windows > RAM Monitor Window). The RAM Monitor displays the current value of the memory area shown on the window. It is updated at a preset value which can be modified by the user.

Double-click an address and enter 400 (hex). KD30 will tell you the page is going to change, click 'OK' (adjust the window size as needed).

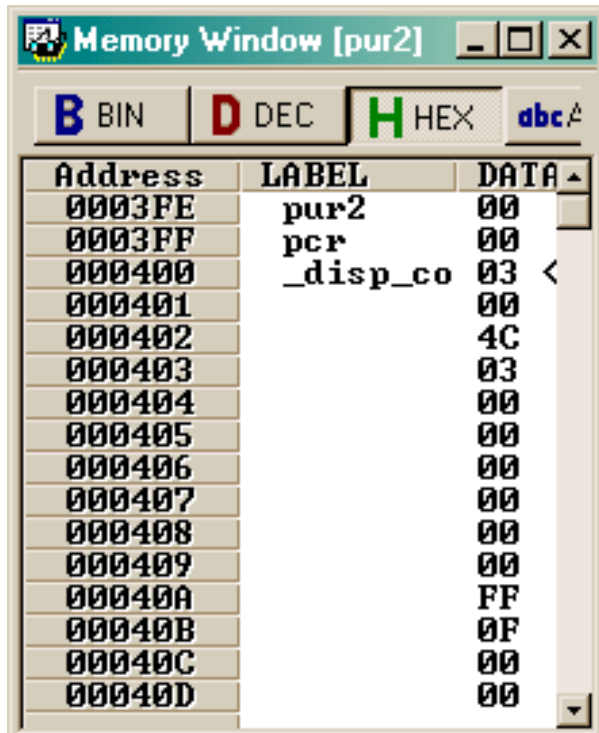


Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
000400	03	00	4C	03	00	00	00	00	00	00	FF	0F	00	00	01	FF	..L.....
000410	0F	07	FF	FF	FF	FF	FF	FF	FF	00	00	00	00	00	00	00
000420	00	00	00	00	00	CA	F5	FF	45	F4	0F	D5	F0	0F	FF	FFE.....
000430	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000440	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000450	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000460	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000470	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000480	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000490	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0004A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0004B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0004C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0004D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0004E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	01	00	00	00	00	00
0004F0	0F	00	00	00	95	FC	84	0F	E8	FA	01	0F	12	00	0F	64d

Click the 'GO' icon. Note you can view the RAM as it is updating. This function is not available in "Free Run" mode. Click the 'STOP' icon before proceeding.

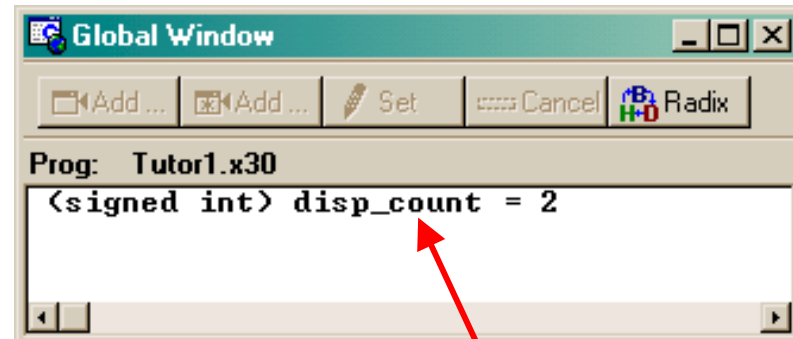
Basic Windows: Memory & C Watch

Open a Memory window (Basic Windows > Memory Window).



