

# **SKP16C62P**

## **Tutorial 1**

### **Software Development Process Using TM (Tool Manager)**

# Overview

The following tutorial is a brief introduction on how to develop and debug programs using the software and hardware tools included with the SKP16C62P.

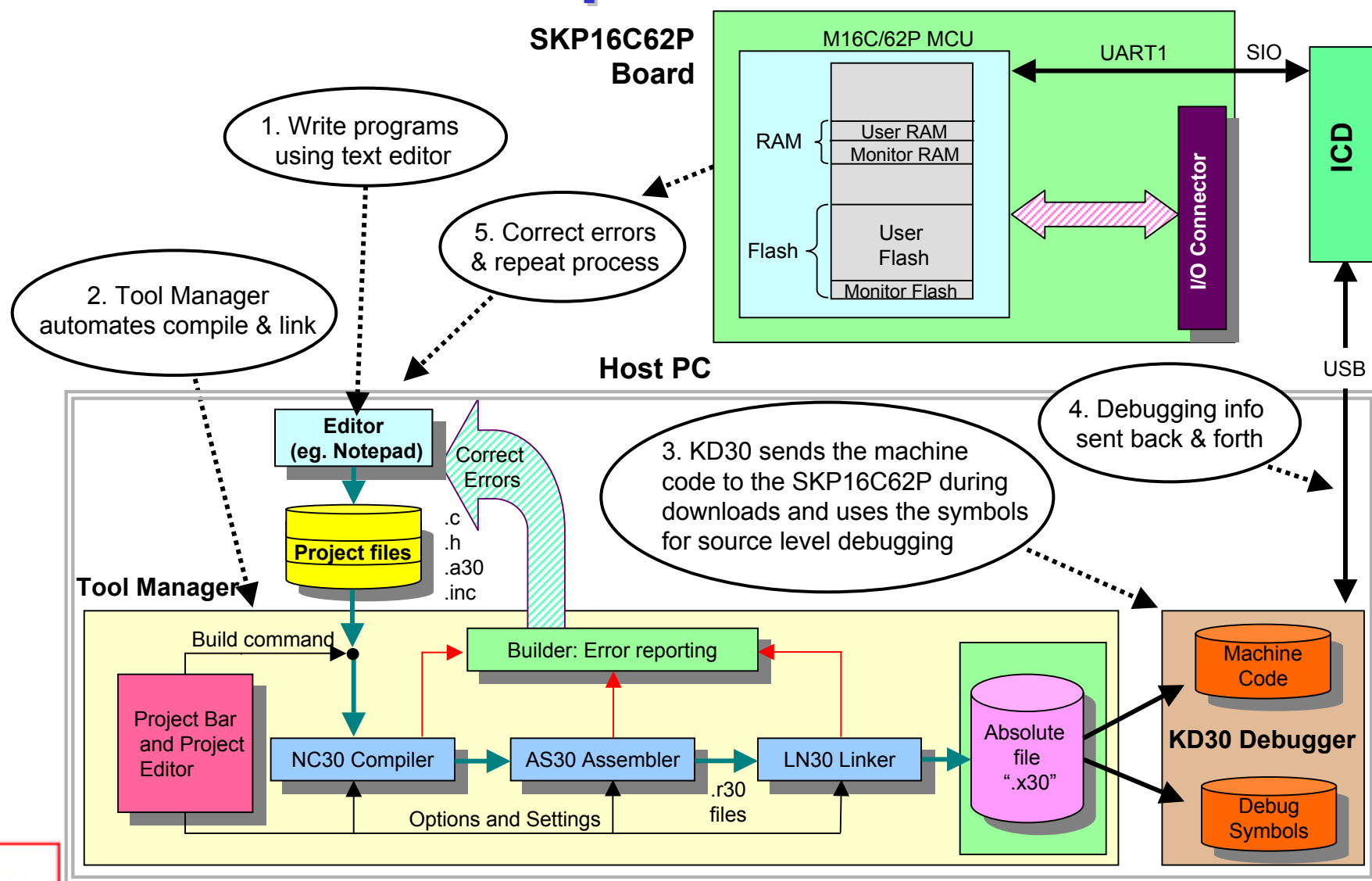
To get the most out of the Starter Kit, check out the references at the end of this tutorial.

**Note:**

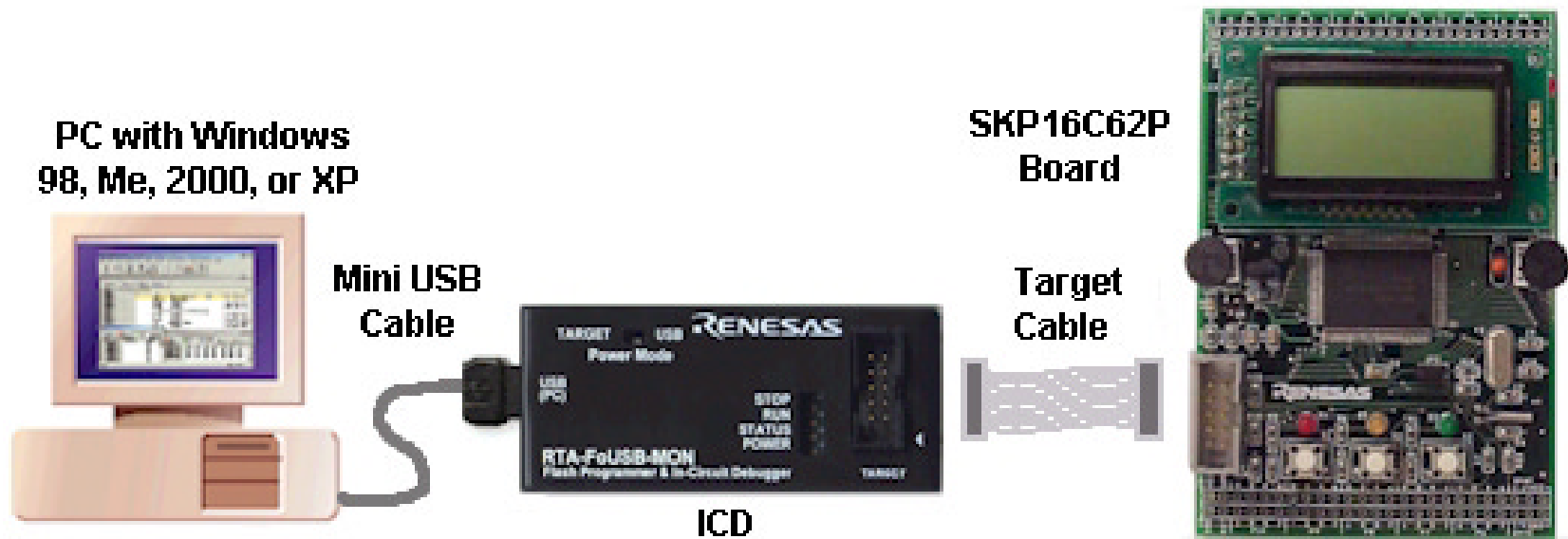
*The tutorial assumes the user has followed the 'Quick Start Guide' and all the software tools and examples have been installed in the default directories.*



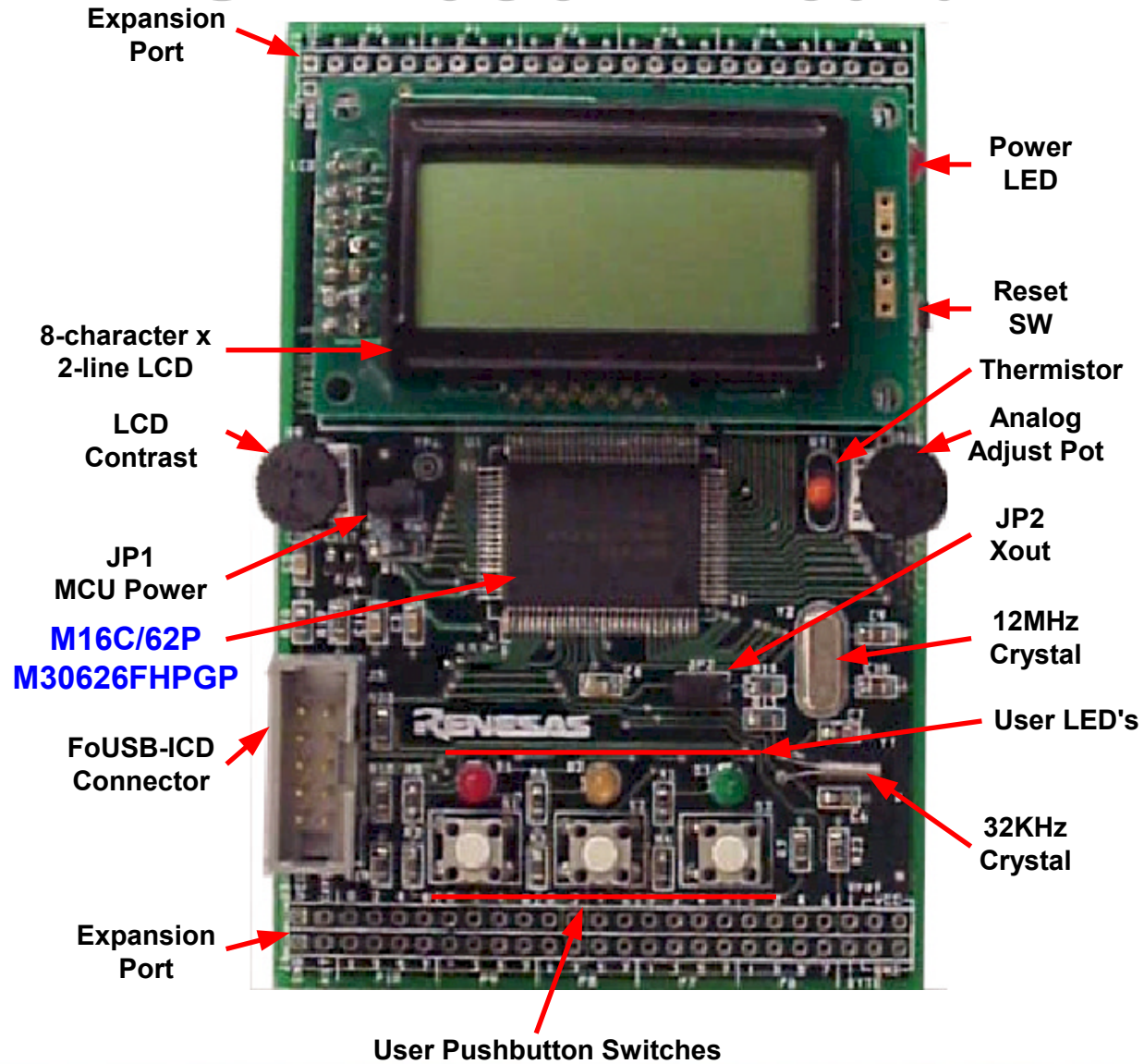
# The Development Process



# SKP16C62P Connectivity



# SKP16C62P Board



# SKP16C62P Board Features

## M16C/62P(M30626FHPGP) MCU

- 24MHz Operating Frequency at 3.0V – 5V
- 384kB Flash ROM, 4kB Virtual EEPROM, and 31kB RAM
- 87 GPIO and 4 Key-on Wakeup Inputs
- 11 Timers plus a Watchdog Timer
- 26-channel 10-bit ADC
- 2-channel 8-bit DAC
- 2 DMAC
- 5 SIO's (supports I<sup>2</sup>C and SPI)
- CRC Circuit

## Onboard User Controls

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

# ICD (RTA-FoUSB-MON)

The ICD (In-Circuit Debugger) provides a USB interface to the Host PC and communicates commands and data to and from the SKP16C62P 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 SKP16C62P Board . This kernel provides a communication interface between the M16C/62P MCU and the ICD + KD30 Debugger application on MCU status. While the kernel uses some resources of the M16C/62P, 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/62P MCU on the SKP16C62P Board and other Renesas' M16C Flash MCU's (the ICD will support other M16C MCU's by downloading a 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

## Tool Manager

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

## KNC30

C-compiler (limited version of NC30). Conforms to ANSI C standards

## AS30

Relocatable Assembler

Supports structured language and wide variety of macro instructions

## Flash-over-USB™ Programmer

Flash programmer for Renesas M16C Flash MCU's.



