

How to develop C programs Metrowerks/Tech Arts 9S12C32 board

Jonathan Valvano, valvano@mail.utexas.edu, August 23, 2008

Installing Metrowerks

CodeWarrior Version 3.1 will compile programs we need for EE319K/EE345L/EE345M. For version 3.1 you should use their 12K free educational license. There is an installer for Version 3.1 on the CD accompanying the second edition of the EE345L/EE345M textbook. Follow these steps to install the Special edition of Metrowerks CodeWarrior Version 4.7 (32 KiB free educational license for C code)

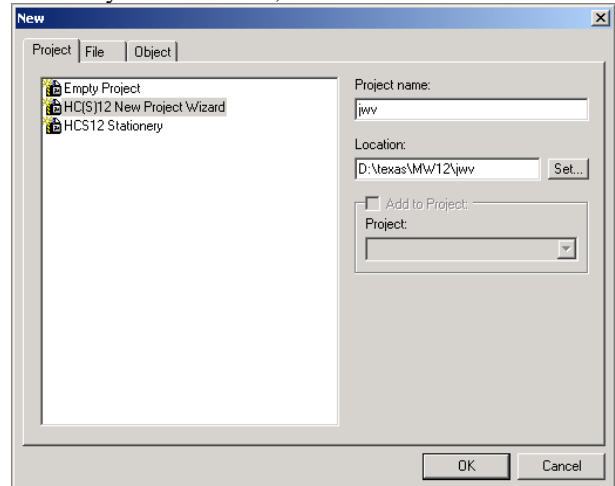
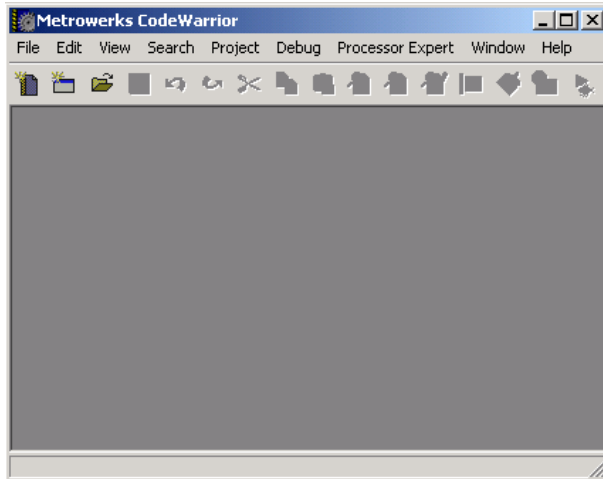
- 1) <http://www.freescale.com/>
- 2) click "**CodeWarrior Development Tools**" under Products
- 3) click "**HCS12(X)**" under "CodeWarrior Development Studio for"
- 4) In the "Special edition" column, click "**Get**".
- 5) Click "**Download**" button for "Special Edition Evaluation: CodeWarrior Development for HCS12X Microcontrollers V4.7"
- 6) If you have to, register as a new user (if you have registered before, just log in)
 - email must be correct
 - decide whether or not you want email from Metrowerks
 - Fill in the page with "project details" stating you are a student taking a class, fill in all required fields
- 7) Download CWHCS12X_v4_7_Special.exe (254 MB) and install (the special edition does not require downloading a separate license)
- 8) Download instructions and starter projects from my web site at <http://users.ece.utexas.edu/~valvano/metrowerks/>

A) To open an existing Metrowerks project

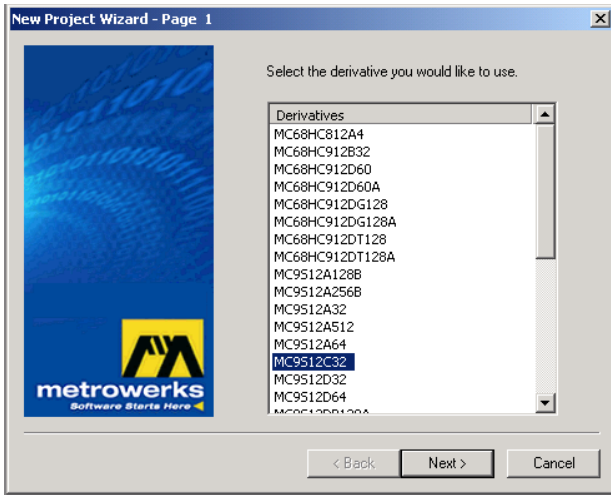
- 1) Start Metrowerks CW12
- 2) Execute File->Open, navigate to an existing *.mcp file, and click "OK"

