

Tutorial 1

Introduction to the M16C Development Environment

Introduction

The purpose of this tutorial is to get you acquainted with the M16C development environment, namely the Tool Manager and KD30 debugger.

In order to complete this tutorial, you will need to download the associated `m16c_tut1.zip` file which contains the source code example plus other associated software. All files referred to in this tutorial are contained in this zip file.

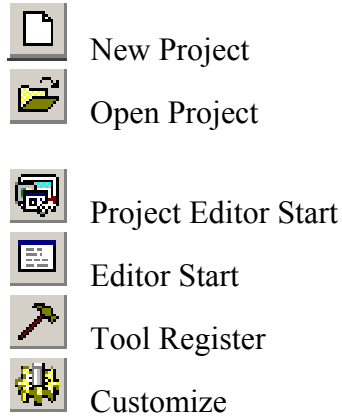
After completing this tutorial, you should be able to:

- Create a new project
- Add and remove files from a project
- Compile a Program
- Launch the KD30 debugger
- Run a program
- Re-flash the M16C Monitor program

1 The Tool Manager



The screenshot above shows the main Tool Manager (TM) toolbar. The toolbar icons of most interest to us are:



The four icons above are the Partial Build, Build, Rebuild and Debug icons respectively.

11. Configuring the TM environment

The first thing we want to do is make sure the TM environment is properly configured.

11.1.□. Configuring the Debug Tool

Later in this tutorial, we will use the KD30 debugger to run and test our programs on the M16C development board. The Tool Manager has the ability to automatically launch KD30 and download your program to the M16C at the click of a button.

To configure the Debug Tool:

- Click the **Tool Register** icon to open the Tools Information dialog
- Under DEBUG TOOL, select KD30 from the list of available debuggers.

12. Configuring a Custom Text Editor

By default, TM uses Notepad as its default text editor. Notepad works fine for simple test programs, but after a while you may find the Notepad interface to be a little

primitive, particularly when you begin editing complex programs that contain more than one or two files. In this case, a MDI editor would be better suited for the task.

There are a wide variety of free text editors available on the Internet. You can download and install a custom text editor, and then configure TM to use your chosen editor by default.

To configure the Tool Manager to a custom text editor:

- Click the **Tool Register** icon to open the Tools Information dialog.
- Navigate to the EDIT TOOL tab.
- Click the **Add...** button to add a new editor.

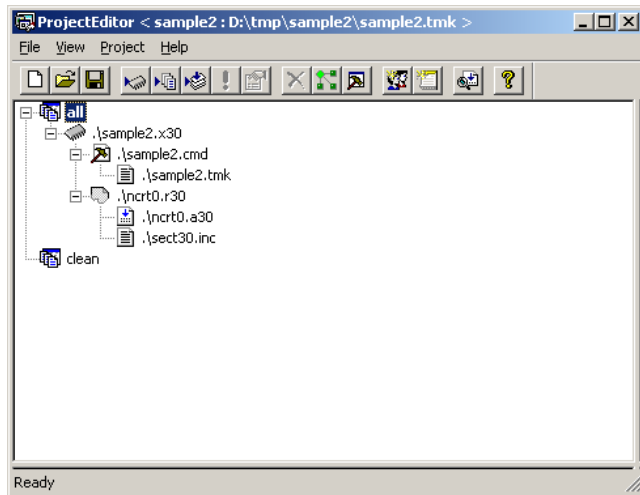
1 Creating a new project

Now that our environment is properly configured, we can proceed to the next step: Creating a New Project.

- Click the New Project icon (looks like a blank sheet of paper) to open the New Project Wizard.
- Enter the project name (required) and a working directory (if desired). We will be using the project name “sample2” for this tutorial. Click Next
- Select C Project. Next.
- Select “NC30WA V.5.20 Release 1” as the compiler package. This should be the default. Click Next.
- Review the project specifications and click Finish to create the project.

2 The Project Editor

The New Project Wizard will automatically launch the Project Editor. The screenshot below shows an expanded view of the default hierarchy of a newly created project.



There are two top-level targets: “all” and “clean”.

The “all” target contains a hierarchy of sub-targets that will be compiled when you build your project. The “all” target contains two sub-targets: “sample2.x30”, the main compilation target; and ncr0.r30, the M16C C runtime library. The “sample2.x30” target is the file that will be generated by the compiler when you build your project later in the tutorial. The C runtime library contains the M16C bootstrapping code, the initialization routines that prepare the microprocessor for running our C programs.

For the purpose of this tutorial, you need only be concerned with “sample2.x30” for now. It contains the executable program code that will be downloaded to the M16C when you debug your program later on.

The “clean” target contains instructions for the build system to remove stale files from previous compilations. This is used when a project is compiled using “Rebuild”. You do not need to edit or modify this target.

The Project Editor icons of interest to us are:



New Project



Open Project



Save Project



Add File



Scan All Dependencies

3 Adding Files to the Project

Before continuing, copy the files from Resources/sample2 into the working directory for your project. There are 5 files in total:

- leddata.h – header file for the 7-segment display
- ncrt0.a30 – the M16C start-up code
- sample2.c – the sample2 program
- sect30.inc – includes the interrupt vector table
- sfr62.h – standard M16C register/port definitions

Make sure that all 5 files are copied. The versions of “ncrt0.a30” and “sect30.inc” that TM inserted into your project *will not work properly* with the debugger by default. You must overwrite these files with the new versions from the resources directory.