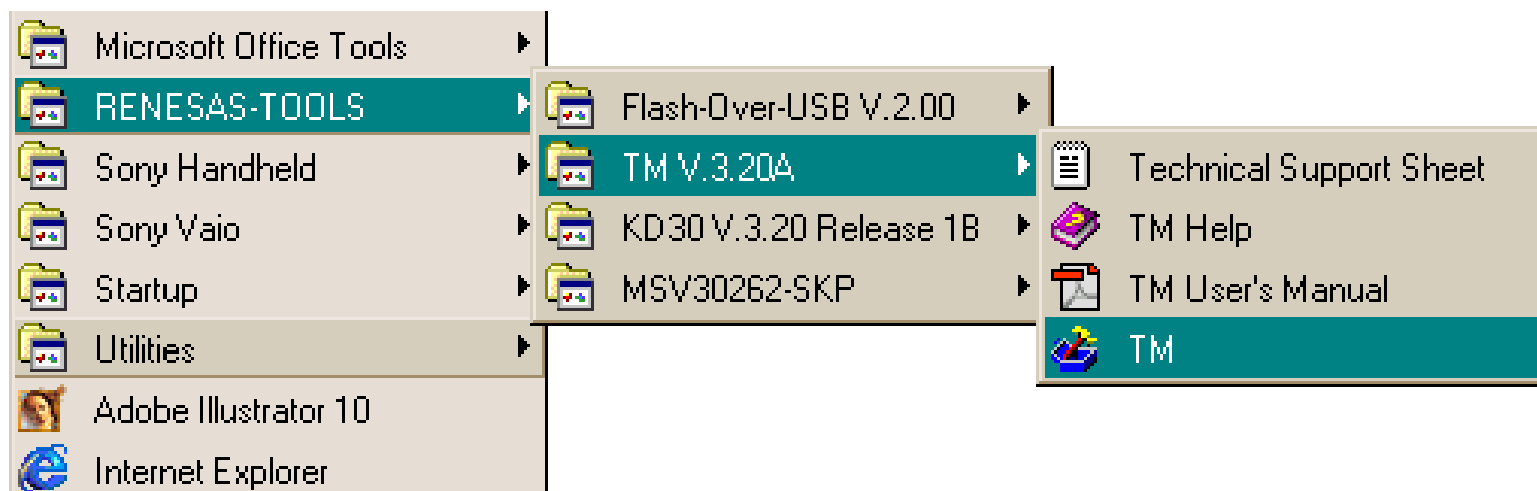
# Tool Manager Overview

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).

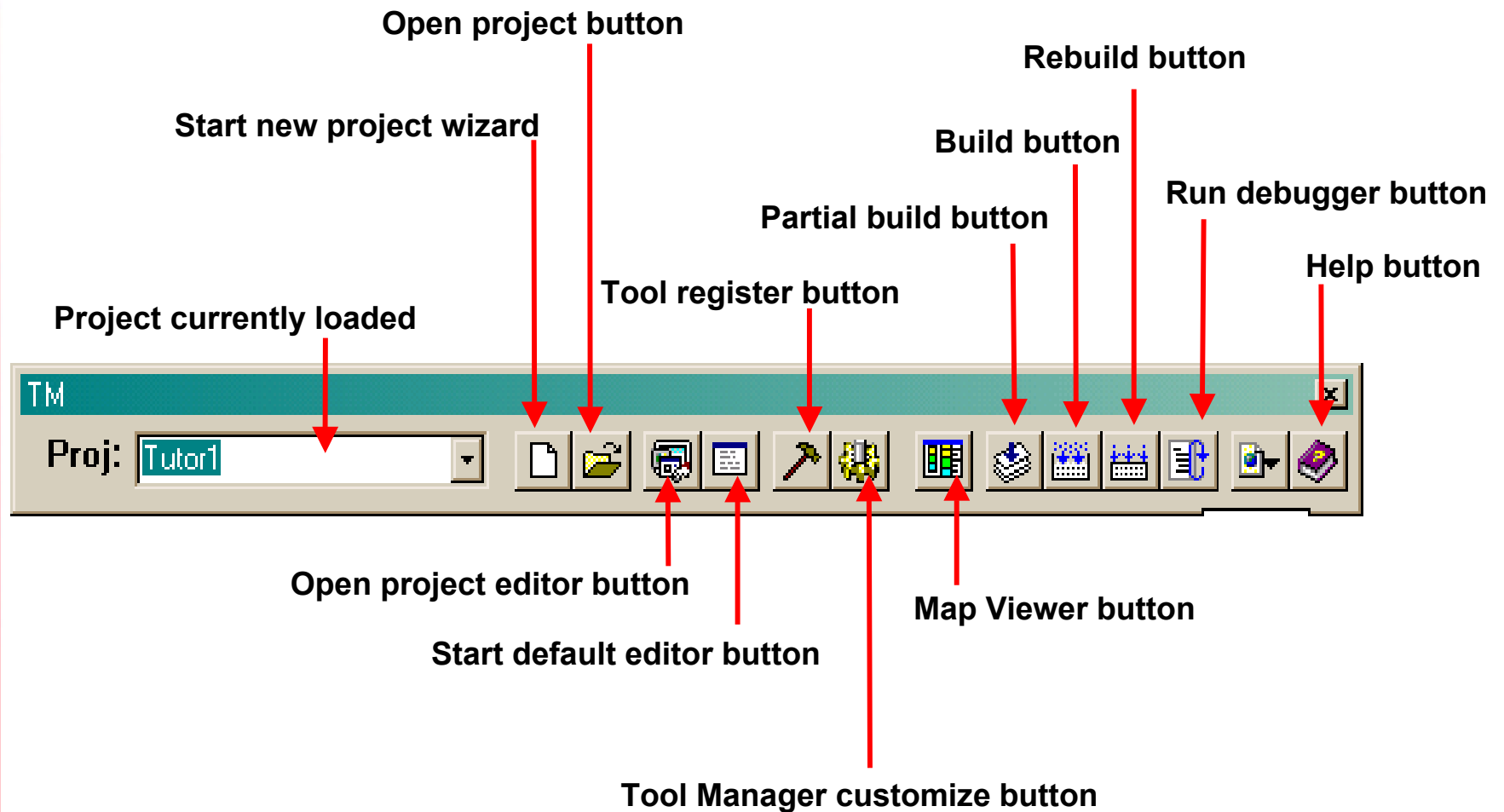
After all the files in a **project** are compiled and assembled, a **linker** combines all the files into a single file. These steps can be tedious and repetitive. To make the process simple, we use an **Integrated Development Environment (IDE)** called “**Tool Manager (TM)**”.

# Start Tool Manager (TM)



From the Windows Start menu, click on  
**Programs > Renesas-tools > TM V3.X > TM**

# Tool Manager Project Bar



# Tool Manager Exercise

We will now perform the following steps:

- Open an existing project
- Build (re-build) an existing project

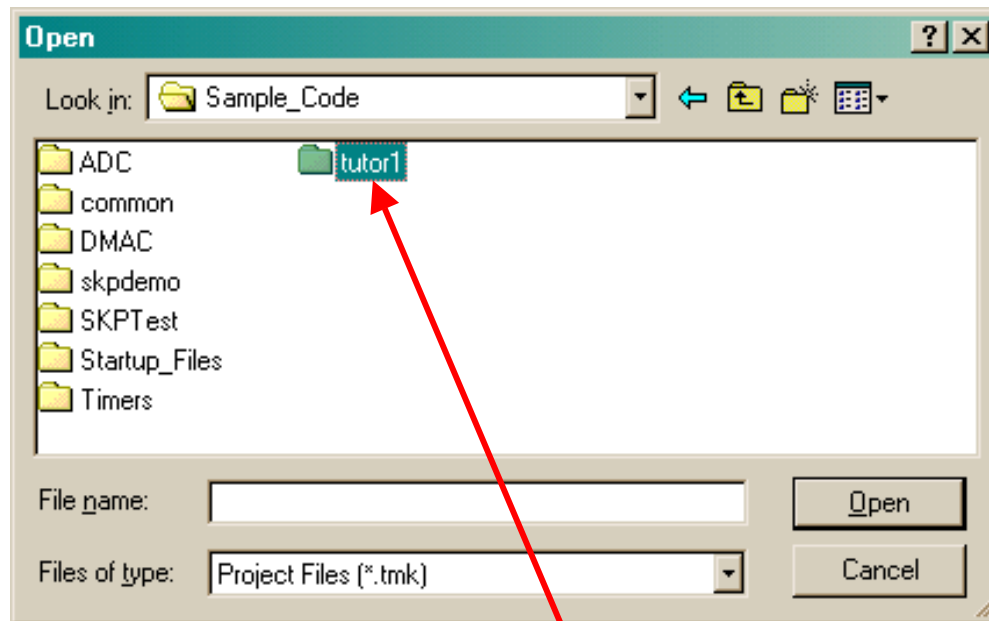


# Open an Existing Project: Step 1



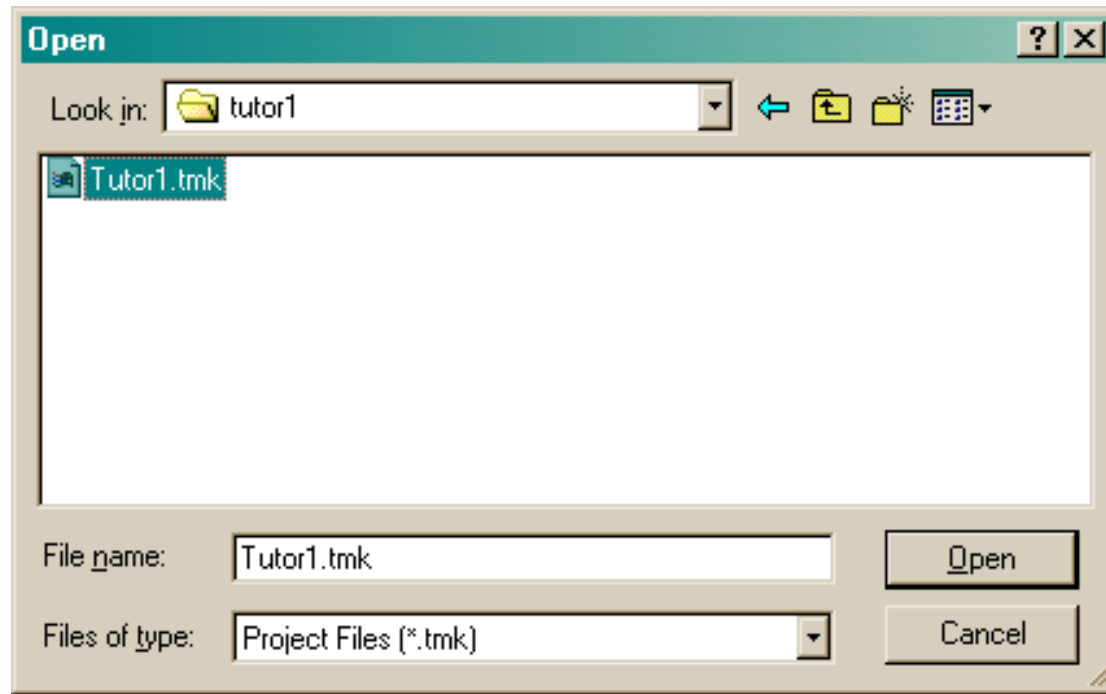
To open an existing project, click on this icon.