B) How to configure create a new Metrowerks project

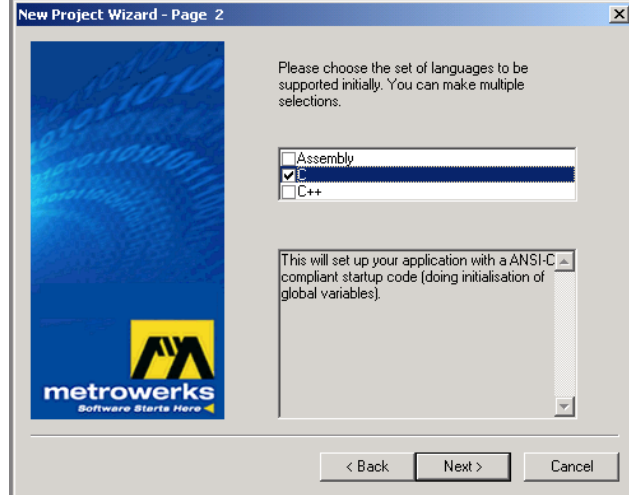
- 1) Start Metrowerks CW12
- 2) Execute File->New, click "Project" Tab
 - select "HC(S)12 New Project Wizard"
 - specify the "Project name"
 - verify its "Location", click "OK"



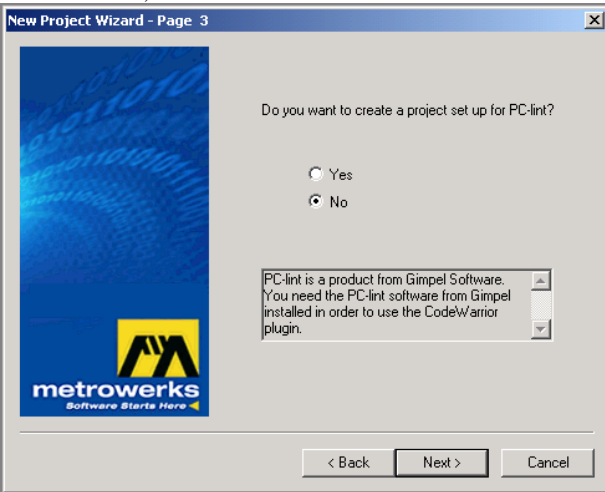
3) Select the derivative you want to use "MC9S12C32", click "next"



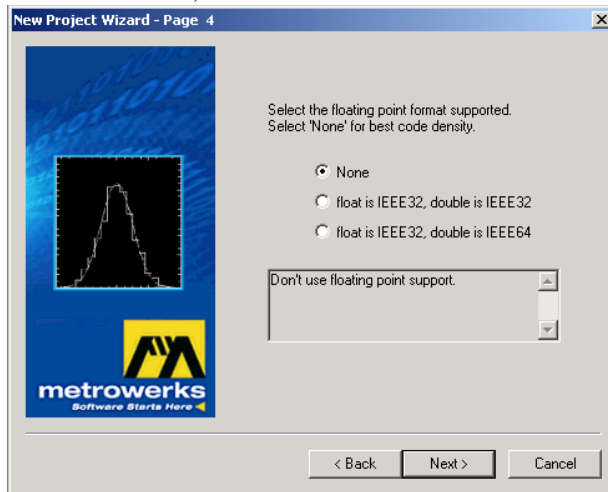
4) Choose the set of languages supported select "C", click "next"