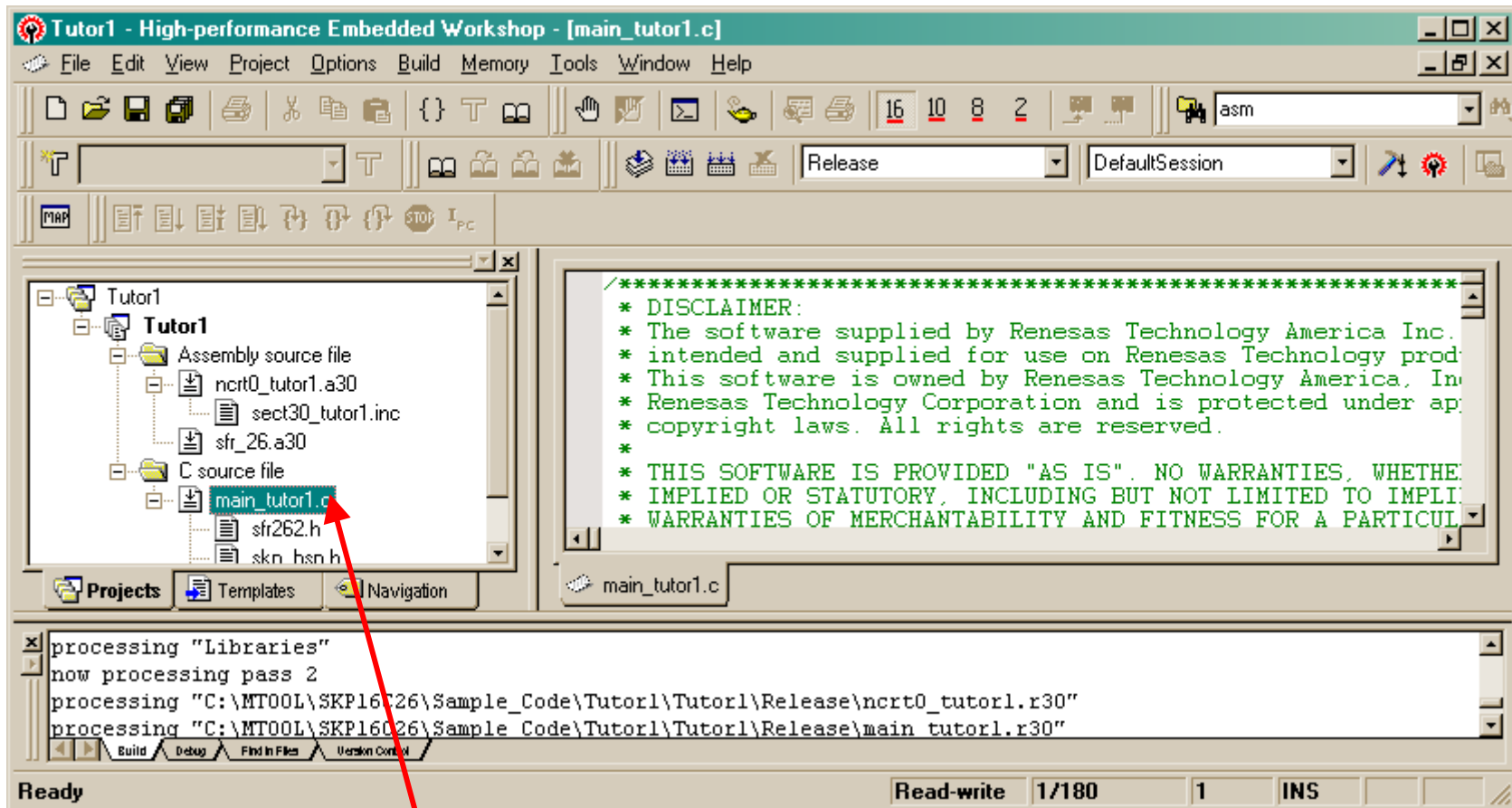
The 'Memory Window' displays the location and contents of variables

Open a C Watch window (Basic Windows > C Watch Window). The 'C Watch Window' allows you to view globals and locals. An example is shown below.



Double-click on the variable to change display format: i.e., change 'char' to 'hex' to 'decimal', etc.

Modifying the Program (1/2)



If `main_tutor1.c` is not shown on the Editor window, double-click on it in the Workspace window and the file will be opened/displayed on the Source window.

Modifying the Program (2/2)

```
by controlling the LED control variable.
*****
void ta1_irq(void){
    adst = 1;           // Start A2D conversion
    while(adst == 1);  // wait for A/D conversion start
    ta1 = ad0;         // read AD value and preload Timer
                    // LED switching rate

    ++disp_count;     // increment display control variable
    if (disp_count > 4) // if LED control variable exceeds 4
        disp_count = 1; // return to initial state
}