# Open an Existing Project: Step 2



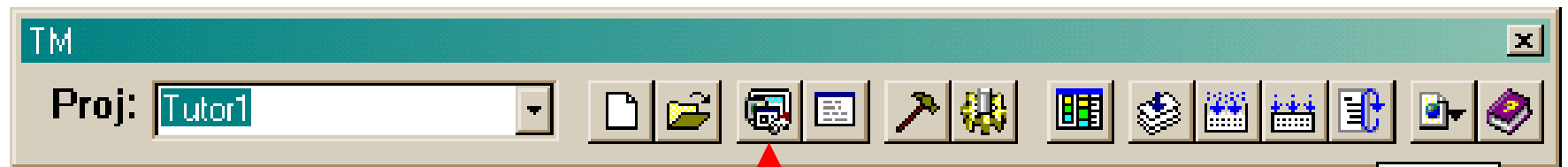
Locate the **C:\MTOOL\SKP16C62P\sample\_code** directory and open the “**tutor1**” folder.

# Open an Existing Project: Step 3



Select “**tutor1.tmk**”

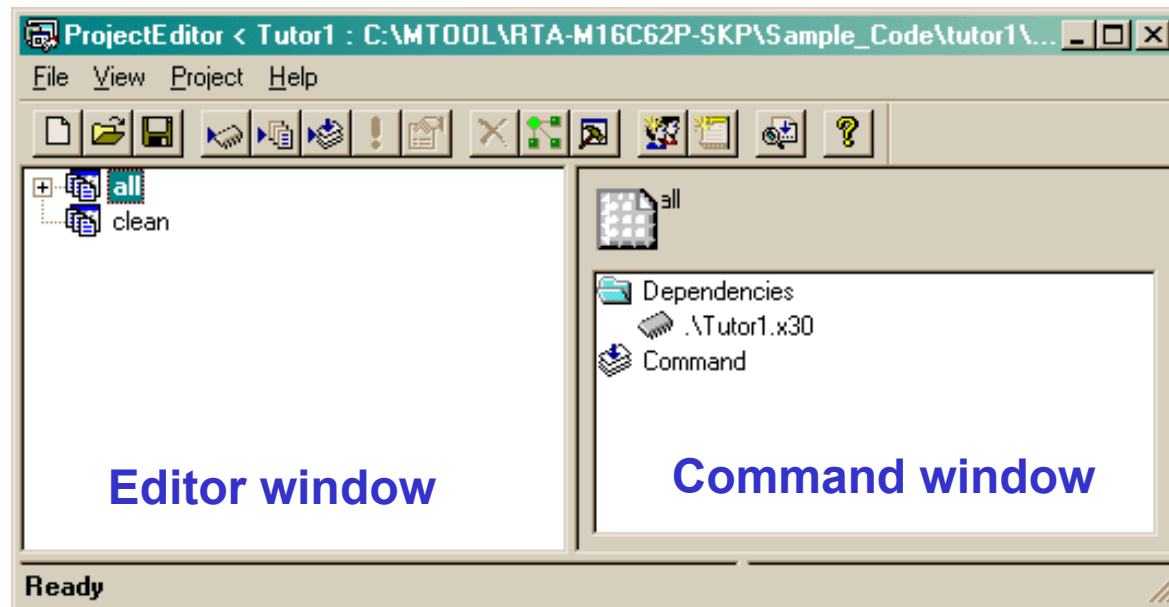
# Open the Project Editor



Click on 'Project Editor' icon.



# Project Editor



Normally, you will only be concerned with the editor window. Now click on the '+' by 'all', then click on the '+' by 'tutor1.x30', then the '+' by 'tutor.r30'.

# Editor Window

This illustrates that the object file 'main\_tutor1.r30' depends on these source files.



Double-click on '[main\\_tutor1.c](#)' to open it in the default editor, 'Notepad'. Tool Manager can also be set up to use your favorite editor. To change the default editor, click on the 'Tool Register' icon on the 'Tool Bar'.

# Build(re-build) an Existing Project



Build the project into an executable module by clicking on the '**Rebuild**' icon. This will re-compile and link all the source files.

Clicking the '**Build**' icon will only compile files that have been modified since the last build.

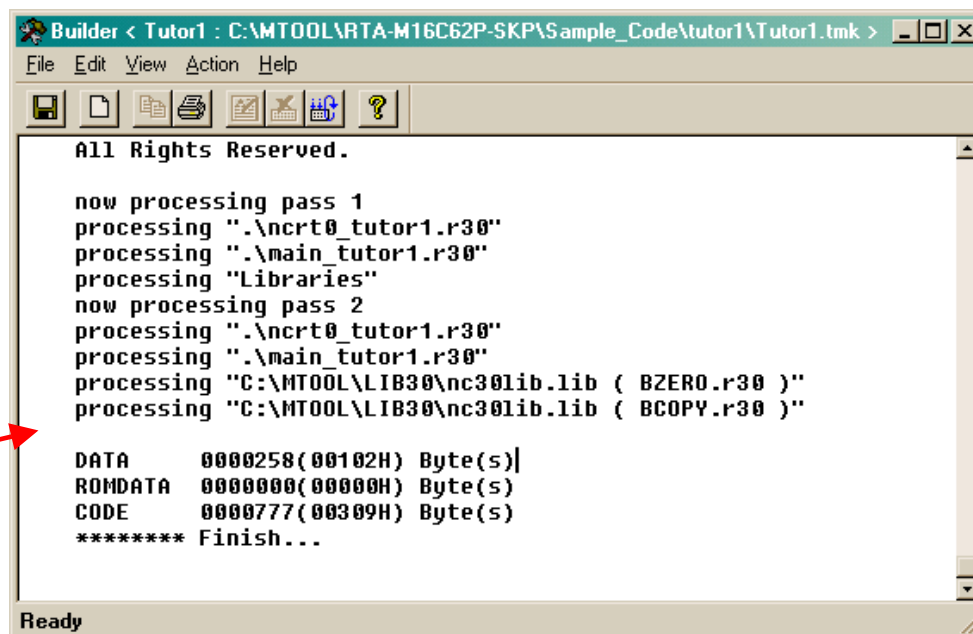
Always '**Rebuild**' when any compiler or Tool Manager options have changed.

Clicking on any of the build icons opens the 'Builder' window...

# Builder Window

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

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 were found, '... Finish...' will be displayed.



The screenshot shows the Builder window with the following text:

```
Builder < Tutor1 : C:\MT00L\RTA-M16C62P-SKP\Sample_Code\tutor1\tutor1.tmk >
File Edit View Action Help
[Icons: Save, Open, Print, Find, Run, Stop, Help]

All Rights Reserved.

now processing pass 1
processing ".\ncrt0_tutor1.r30"
processing ".\main_tutor1.r30"
processing "Libraries"
now processing pass 2
processing ".\ncrt0_tutor1.r30"
processing ".\main_tutor1.r30"
processing "C:\MT00L\LIB30\nc30lib.lib ( BZERO.r30 )"
processing "C:\MT00L\LIB30\nc30lib.lib ( BCOPY.r30 )"

DATA      0000258(00102H) Byte(s)
ROMDATA   0000000(00000H) Byte(s)
CODE      0000777(00309H) Byte(s)
***** Finish...
```

A red arrow points from the text 'Finish...' in the paragraph to the '\*\*\*\*\* Finish...' line in the window.

Now that you have finished compiling/linking the file, the next step is to download and run the program on the SKP16C62P Board using the KD30 Debugger and ICD...

Do not close Tool Manager 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 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 the M16C/62P, the maximum no. of breakpoints for KD30 is 8.**

KD30 also 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.

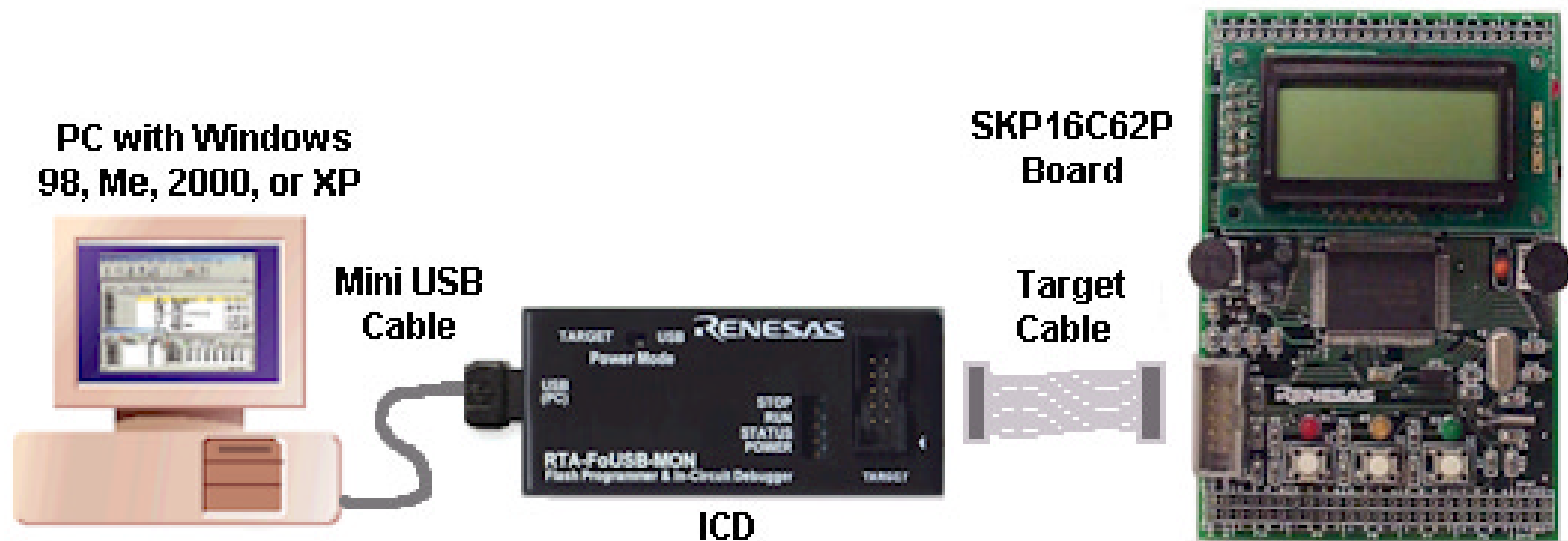
# KD30 Debugger Exercise

- Download and run a program on the SKP16C62P board
- General use of the KD30 Debugger including stepping and setting breakpoints
- Return to Tool Manager, modify the program, rebuild, and run the updated program on the SKP16C62P board