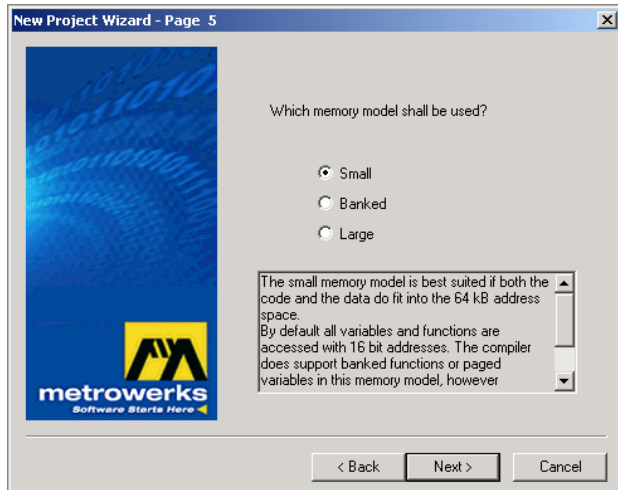
5) Do you want to create a setup for PC-lint select "no", click "next"



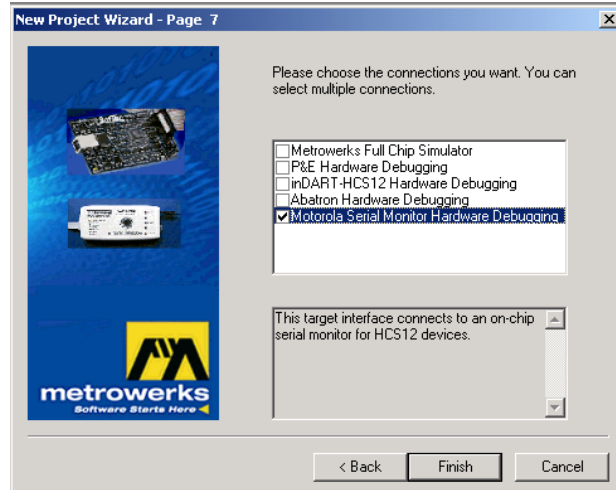
6) Select the floating point format supported select "None", click "next"



7) Which memory model should be used? select "Small" click "next"



8) Please choose the connections you want select "Serial Monitor Hardware Debugging" click "Finish"

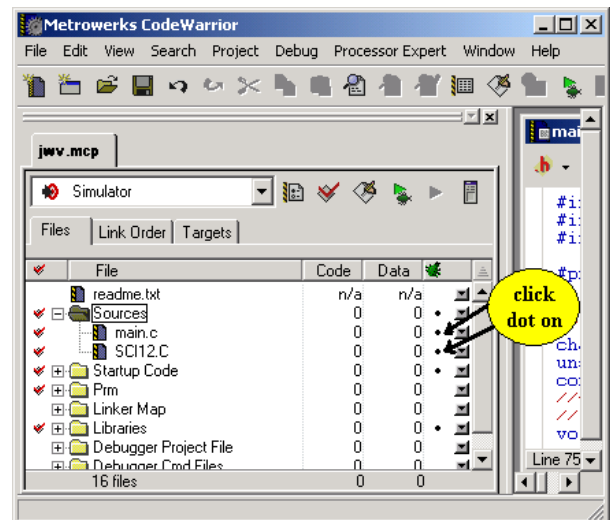
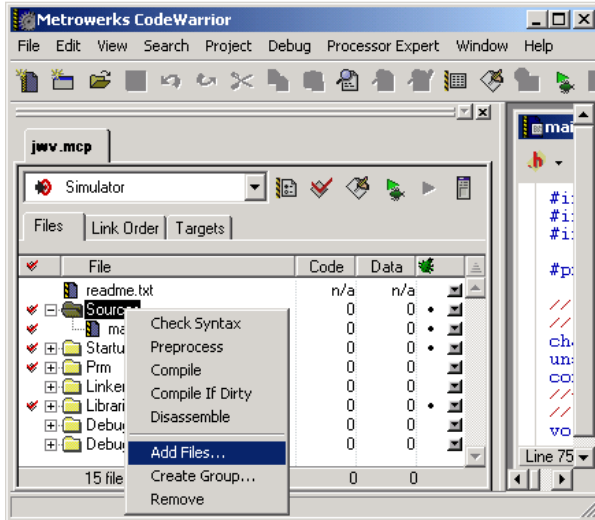