```

main_tutor1.c

Read-write 82/180 11 INS

1. Scroll down and find the function 'ta1_irq' routine.

2. Change this line to 'ta1 = (0x3FF - ad0);'.

3. Click this to save the revised file.

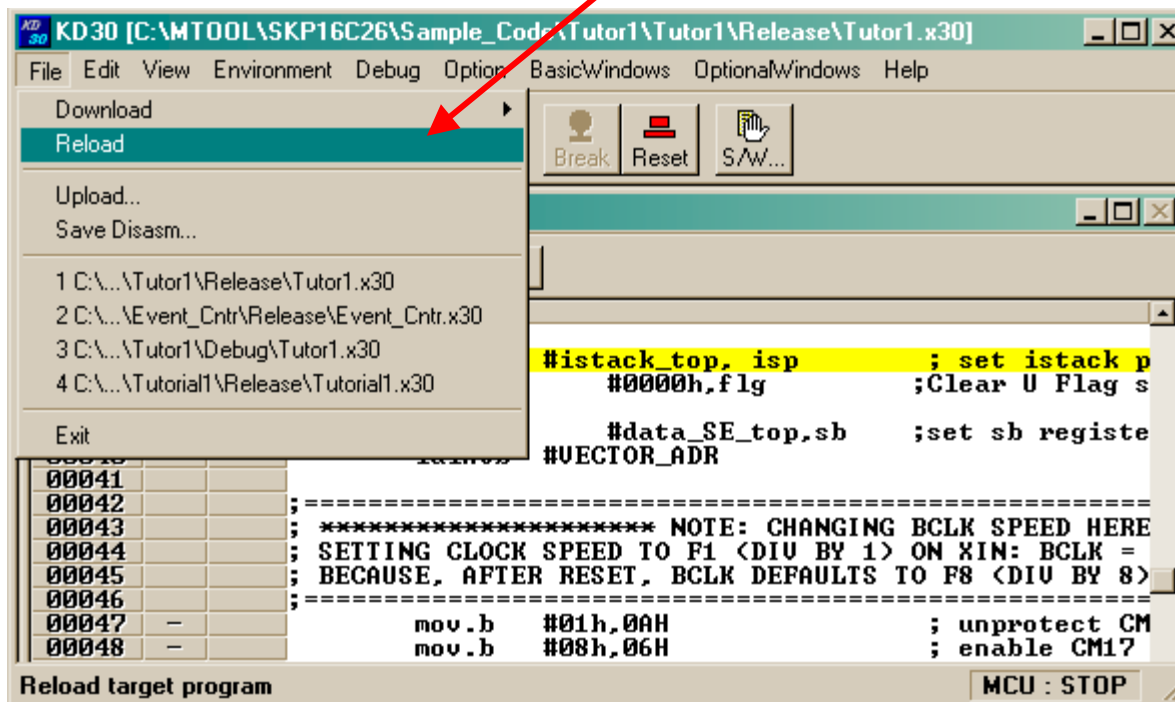


4. Build the project again.



Load (re-load) Modified Program

In KD30, with the program stopped, reload code by selecting 'Reload' from the File menu.



Turning Analog Adjust potentiometer on SKP16C26 Board clockwise decreases the LED blink rate. Turning it counter-clockwise increases the blink rate.

End of Tutorial

This is the end of the tutorial. You can try downloading other sample programs from the \Sample_Code directory.

For a tutorial on creating a new project, check Tutorial 2 for details.

In addition, check out the references on the next page.

Have Fun!!



References and Recommended Reading

All documents that came with the SKP can be found using the “Document Description” from the Start > Programs > Renesas-Tools > SKP16C26 menu.

- **SKP16C26 User’s Manual:** This is a “must read” document! It details all the things you need to know on how to use the Starter Kit.
- **HEW User’s Manual:** To fully understand and get the most out of HEW, this is recommended reading.
- **KD30 Version X.XX Help:** The tutorial only covered the basics of KD30. Check out the Help menu to find out all of KD30’s features.
- **NC30 Version X.XX User’s Manual:** Check this manual out for features specific to the NC30 compiler.
- **M16C/26 Datasheet and SKP16C26 Board Schematic:** These are required to write user application programs.
- **RTA-FoUSB-MON User’s Manual:** Read this manual to understand how the ICD works.

References and Recommended Reading

- **M16C/10/20/60 Series C Language Programming Manual:** This is a great document for any level of programmer. The first chapter is an intro to C programming. The next chapter explains the memory map of C programs on microcontrollers and the role of startup programs.
- **M16C/10/20/60 Series Software Manual:** This document describes the instruction set and timing information for the M16C/20/60 series CPU cores.
- **AS30 Version X.XX User's Manual:** Read this manual if you plan on writing programs in Assembly or when making changes to the startup file.
- **Application Notes and Sample Programs:** Application notes and other sample programs can be accessed from Renesas Technology America's website: <http://www.renesas.com>.