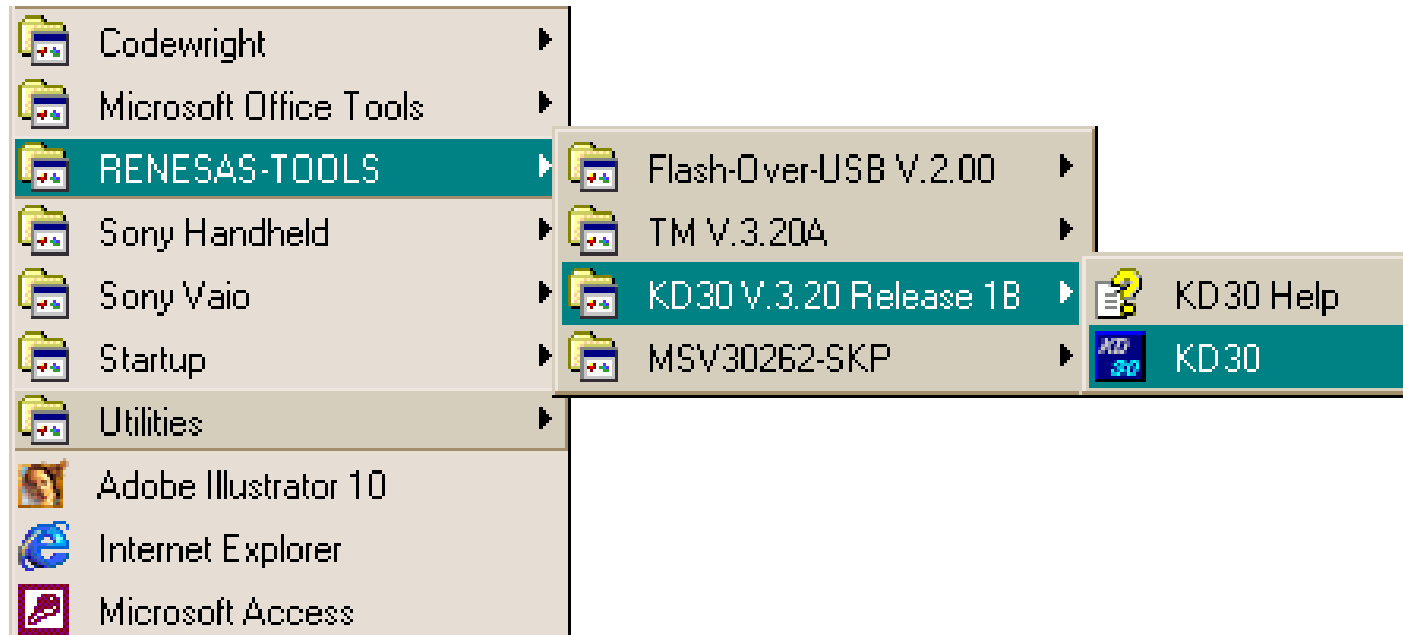
# Connect Hardware

Before starting KD30, connect the ICD to the SKP16C62P 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™).



# Start KD30

**Launch KD30 from the Windows Start Menu,**



**or from 'Tool Manager'.**

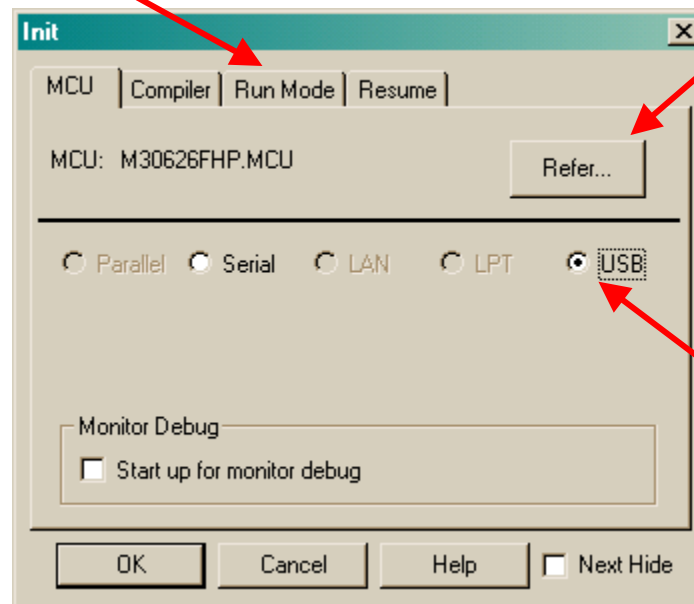




# KD30 Init Window

**Step 3. Now click the 'Run Mode' tab**

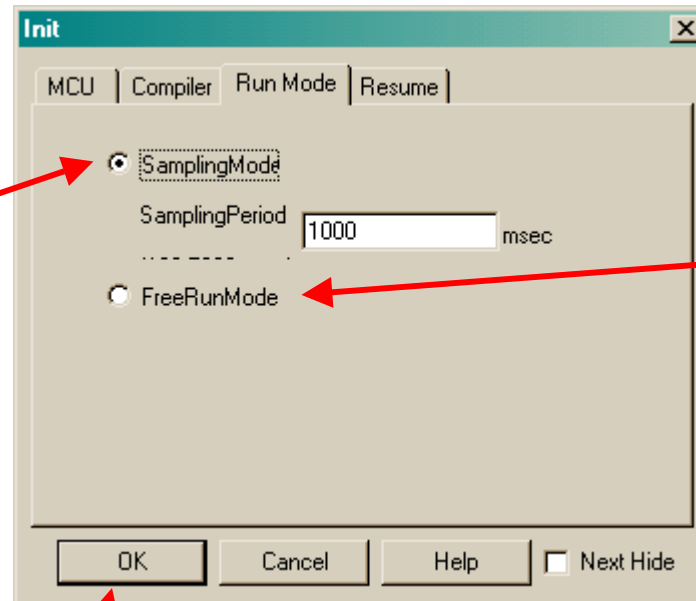
**Step 1. Click on 'Refer..' and select 'M30626FHP.mcu'.**



**Step 2. Select USB**

# KD30 Init Window

For full debugging features, be sure 'Sampling Mode'<sup>1</sup> is selected.

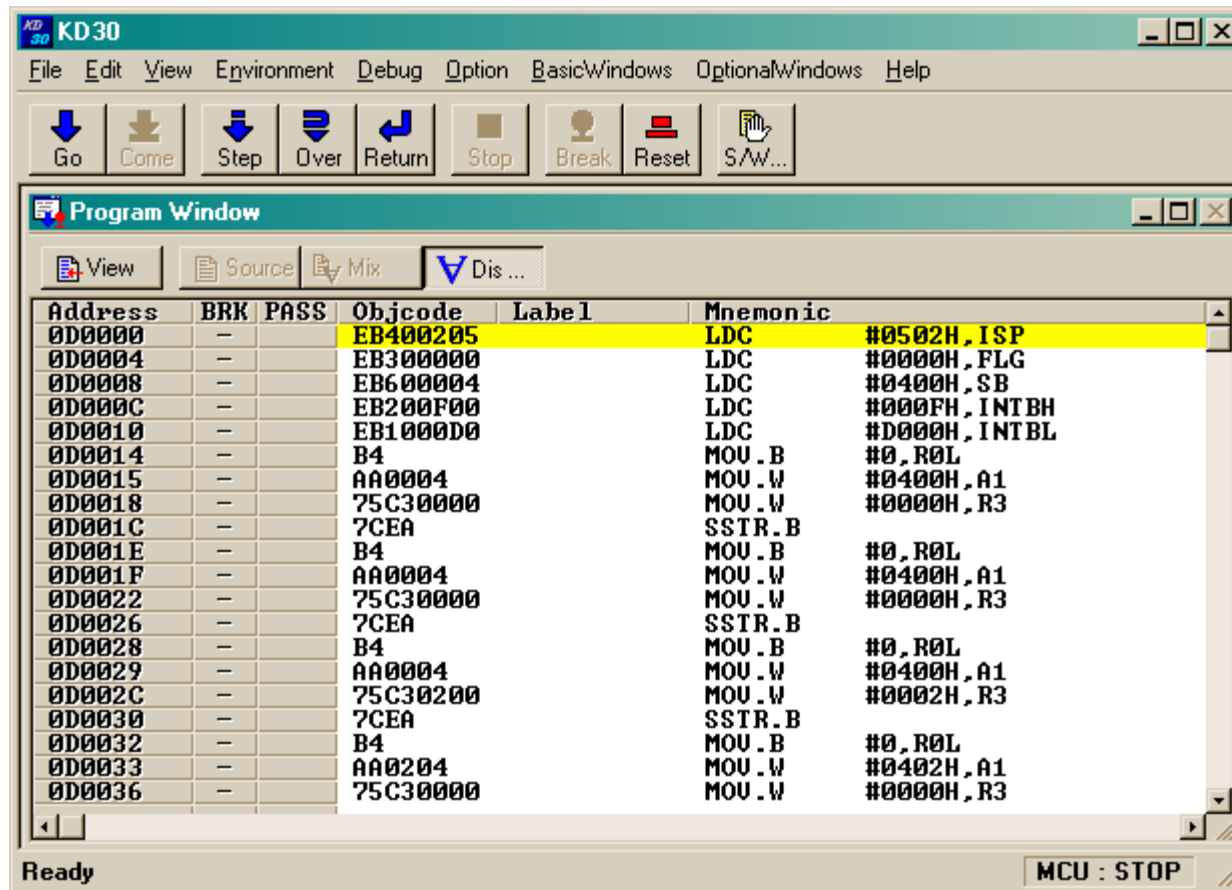


'Free Run Mode'<sup>1</sup> is for real time execution of your program, but debugging is limited. Do not select for this tutorial.

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.