9) Create or copy program files ***.c** and ***.h**
 place them into the **"Sources"** directory of your project

10) Add the necessary C files to project

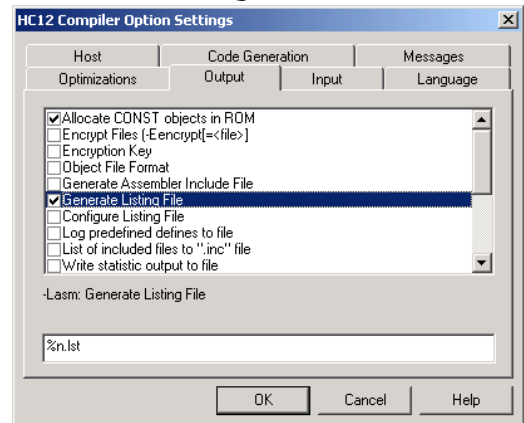
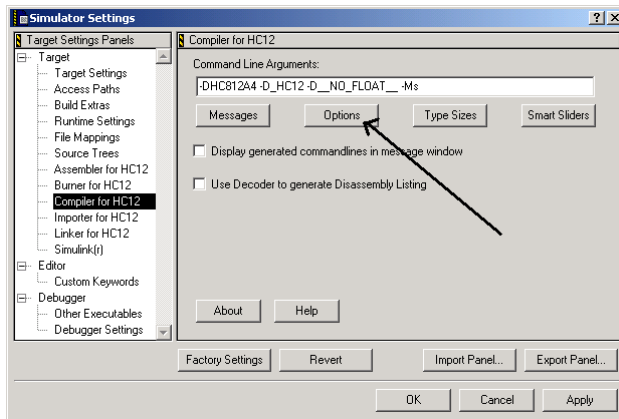
click on Sources in the **"mcp"** window
 right click and execute **"Add Files..."**
 "click dot on" in the field associated all C source files under the "bug" icon

Metrowerks adds the file **datapage.c**, which is not needed. So, this file can be deleted from the project.

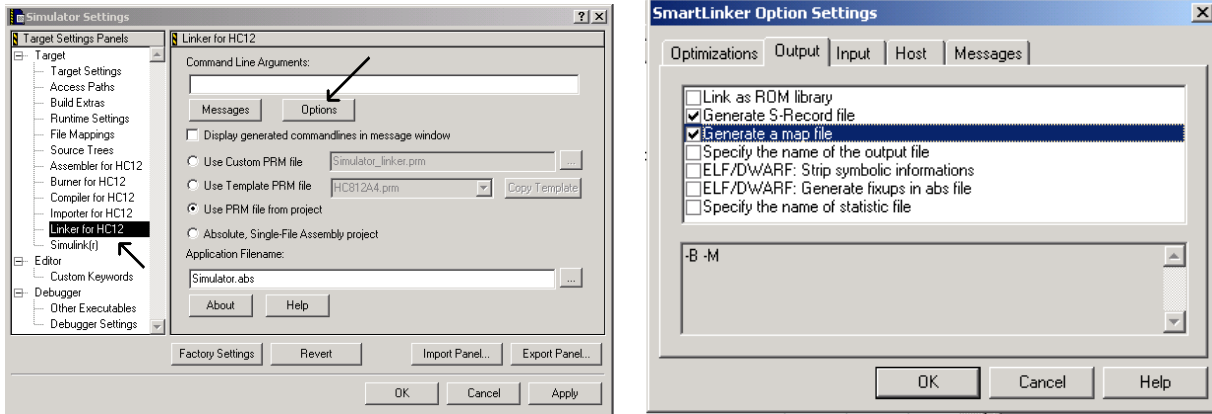


13) Change compiler/linker options

click the right-most toolbar ICON called **"simulator settings"**
 click **"Compiler for HC12"** choice
 click **"Options"**
 click **"Output"** tab
 select **"Allocate CONST objects in ROM"** and **"Generate Listing File"**