Now, add the files to the project:

- In the Project Editor window, select and expand the “sample2.x30”.
- Highlight the “ncrt0.a30” target and hit Delete to remove this target from the project.
- Click “Add Files”. Browse to the working directory for your project. Select “ncrt0.a30” and “sample2.c” for inclusion. Alternatively, you may right-click the “sample2.x30” target and select Edit Item → Add File from the pop-up menu.
- Click the “Scan Dependencies” icon. This will scan the project source files for dependencies (header files, includes, etc) and automatically include them in the project.

4 Compiling the Project

Once you have added the required files, click “Rebuild” in the TM window to compile the project.

This will create a file **sample2.x30** in your working directory.

2 Using the KD30 Debugger

Now that the sample2 program is compiled, we will download it to the M16C development board and use the KD30 debugger to run the program.

But first, a word of advice:

- ALWAYS use the reset button on the M16C board to reinitialize the M16C microprocessor BEFORE downloading your program to the development board.

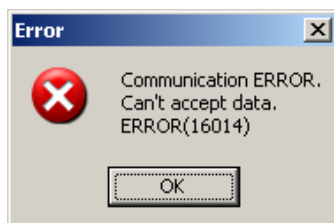
Not resetting the M16C before downloading your program is almost guaranteed to cause KD30 to lock-up. As you will learn, KD30 is not a very robust piece of software; it is fond of locking-up. If this happens, terminate the KD30 process using the Windows task manager (CTRL+SHIFT+ESC), reset the M16C, and then restart the debugger to continue working.

5 Launching the KD30 Debugger

- Ensure the serial cable from the PC to M16C development board is connected.
- Hit the “Reset” button on the M16C development board.
- Click the “Debug” icon in the main TM window to launch the KD30 debugger.
- An initialization dialog will appear. Ensure that the serial port and baud rate are set to “COM1” and “38400” respectively.
- Click OK to start KD30.

21. Communication Error

If the KD30 debugger cannot connect to the M16C then the following dialog will be displayed:



If the communication error dialog is displayed, check the following items before proceeding:

- Check the serial connection between the PC and the M16C

- Check the power to the M16C board
- Check that no other instances of KD30 are running
- Check that no other program is using COM1

If all the items on the bullet list above are correct and KD30 still cannot connect to the M16C, try the following:

- Power the M16C off, then back on again. Hit the reset button.

If KD30 *still* refuses to connect to the M16C, there may be a problem with the M16C monitor program.

The M16C monitor program is a piece of software that facilitates communication with the KD30 debugger. The monitor program resides in flash-memory on the M16C board; it must be both present and of the correct version if KD30 is to successfully connect to the M16C. If the monitor has been removed from flash, or, as happens occasionally, has become corrupted, then it will be necessary to re-flash the monitor software onto the M16C board. The section titled “Re-Flashing the M16C Monitor” at the end of this document provides instruction for repairing a corrupted monitor.

If the monitor re-flashing procedure fails, there is probably a hardware fault with your development kit. Please consult a lab demonstrator.

6 Downloading your Program to the M16C

The next step is to download the compiled sample2 binary to the M16C development board. Downloading the program to the M16C board will store a copy of your program code in flash-memory on the M16C, ready for execution.

- In KD30 select File → Download → Load Module...
- Browse to the project working directory and select “sample2.x30”.

7 Running the Program

Click the “Go” button to start executing the sample2 program.

If everything worked properly, you should now see LED2 on the development board cycling through the digits 0-9.

This is the end of Tutorial #1. Thanks to Jacob Hart for his significant help in developing this tutorial.

3 Re-Flashing the M16C Monitor

If the debugger repeatedly fails to connect to the M16C, it may be necessary to re-flash the monitor software.

You will need a copy of the M16CFlasher utility and the Monitor program (mon_uart.mot). Both are available in the resources directory.

1. Disconnect power from the M16C development board.
2. Close the CNVSS jumper on the development board. It is located at the top-left of the board near the power input.
3. Reapply power to the board.
4. Run the M16CFlasher utility.
5. Click “Settings”. Select serial port settings “COM1” and “57600”. Click close.
6. Click “Connect”. You should receive a message:

```
Connecting at 9600...Ok.  
Switching to 57600...Ok.  
Reading version information...VER.X.XX...Check Ok.
```

7. Click the down-arrow next to the Prog button and choose “Prog. new file...”
8. Choose “mon_uart.mot”.
9. If programming was successful, you will receive a message similar to the following:

```
Reading mon_uart.mot...Ok. (0x0FE000 - 0x0FFFFFF)  
Erasing ALL blocks...Ok.  
File: mon_uart.mot (date time)  
Programming (0x0FE000 - 0x0FFFFFF)...CRC (8A18) Ok, Programming Ok.
```

10. Close the M16Flasher utility.
11. Disconnect power from the development board and open the CNVSS jumper.