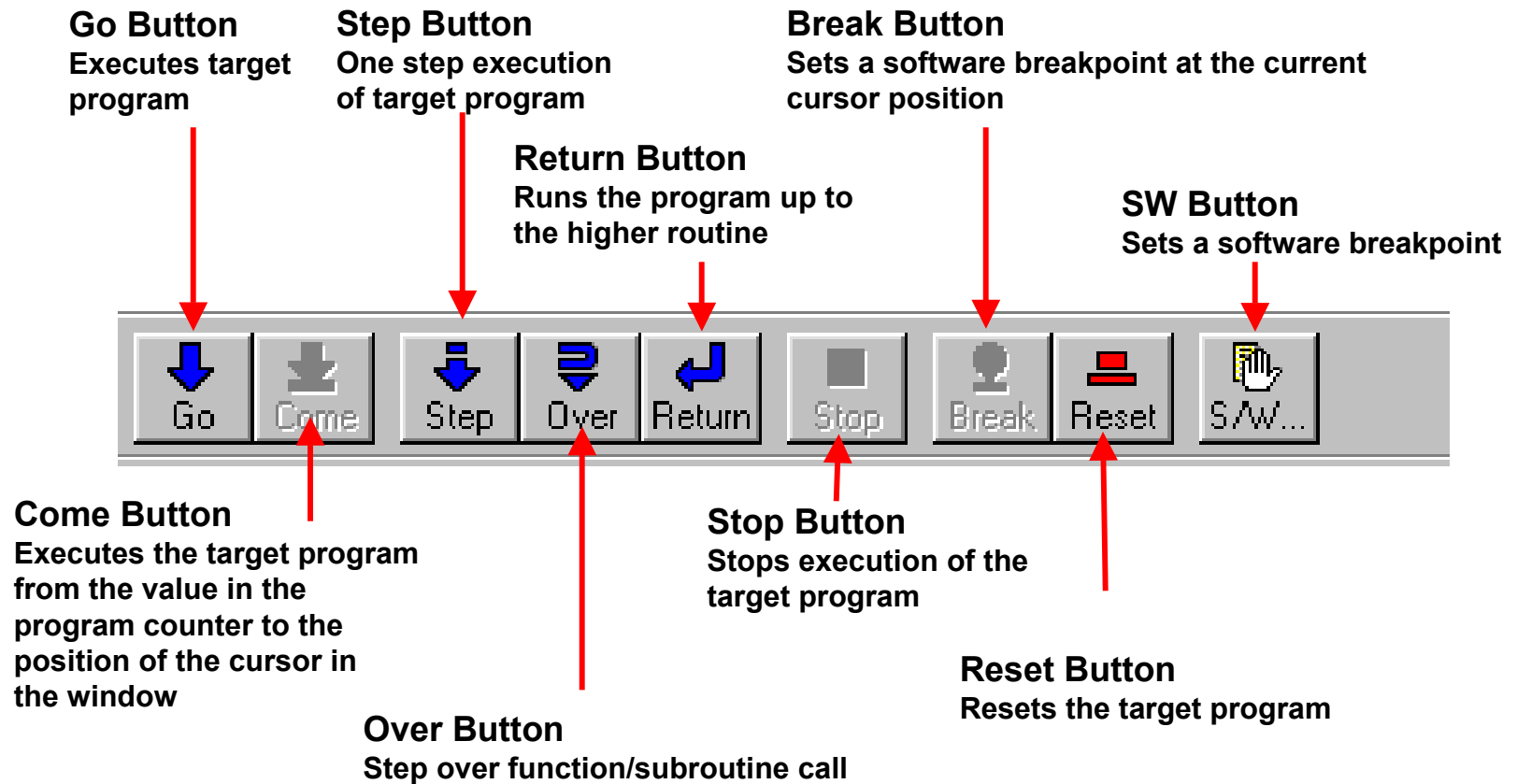
Note 1. See KD30 Help for the differences between Sampling Mode and Free Run Mode. Also, see the ICD User's Manual for details on how ICD works under these two modes.

# KD30 Program Window

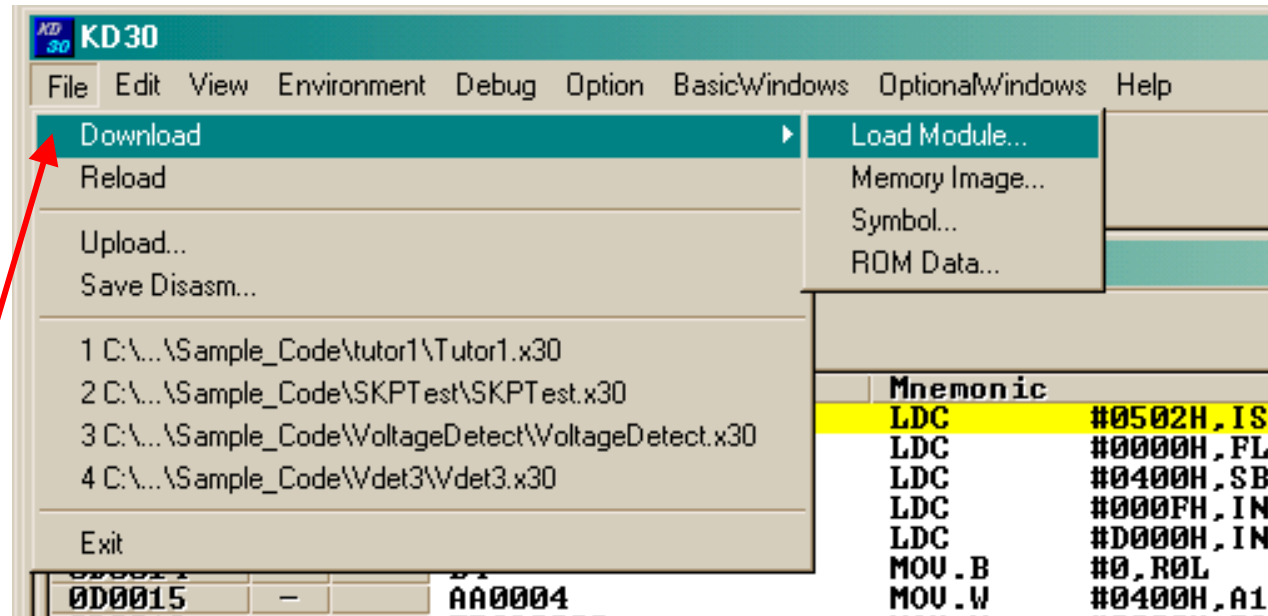


KD30 will disassemble the flash contents or display 'UND' if the flash is blank.

# KD30 Toolbar

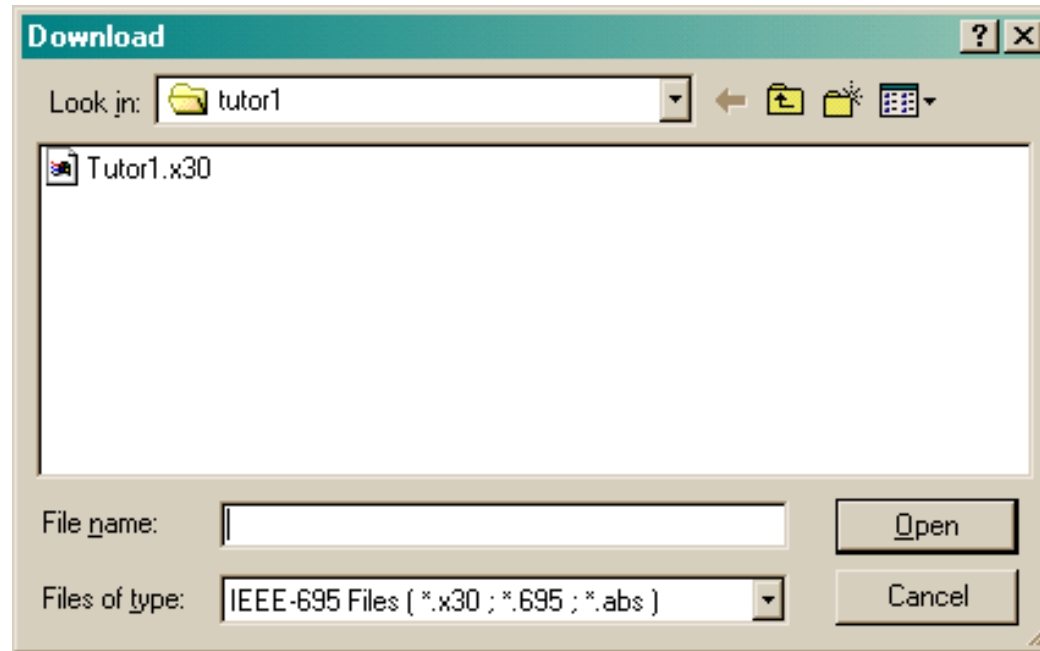


# Download a Program to the SKP16C62P Board (M16C/62P MCU)



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

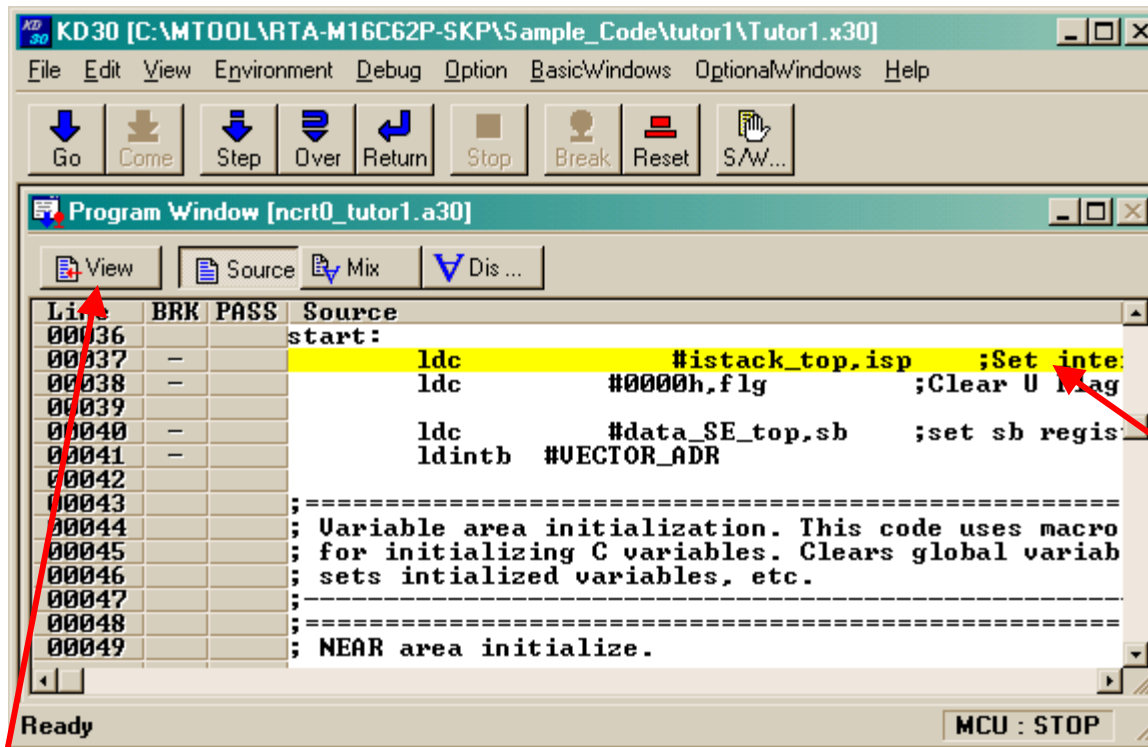
# Download a Program to the SKP16C62P Board (M16C/62P MCU)



In the c:\MTOOL\SKP16C62P\sample\_code\tutor1 directory, select '**tutor1.x30**'.

# Download a Program to the SKP16C62P Board (M16C/62P MCU)

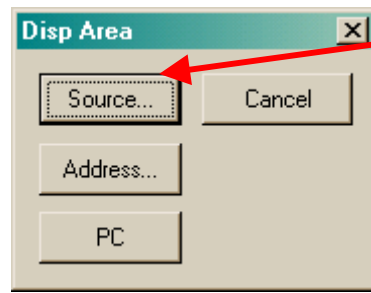
After downloading the program, KD30 opens the file where the reset vector points to.



Current location of MCU program counter is highlighted.