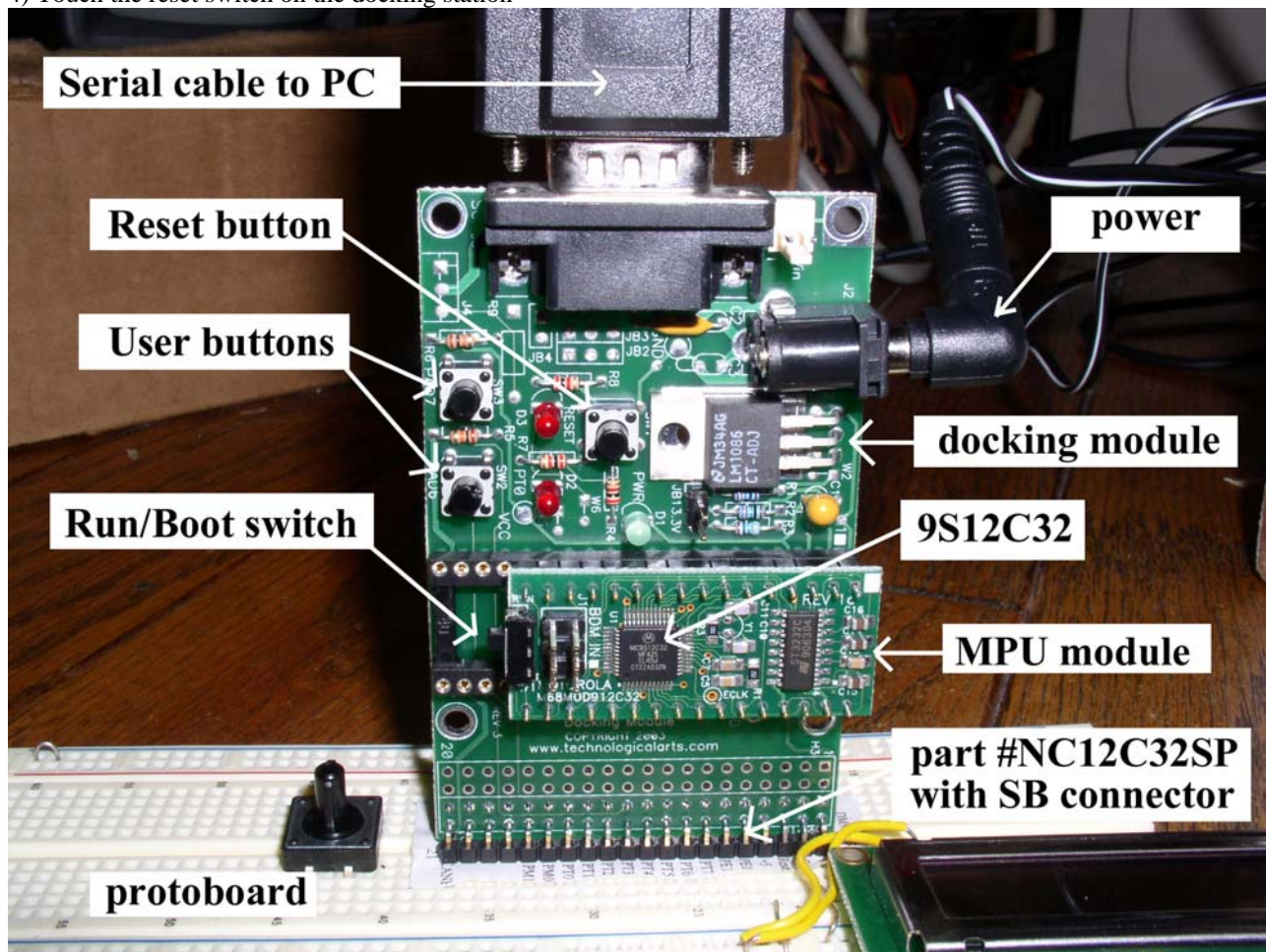
click **"Linker for HC12"** choice
 click **"Options"**
 click **"Output"** tab
 select **"Generate S-Record"** and **"Generate a map file"**