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

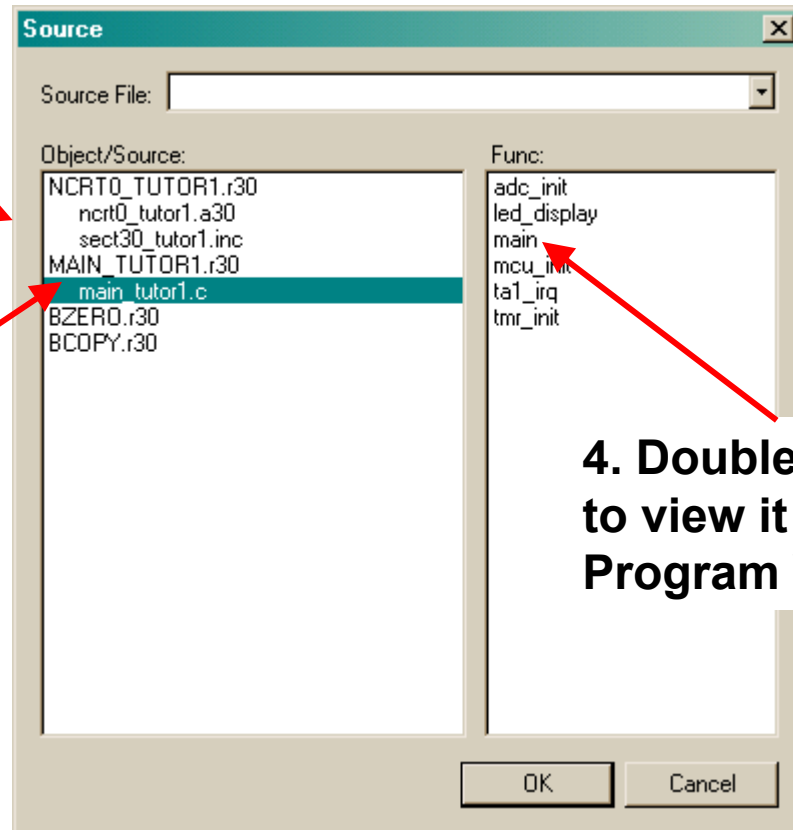
# Viewing Source Files in the Project



1. Click 'source'

2. Source window is displayed.

3. Click 'main\_tutor1.c'

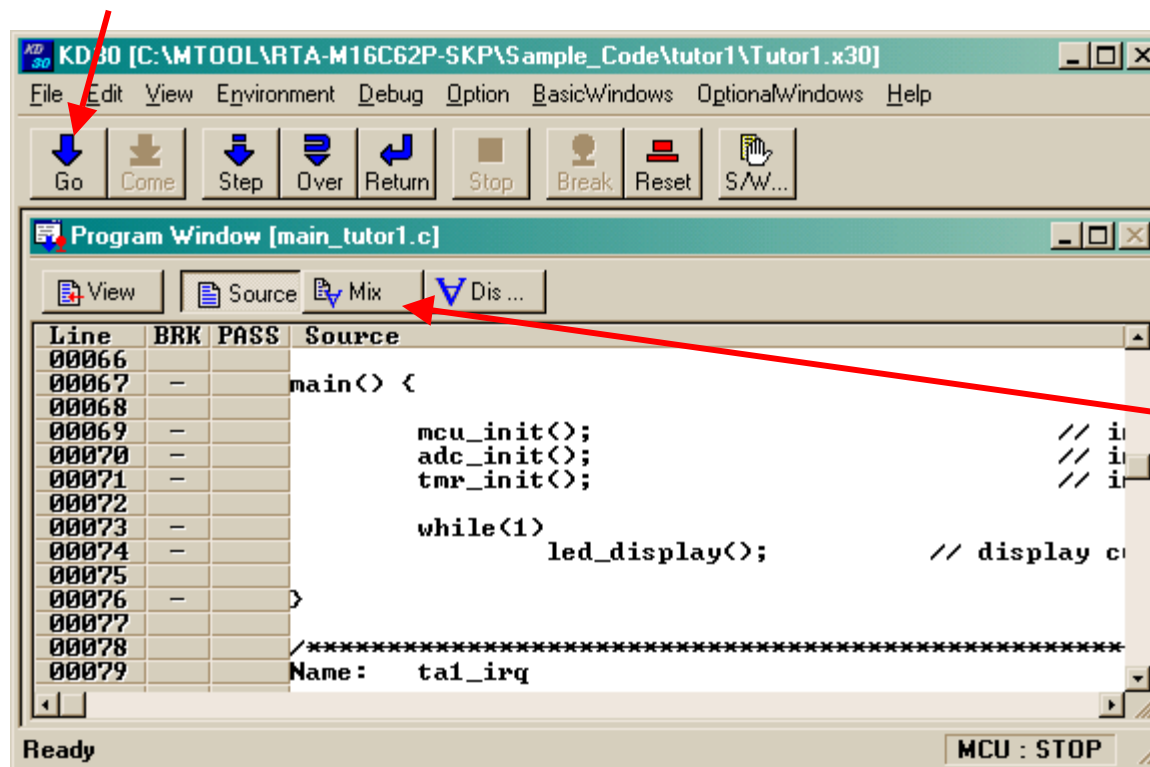


4. Double-click 'main' to view it on the Program Window



# Running Downloaded Program

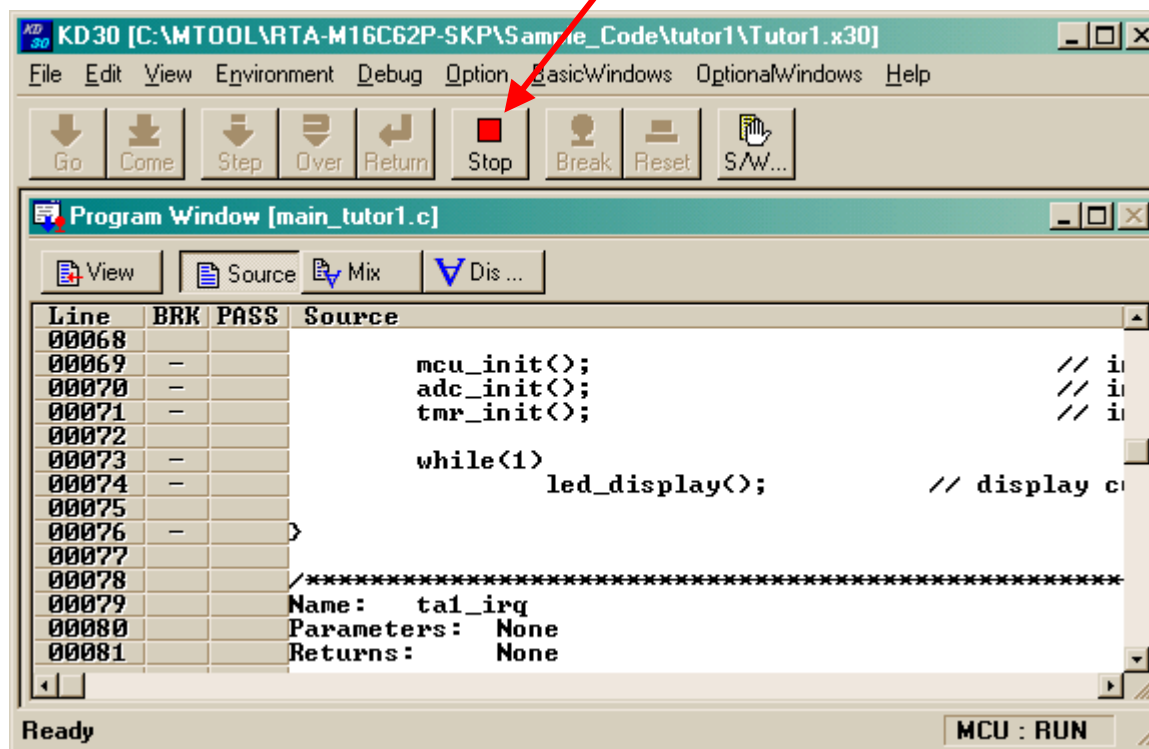
Click on the 'Go' icon to run the tutor1 program you just downloaded. LED's D1, D2, & D3 will blink sequentially. Turning the Analog Adjust potentiometer on clockwise increases the LED blink rate and turning it counter-clockwise decreases the LED blink rate.



Click 'MIX' to view the source code and assembler code.

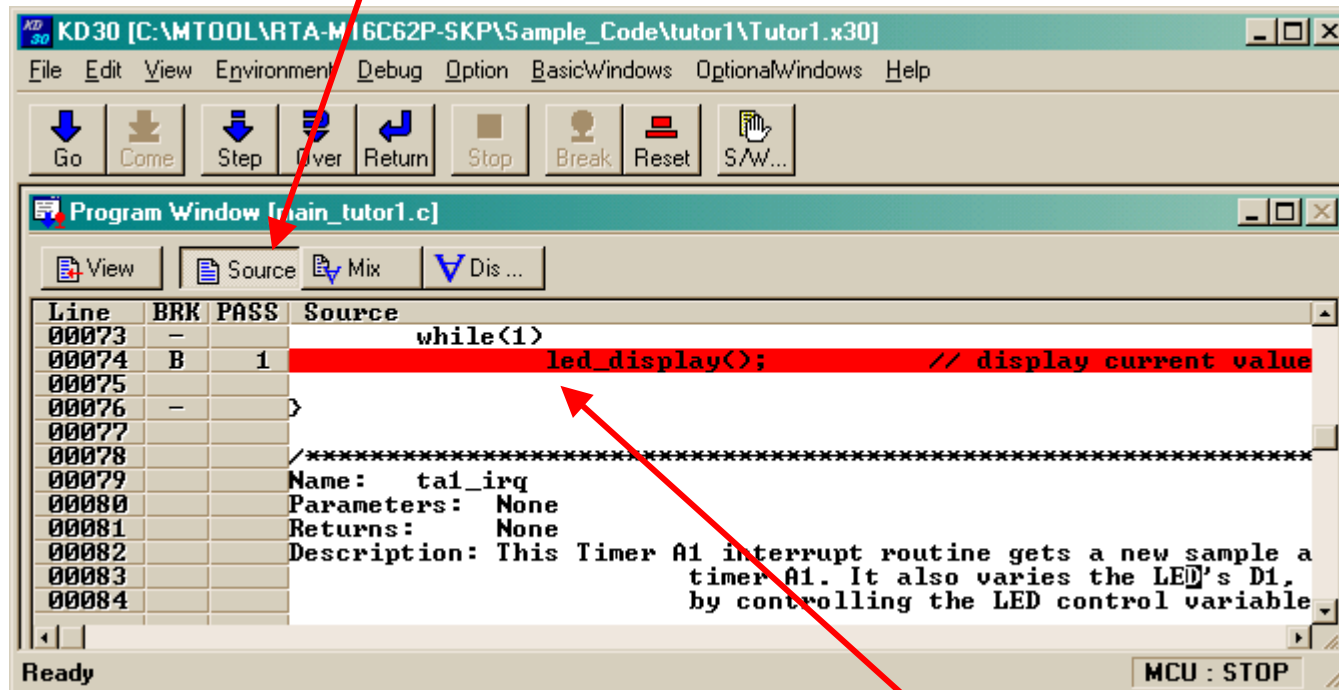
# Stopping Program Execution

Click on the 'Stop' icon to stop the program



# Setting Breakpoints

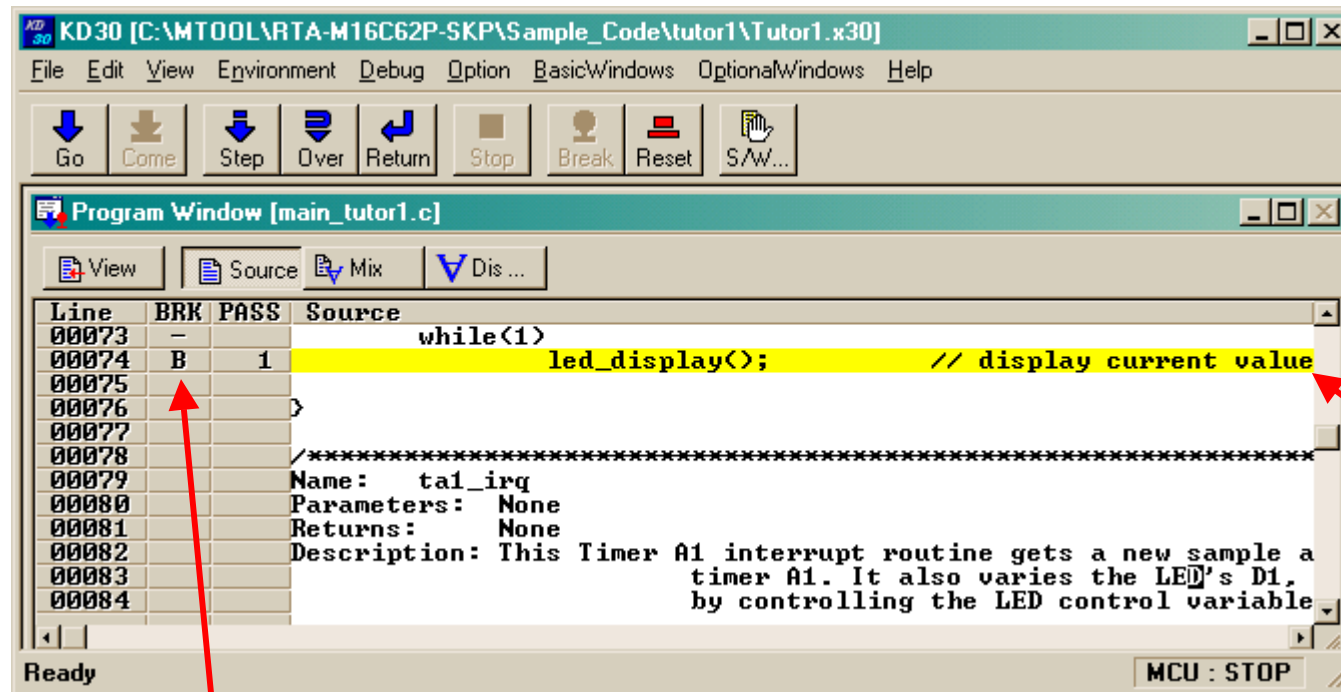
1. Click on the 'Source' to view source code only (not MIX display).



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...

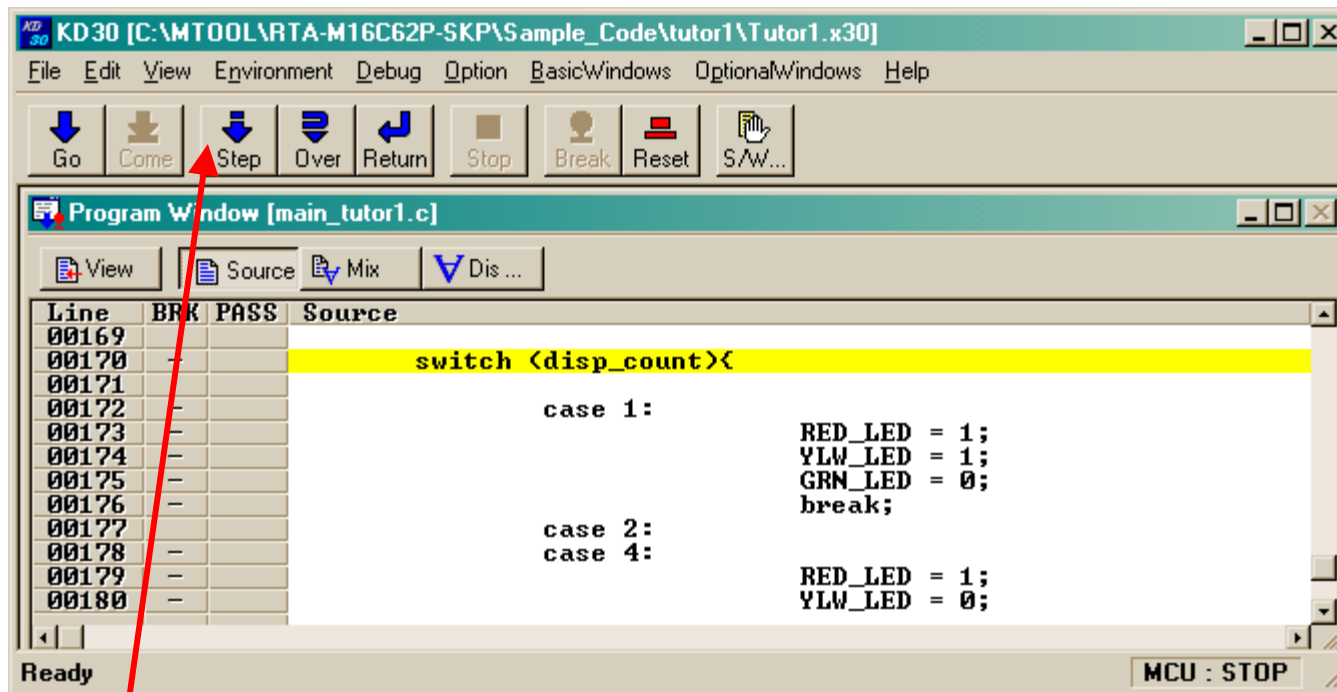
# Removing Breakpoints



Program stops at breakpoint (highlighted in Yellow).

You can remove the breakpoint by double-clicking on it at the 'BRK' column.

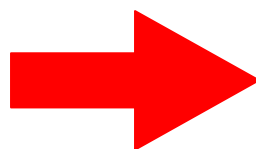
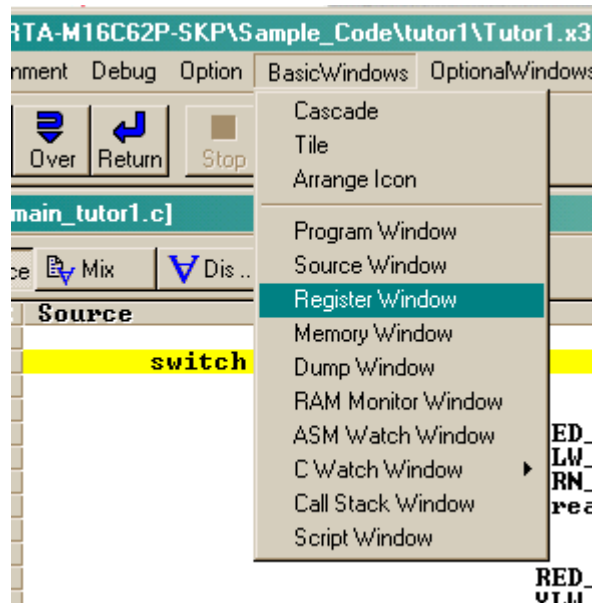
# Program 'Stepping'



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



The Register Window shows a table of CPU registers. The 'Value' column contains hexadecimal values, with some in red to indicate changes. A red arrow points from the text 'Values in red indicate changes since last viewed' to the red value '0D00' in the R1 register row.

Name	Value	R...
PC	0D0176	Hex
R0	0000	Hex
R1	0D00	Hex
R2	0000	Hex
R3	0000	Hex
A0	0000	Hex
A1	0402	Hex
FB	0000	Hex
USP	0000	Hex
ISP	04D4	Hex
SB	0400	Hex
INTB	0FD000	Hex

At the bottom, there is a status bar with labels: IPL, UI, O, B, S, Z, D, C. The values are: 0, 0, 1, 0, 0, 1, 0, 0, 0.

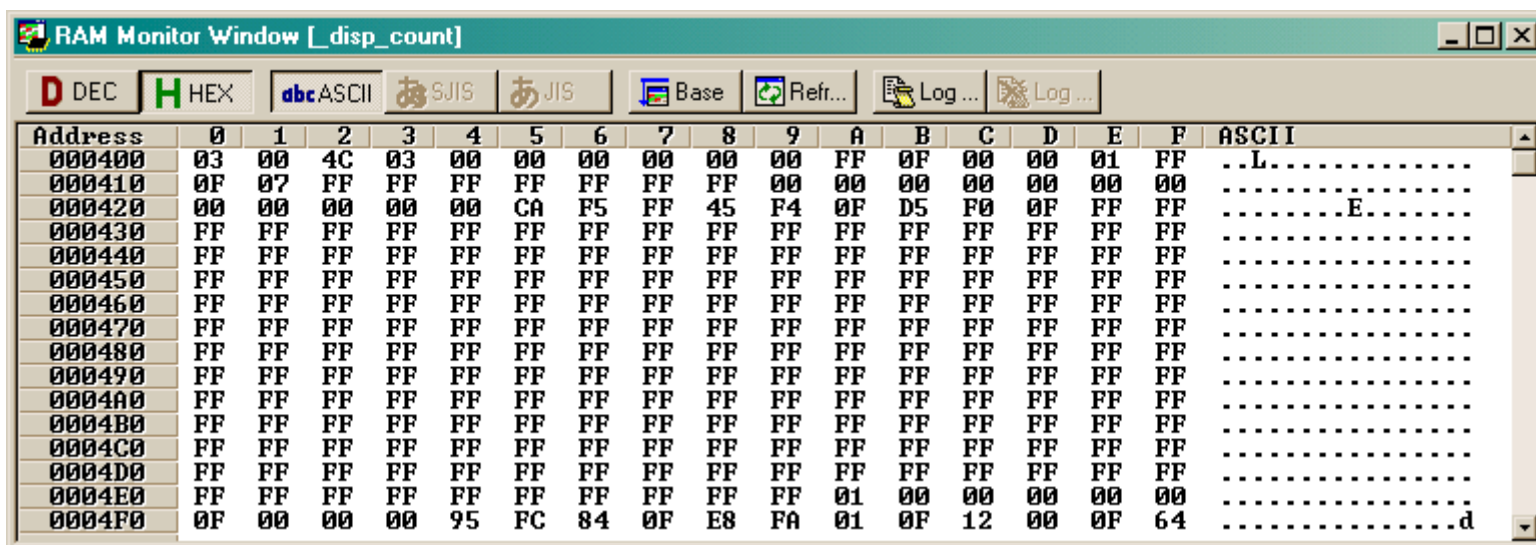
Values in red indicate changes since last “viewed”. Try ‘stepping’ and note the changes.

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

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). 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.

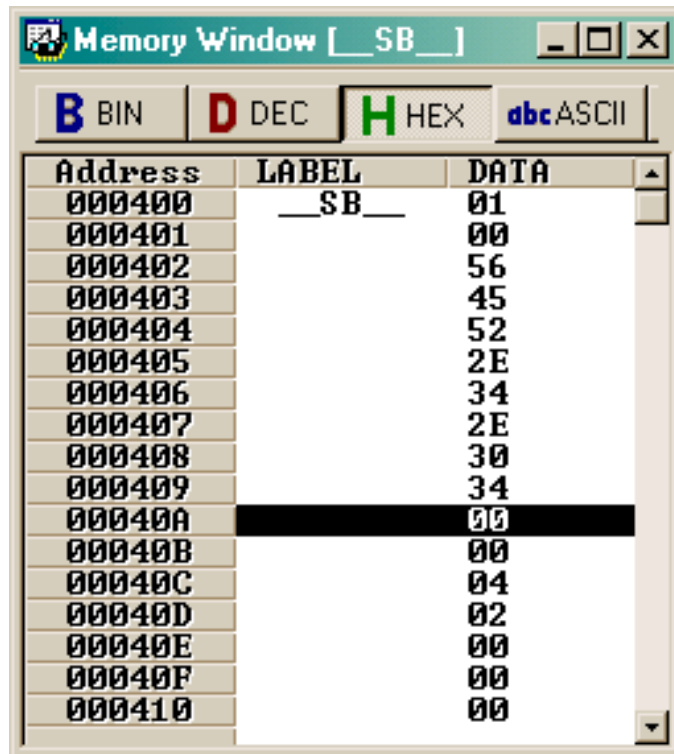


The screenshot shows a window titled "RAM Monitor Window [disp\_count]". It has a menu bar with "D DEC", "H HEX", "abc ASCII", "あ SJIS", and "あ JIS". Below the menu bar are buttons for "Base", "Refr...", "Log ...", and another "Log ...". The main area is a table with columns for Address, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and ASCII. The table displays memory values in hexadecimal and their corresponding ASCII characters.