C) How to run Metrowerks on the Real 9S12C32 board

Do this once

- 1) Connect PC-COM1 to the 9S12C32 docking station,
- 2) Place the Run/Boot or Run/Load switch on the 9S12C32 board in Load mode
- 3) Connect power to 9S12C32 docking station.
- 4) Touch the reset switch on the docking station



For each edit/compile/run cycle for software that does not use SCI

- 1) In Metrowerks, perform editing to source code
- 2) In Metrowerks, compile/Link/Load
Execute **Project->Debug**
- 3) Click the green arrow in the debugger to start. Runs at 24 MHz.
- 4) Quit the debugger when done (it doesn't work when you have more than one debugger window open).

For each edit/compile/run cycle for software that does use SCI

- 1) set the Run/Boot switch to Boot mode, push the reset button on the 9S12C32 docking station
- 2) execute Project->Debug (compiles and downloads code to 9S12C32)
- 3) quit MW debugger after programming is complete. Quitting the debugger will release the COM port.
- 4) start a terminal program (like HyperTerminal)
specify proper COM port, 19200 bits/sec, no flow control (match the baud rate of your 6812 software)
- 5) set the Run/Boot switch to Run mode and push the reset button on the 9S12C32 docking station.
Runs at 4 MHz if you do not modify the PLL.
You can adjust the E clock rate by configuring the PLL
- 6) when done, quit terminal program. Quitting the terminal program will release the COM port.

To run in embedded mode

- 1) disconnect the serial cable if not needed
- 2) set the Run/Boot switch to Run mode,
- 3) apply power to the 9S12C32 docking station
Runs at 4 MHz if you do not modify the PLL.
You can adjust the E clock rate by configuring the PLL

Add this code to your project, we you wish to run at 24 MHz in both Run and Boot modes

```
//***** PLL_Init *****
// Set PLL clock to 24 MHz, and switch 9S12 to run at this rate
// Inputs: none
// Outputs: none
// Errors: will hang if PLL does not stabilize
void PLL_Init(void){
    SYNCR = 0x02; // OSCCLK is 8 MHz Crystal Clock Frequency
    REFDV = 0x00; // PLLCLK = 2 * OSCCLK * (SYNR + 1) / (REFDV + 1)
    CLKSEL = 0x00; // PLLCLK of 24 MHz with 8 MHz crystal
// Meaning for CLKSEL:
// Bit 7: PLLSEL = 0 Keep using OSCCLK until we are ready to switch to PLLCLK
// Bit 6: PSTP = 0 Do not need to go to Pseudo-Stop Mode
// Bit 5: SYSWAI = 0 In wait mode system clocks stop.
// Bit 4: ROAWAI = 0 Do not reduce oscillator amplitude in wait mode.
// Bit 3: PLLWAI = 0 Do not turn off PLL in wait mode
// Bit 2: CWAI = 0 Do not stop the core during wait mode
// Bit 1: RTIWAI = 0 Do not stop the RTI in wait mode
// Bit 0: COPWAI = 0 Do not stop the COP in wait mode
    PLLCTL = 0xD1;
// Meaning for PLLCTL:
// Bit 7: CME = 1; Clock monitor enable - reset if bad clock when set
// Bit 6: PLLON = 1; PLL On bit
// Bit 5: AUTO = 0; No automatic control of bandwidth, manual through ACQ
// Bit 4: ACQ = 1; 1 for high bandwidth filter (acquisition); 0 for low (tracking)
// Bit 3: (Not Used by 9s12c32)
// Bit 2: PRE = 0; RTI stops during Pseudo Stop Mode
// Bit 1: PCE = 0; COP disabled during Pseudo STOP mode
// Bit 0: SCME = 1; Crystal Clock Failure -> Self Clock mode NOT reset.
    while((CRGFLG&0x08) == 0){ } // Wait for PLLCLK to stabilize.
    CLKSEL_PLLSEL = 1; // Switch to PLL clock
}
```