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	FF	.....E.....
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	64	.....d

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

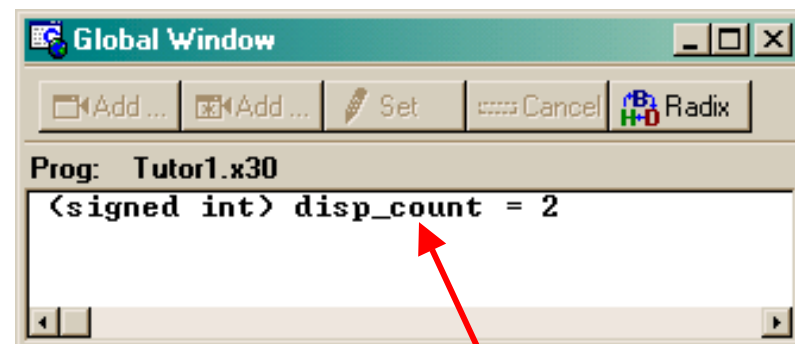


The 'Memory Window' displays a table of memory addresses, labels, and data. The window title is 'Memory Window [ \_SB\_ ]'. It has tabs for BIN, DEC, HEX, and ASCII, with HEX selected. The table has three columns: Address, LABEL, and DATA. The data is shown in hexadecimal format. The address 00040A is highlighted.

Address	LABEL	DATA
000400	_SB_	01
000401		00
000402		56
000403		45
000404		52
000405		2E
000406		34
000407		2E
000408		30
000409		34
00040A		00
00040B		00
00040C		04
00040D		02
00040E		00
00040F		00
000410		00

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

The 'C Watch Window' allows you to view globals and locals. An example is shown below.

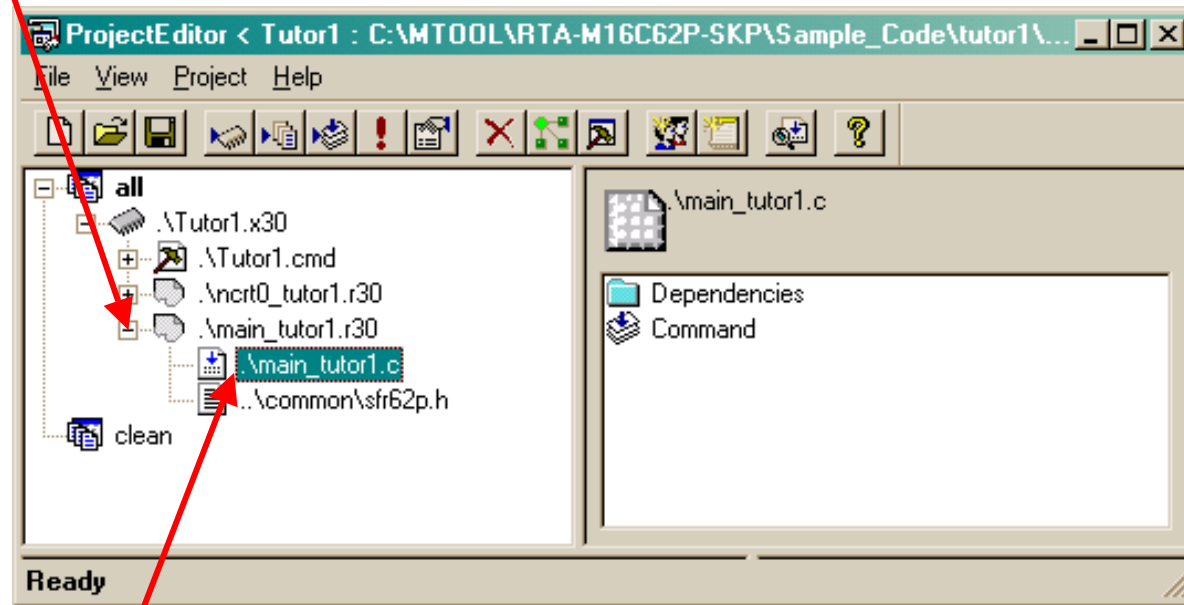


Double-click on the variable to change display format: i.e., change 'char' to 'hex' to 'decimal', etc.



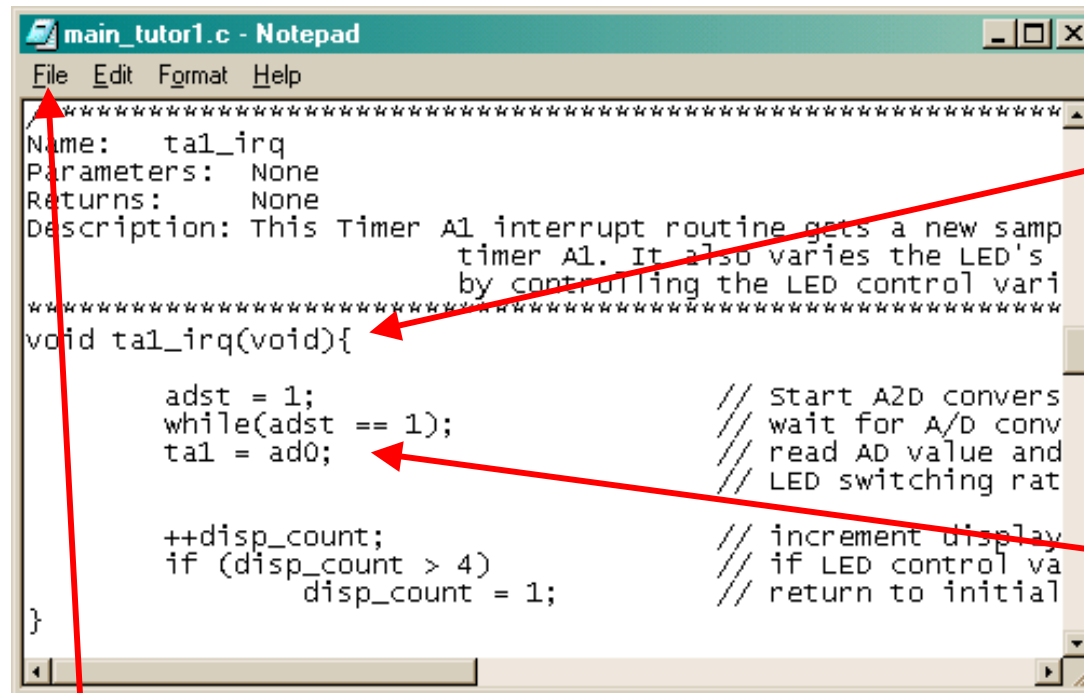
# Modifying the Program: Step 1

In the Project Editor, click on the '+' by `.\main_tutor1.r30`.



Double-click on `.\main_tutor1.c` to edit it.

# Modifying the Program: Step 2



```
main_tutor1.c - Notepad
File Edit Format Help
*****
Name: ta1_irq
Parameters: None
Returns: None
Description: This Timer A1 interrupt routine gets a new samp
              timer A1. It also varies the LED's
              by controlling the LED control vari
*****
void ta1_irq(void){
    adst = 1;
    while(adst == 1);
    ta1 = ad0;

    ++disp_count;
    if (disp_count > 4)
        disp_count = 1;
}

// Start A2D convers
// wait for A/D conv
// read AD value and
// LED switching rat
// increment display
// if LED control va
// return to initial
```

1. Scroll down and find the function 'ta1\_irq' routine.

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

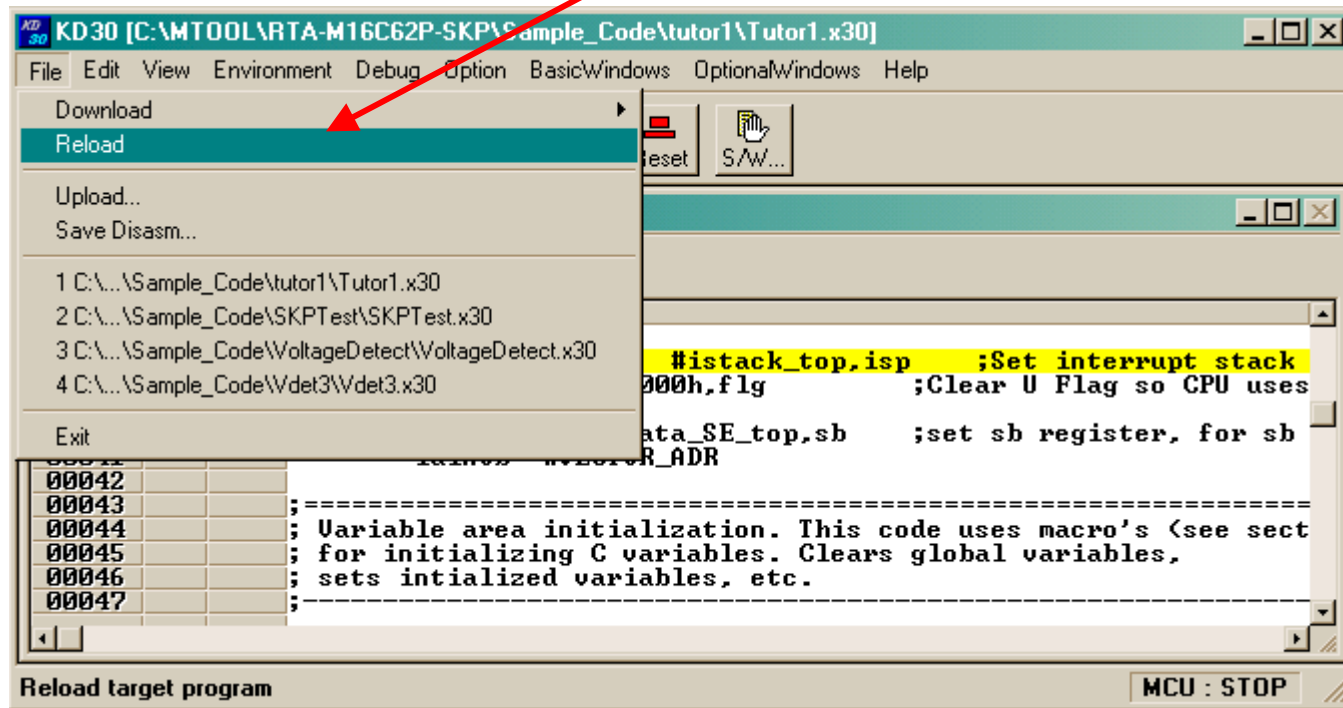
3. Save the revised file (from the 'File' menu)

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 SKP16C62P 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 > SKP16C62P menu.

- **SKP16C62P 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.
- **Tool Manager V3.X User's Manual:** To fully understand and get the most out of “Tool Manager”, this is recommended reading.
- **KD30 Version X.XX User's Manual:** The tutorial only covered the basics of KD30. Read this manual 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/62P Datasheet and SKP16C62P Board Schematic:** These are required to write user application programs for the SKP.
- **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>.

